**CHAPTER 10: APPLICATION TO THE FINANCIAL MARKETS**

* Creación de neuronas de entrada y salida para predicción
* Cómo crear conjuntos de entrenamiento para una red neuronal predictiva
* Predicción de la ola sinusoidal

En el último capítulo, vio que las redes neuronales se pueden utilizar para predecir tendencias en datos numéricos, como en el ejemplo de onda sinusoidal. Predecir la onda sinusoidal fue útil para demostrar cómo crear redes neuronales que puedan predecir, pero tiene poca aplicación en el mundo real. El propósito del capítulo 9 era introducir los fundamentos de cómo predecir con una red neuronal. Este capítulo se basa en el material presentado en el capítulo 9 proporcionándole una base para aplicar redes neuronales a problemas del mercado financiero.

En este capítulo, se presenta un programa relativamente simple que intenta predecir el índice S&P 500. La palabra clave de la última oración es "intentos". Este capítulo es sólo para fines educativos y no es de ninguna manera una estrategia de inversión, ya que el rendimiento pasado no es una indicación de rendimientos futuros. El material presentado aquí se puede utilizar como punto de partida desde el que adaptar las redes neuronales para aumentar su propia estrategia investment.

# Recopilación de datos para la red neuronal S&P 500

Antes de discutir cómo predecir la dirección en el S&P 500, primero debemos aclarar cuál es el S&P 500 y cómo funciona.

"El S&P 500 es un índice bursátil que contiene las acciones de 500 grandes corporaciones, la mayoría de las cuales son estadounidenses. El índice es el más notable de los muchos índices propiedad y mantenidos por Standard &Poor's, una división de McGraw-Hill. S&P 500 se utiliza en referencia no sólo al índice, sino también a las 500 empresas que tienen sus acciones comunes incluidas en el índice." (Desde www.wikipedia.org)

La Figura 10.1 muestra el movimiento en el S&P 500 desde 1950.

### Figura 10.1: El índice bursátil S&P 500 (De www.wikipedia.org).

S&P 500

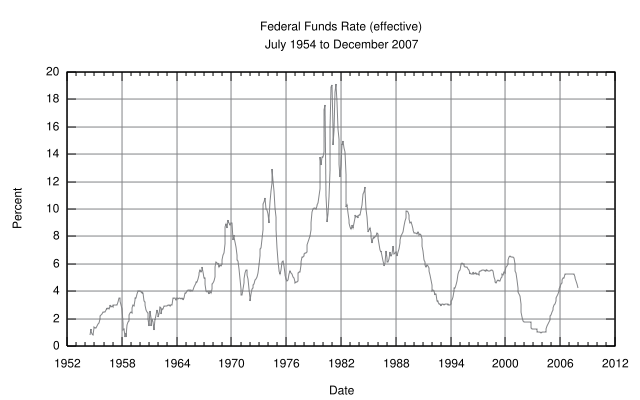
50 1960 1970 1980 1990 2000

Los valores históricos de S&P 500 se utilizarán para predecir futuros valores S&P 500; sin embargo, los datos del S&P 500 no se considerarán en el vacío. El tipo de interés preferencial actual también se ajustará para ayudar en la detección de patrones. El tipo de interés preferencial se define de la siguiente manera:1

"La tasa preferencial es un término aplicado en muchos países a un tipo de interés de referencia utilizado por los bancos. El término indicaba originalmente la tasa de interés a los bancos prestado a los clientes favorecidos, [...] aunque esto ya no siempre es el caso. Algunas tasas de interés variables pueden expresarse como un porcentaje por encima o por debajo de la tasa preferencial." (Desde www.wikipedia. org)

La Figura 10.2 muestra la tasa de interés preferencial de los Estados Unidos con el tiempo.

### Figura 10.2: Tasa de interés preferencial estadounidense (Desde www.wikipedia.org).



La red neuronal presentada en este capítulo debe proporcionarse tanto con los datos históricos del S&P 500 como con los datos históricos de los tipos de interés preferencial. El programa está diseñado para recibir ambas entradas de datos en archivos de valores separados por comas (CSV).

**Obtención de datos** **históricos** **de S&P** **500**

Al descargar los ejemplos de este libro, también descargará un conjunto de datos históricos de S&P 500. Los datos facilitados estaban actualizados a mayo de 2008. Si desea obtener datos financieros más actuales, puede obtenerlo de muchos sitios en Internet, free de cargo. Uno de esos sitios es Yahoo! Finance. Se puede acceder a los datos históricos de S&P 500 de la década de 1950 a la actualidad en la URL:

[**http://finance.yahoo.com/q/hp?s=%5EGSPC**](http://finance.yahoo.com/q/hp?s=%5EGSPC)

El archivo **sp500.csv**, que se incluye con la descarga complementaria de este libro, contiene datos históricos de S&P 500 de la década de 1950 al 14 de mayo de 2008. Los datos de este archivo se muestran en el listado 10.1.

### Listado 10.1: Datos históricos de S&P 500 (sp500.csv)

Fecha,Apertura, Alto, Bajo, Cierre, Volumen,Adj Cierre

2008-04-18,1369.00,1395.

90.1369,00,1390,33,4222380000,1390.33 2008-04-17,1363.37,1368.60,1

357,25,1365.56,371388000,1365.56 2008-04-16,1337.02,1365.49,1337.

02.1364,71,4260370000,1364.71 2008-04-15,1331.72,1337.72,1324.35,1

334,43,358123000,1334.43 2008-04-14,1332.20,1335.64,1326.16,1328.

32.356502000.1328,32 2008-04-11.1357,98.1357,98.1331,21.1332,83,3

723790000,1332.83 2008-04-10,1355.37,1367.24,1350.11,1360.55,36861

50000,1360.55 2008-04-09,1365.50,1368.39,1349.97,1354.49,35566700

00.1354,49 2008-04-08.1370,16.1370,16.1360,62,1365,54,36025000000,

1365,54 2008-04-07.1373,69,1386,74,1369.02,1372.54,374778000,137

2,54 2008-04-04,1369,85,1380,91,1362.83,1370.40,3703100000,1370.40

...

1950-01-23,16.92,16.92,16.92,16.92,1340000,16.92 1950-01-20,16.90

16.90,16.90,16.90,1440000,16.90 1950-01-19,16.87,16.87,16.87,16,16.

87.1170000,16,87 1950-01-18,16,85,16,85,16,85,16,85,1570000,16.85

1950-01-17,16.86,16.86,16.86,16.86,1790000,16.86 1950-01-16,16.72

16,72,16,72,16,72,1460000,16.72 1950-01-13,16.67,16.67,16.67,16.16.

67.3330000,16,67 1950-01-12,16,76,16,76,16,76,16,76,297000,16.76

1950-01-11,17.09,17.09,17.09,17.09,2630000,17.09 1950-01-10,17.03

17.03.17.03,17.03,2160000,17.03 1950-01-09,17.08,17.08,17.08,17,17.

08.2520000,17,08 1950-01-06,16,98,16,98,16,98,16,98,2010000,16.98

1950-01-05,16.93,16.93,16.93,16.93,2550000,16.93 1950-01-04,16.85

16,85,16,85,16.85,1890000,16.85 1950-01-03,16.66,16.66,16.66,16,16.

66,1260000,16.66

Un archivo CSV contiene datos de tal manera que cada línea es un registro y comas separan los campos in-dividual dentro de cada línea. Como se mencionó anteriormente, el ejemplo presentado en el capítulo thi s también utiliza datos de tipos de interés preferencial.

## Obtención de datos de tipos de interés preferencial

Hay muchos sitios de Internet que proporcionan datos históricos de tasas de interés preferenciales. La descarga complementaria de este libro contiene un archivo denominado **prime.csv**. Este archivo contiene tasas de interés preferenciales de aproximadamente el mismo período de tiempo que los datos del S&P 500 proporcionados. El contenido de **prime.csv** se muestra en el listado 10.2.

### Listado 10.2: Prime Interest Rate Datos históricos

fecha, primo

1955-08-04,3.25

1955-10-14,3.50

1956-04-13,3.75

1956-08-21,4.00

1957-08-06,4.50

1958-01-22,4.00

1958-04-21,3.50

1958-09-11,4.00

1959-05-18,4.50

...

2005-12-13,7.25

2006-01-31,7.50

2006-03-28,7.75

2006-05-10,8.00

2006-06-29,8.25

2007-09-18,7.75

2007-10-31,7.50

2007-12-11,7.25

2008-01-22,6.50

2008-01-30,6.00

2008-03-18,5.25

Los datos de este archivo se combinarán con los datos S&P 500 para formar los datos reales que se utilizarán para entrenar la red neuronal S&P 500.

# Ejecución del Programa de Predicción S&P 500

Hay dos modos de operación para la aplicación de predicción S&P 500. El modo de operación depende de los argumentos de línea de comandos proporcionados al programa. Si no se especifica ningún argumento de línea de comandos, la red neuronal se carga desde el archivo **sp500.net**. Si se especifica el argumento de comando **FULL,** la red neuronal se tra en una nueva red neuronal y la guardará en el disco bajo el nombre **sp500.net**.

Puede tomar muchas horas entrenar completamente la red neuronal. Por lo tanto, no querrá ejecutarlo en modo completo cada vez. Sin embargo, si decide cambiar algunos de los parámetros de entrenamiento, debe volver a entrenar la red neuronal y generar un nuevo archivo de **sp500.net.** La descarga complementaria contiene un archivo **sp500.net** que ha sido entrenado conuna precisión del 2% de los conjuntos de entrenamiento.

Si ejecuta el programa en modo de entrenamiento completo, la siguiente salida será pro- duced:

### Listado 10.3: Capacitación de la Red Neuronal SP500

Muestras leídas: Iteración 14667 (Backprop) #1

Error:0.6999154401150052 Iteración(Backprop) #2

Error:0.6464464887928701 Iteración(Backprop) #3

Error:0.584286620498403 Iteración(Backprop) #4

Error:0.5161413540009822 Iteration(Backprop) #5

Error:0.44688028770366317 Iteración(Backprop) #6

Error:0.3832980672593392 Iteración(Backprop) #7

Error:0.33189098575632436 Iteración(Backprop) #8

Error:0.2958585679317178 Iteración(Backprop) #9

Error:0.2738894563079073 Iteración(Backprop) #10

Error:0.2619015539956993

...

Iteración (Backprop) #2038 Error:0.02003270683329087

Iteración (Backprop) #2039 Error:0.02002936831637675

Iteración (Backprop) #2040 Error:0.020026031153749693

Iteración (Backprop) #2041 Error:0.020022695344982695

Iteración(@iteración) #2042 Error:0.02001936088961063

Iteración (Backprop) #2043 Error:0.02001602778716852

Iteración (Backprop) #2044 Error:0.0200126960371914

Iteración (Backprop) #2045 Error:0.020009365639214557

Iteración (Backprop) #2046 Error:0.020006036592773283

Iteración (Backprop) #2047 Error:0.02000270889740304

Iteración (Backprop) #2048 Error:0.019999382552639385

Como se puede ver, se necesitó un número considerable de iteraciones de entrenamiento para entrenar la red neuronal al nivel deseado. Inmediatamente después de completar el entrenamiento para esta red, la red neuronal se ejecutó en modo de predicción y se produjo la siguiente salida:

### Listado 10.4: Predicción de la red neuronal SP500

2007-01-03:Inicio=1416. 6,Cambio del % real=-0,12%,Cambio % previsto= 1,78%:D ifference=1,90%

2007-01-04:Inicio=1418,34,Cambio real%=0,12%,Cambio % previsto= 1,39%:D ifference=1,27%

2007-01-05:Inicio=1409.71,Cambio del % real=-0.61%, Cambio % predicho= 1.06%:D ferencia=1.67%

2007-01-08:Inicio=1412.84,Cambio real%=0.22%,Cambio % predicho= 1.28%:D ferencia=1.06%

2007-01-09:Inicio=1412.11,Cambio real=-0.05%,Cambio % predicho= 1.35%:D ifference=1.41%

2007-01-10:Inicio=1414.85,Cambio real%=0.19%,Cambio % predicho= 0.87%:D ferencia=0.67%

2007-01-11:Inicio=1423,82,Cambio real%=0,63%,Cambio % previsto= 0,53%:D ifference=0,11%

2007-01-12:Inicio=1430,73,Cambio del % real=0,49%,Cambio % previsto= 1,42%:D ifference=0,94%

2007-01-16:Inicio=1431,9,Cambio real%=0,08%,Cambio % predicho= 1,71%:D ifference=1,63%

...

2008-03-27:Inicio=1325.76,Cambio real%=-1.15%,Cambio % predicho= 2.85%:D ferencia=3.99%

2008-03-28:Inicio=1315.22,Cambio real%=-0.80%,Cambio % predicho= 3.20%:D ifference=4.00%

2008-03-31:Inicio=1322.7,Cambio real%=0.57%,Cambio % predicho=

-0.78%:D ferencia=1.34%

2008-04-01:Inicio=1370.18,Cambio real%=3.59%,Cambio % predicho=

-0.43%:D ifference=4.02%

2008-04-02:Inicio=1367,53,Cambio del % real=-0,19%,Cambio % previsto= 1,30%:D ifference=1,49%

2008-04-03:Inicio=1369.31,Cambio real%=0.13%,Cambio % predicho=

-0,56%:D ifference=0,69%

2008-04-04:Inicio=1370,4,Cambio real%=0,08%,Cambio % predicho= 1,30%:D iefervescencia=1,22%

2008-04-07:Inicio=1372,54,Cambio del % real=0,16%,Cambio % previsto=

2.25%:D ferencia=2.09%

2008-04-08:Inicio=1365,54,Cambio del % real=-0,51%, Cambio % previsto= 2,09%:D ifference=2,60%

2008-04-09:Inicio=1354,49,Cambio del % real=-0,81%, Cambio % previsto= 2,23% :D ifference=3,04%

2008-04-10:Inicio=1360,55,Cambio real%=0,45%,Cambio % previsto= 1,50%:D ifference=1,05%

2008-04-11:Inicio=1332,83,Cambio real=-2,04%,Cambio % previsto= -0,94% :D ifference=1,10%

2008-04-14:Inicio=1328,32,Cambio real=-0,34%,Cambio % previsto= -1,25% :D ifference=0,91%

2008-04-15:Inicio=1334.43,Cambio real%=0.46%,Cambio % predicho= 0.43%:D ferencia=0.03%

2008-04-16:Inicio=1364.71,Cambio real%=2.27%,Cambio % predicho=

-0,03%:D ifference=2,30%

2008-04-17:Inicio=1365.56,Cambio real%=0.06%,Cambio % predicho=

-0,02%:D ifference=0,08%

2008-04-18:Inicio=1390.33,Cambio real% =1.81%,Cambio % predicho= 1.04%:D ifference=0.77%

Como se puede ver en los datos anteriores, la predicción está lejos de ser perfecta, aunque generalmente se mantiene dentro de un rango de precisión del 10%. Fácilmente podría ser mucho mejor, o mucho peor, a medida que se dispone de datos adicionales.

# Creación de los datos reales de S&P 500

Como se mencionó anteriormente, los datos reales de entrada de la red neuronal se componen tanto de la tasa de interés preferencial como de los datos históricos del S&P 500. Estos datos se almacenan en una clase denominada **SP500Actual**. Esta clase se muestra en el listado 10.5.

### Listado 10.5: Almacenamiento de datos reales de S&P 500 (SP500Actual.java)

paquete com.heatonresearch.book.introneuralnet.ch10.sp500;

importar java.io.IOException; importar java.text.ParseException; importar java.util.Date;

importar java.util. Establecer; importar java.util.TreeSet;

importar com.heatonresearch.book.introneuralnet.common.ReadCSV;

/\*\*

* Capítulo 10: Aplicación a los mercados financieros

\*

* SP500Actual: Contiene datos sp500 reales y tipos de interés preferenciales.

\*

* @author Jeff Heaton
* @version 2,1

\*/

clase pública SP500Actual {

conjunto final privado< tasas de > de tasa deinterés = nuevo TreeSet<> de tasa deinterés();

conjunto final privado< muestras de > FinancialSample = nuevo TreeSet<FinancialSample>();

entrada final privadaSize; entrada final privadaSize ;

SP500real (entrada finalSize, salida final intSize)

{

this.inputSize = inputSize; this.outputSize = outputSize;

}

valor público void calculatePercents() { double prev = -1;

para (ejemplo final de FinancialSample : this.samples) { if (prev != -1) {

doble movimiento final = sample.getAmount() - prev;

doble porcentaje final = movimiento / prev; sample.setPercent(porcentaje);

}

prev = sample.getAmount();

}

}

valor void público getInputData(desplazamientofinal int, entrada final double[] {

objeto final[] samplesArray = this.samples.toArray();

obtener SP500 & datos primos

para (int i = 0; i < this.inputSize; i++) { muestra final de FinancialSample = (FinancialSample) samplesArray[offset

+ i];

entrada[i] = sample.getPercent();

entrada[i + this.outputSize] = sample.getRate();

}

}

valor void público getOutputData(desplazamientofinal int, salida doble final) {

objeto final[] samplesArray = this.samples.toArray(); para (int i = 0; i < this.outputSize; i++) {

ejemplo final de FinancialSample = (FinancialSample) samplesArray[offset

+ this.inputSize + i]; salida[i] = sample.getPercent();

}

}

doble getPrimeRatepúblico(fecha de fecha final) { double currentRate = 0;

para (tasa de interés final : this.rates) {

si (tasa. getEffectiveDate().after(date)) { return currentRate;

} else {

currentRate = rate.getRate();

}

}

devolver currentRate;

}

/\*\*

* + @return las muestras

\*/

conjunto público<FinancialSample> getSamples() { devolver this.samples;

}

carga pública void ( string final sp500Filename, string primeFilename final )

produce IOException, ParseException { loadSP500(sp500Filename); loadPrime(primeFilename); stitchInterestRates(); calculatePercents();

}

public void loadPrime(final String primeFilename) produce IOException,

Excepción De excepción {

final ReadCSV csv = nuevo ReadCSV(primeFilename);

mientras que (csv.next()) {

fecha final = csv.getDate("fecha");

doble velocidad final = csv.getDouble("prime"); final InterestRate ir =

nueva tasa de interés(fecha, tasa); this.rates.add(ir);

}

csv.close();

}

carga pública voidSP500(final String sp500Filename) produce IOException,

Excepción De excepción {

final ReadCSV csv = nuevo ReadCSV(sp500Filename); mientras que (csv.next()) {

fecha final = csv.getDate("fecha");

cantidad doble final = csv.getDouble("adj close"); muestra final de FinancialSample =

nuevo FinancialSample(); sample.setAmount(importe); sample.setDate(fecha); this.samples.add(muestra);

}

csv.close();

}

tamaño público int( ) {

devolver this.samples.size();

}

punto de sutura del vacío públicoInterestRates() {

para (ejemplo final de FinancialSample : this.samples) { tasa doble final = getPrimeRate(sample.getDate()); sample.setRate(velocidad);

}

}

}

Hay varias funciones que proporciona esta clase. Estas funciones serán explorada en las siguientes secciones.

**Muestras** financieras

El propósito principal para la clase **SP500Actual** es proporcionar una cotización SP500 y la tasa de interés preferencial para cualquier día dado que el mercado de valores estadounidense estuviera abierto. Además, se almacena el cambio de porcentaje entre la cotización actual y la cotización anterior. Juntos, estos valores se denominan ejemplo. Los ejemplos se almacenan en la clase **FinancialSample.** La clase **FinancialSample** se muestra en el listado 10.6.

### Listado 10.6: Muestras financieras (FinancialSample.java)

paquete com.heatonresearch.book.introneuralnet.ch10.sp500;

importar java. texto. NumberFormat; importar java.util.Date;

importar com.heatonresearch.book.introneuralnet.common.ReadCSV;

/\*\*

* Capítulo 10: Aplicación a los mercados financieros

\*

* FinancialSample: Contiene una muestra de datos financieros en el
* fecha especificada. Esto incluye el cierre del SP500 y
* la tasa de interés preferencial.

\*

* @author Jeff Heaton
* @version 2,1

\*/

implementa FinancialSample de clase pública

Comparable<FinancialSample> { cantidad doble privada; doble tarifa privada; fechade fecha privada; doble por ciento privado;

public int compareTo(final FinancialSample other) { return getDate(). compareTo(other.getDate());

}

/\*\*

* + @return la cantidad

\*/

público double getAmount() { devolver this.amount;

}

/\*\*

* + @return la fecha

\*/

fecha pública getDate() { devolver this.date;

}

/\*\*

* + @return el porcentaje

\*/

público double getPercent() { devolver this.percent;

}

/\*\*

* + @return la tasa

\*/

double getRatepúblico() { return this.rate;

}

/\*\*

* + @param cantidad
  + la cantidad a establecer

\*/

valor público void setAmount( importe doble final) { this.amount = amount;

}

/\*\*

* + fecha @param
  + la fecha a fijar

\*/

público void setDate(fecha de fecha final) { this.date = fecha;

}

/\*\*

* + @param por ciento
  + el porcentaje a establecer

\*/

conjunto de vacío públicoPercent( doble porcentaje final) { this.percent = por ciento;

}

/\*\*

* + tasa de @param
  + la tasa a establecer

\*/

valor anulado público( tasa doble final) { this.rate = rate;

}

@Override

string toStringpúblico() { final NumberFormat nf =

NumberFormat.getPercentInstance(); nf.setMinimumFractionDigits(2); nf.setMaximumFractionDigits(2);

resultado final de StringBuilder = nuevo StringBuilder(); result.append(ReadCSV.displayDate(this.date)); result.append(", Cantidad: "); result.append(this.amount);

resultado.append(", Tasa preferencial: "); result.append(this.rate);

resultado.append(", Porcentaje de Anterior: "); result.append(nf.format(this.percent)); return result.toString();

}

}

Otra característica importante de la clase **FinancialSample** es que es implementa interfaz **comparable.** Esto permite que los objetos **FinancialSample** se agreguen a una colección Java ordenada y se ordenen por sus fechas.

## Obtener la tarifa prime por un día

El archivo de datos de tipos de interés preferencial, denominado **prime.csv**, sólo contiene la tasa de interés preferencial durante días en los que la tasa de interés cambió. Por lo tanto, se requiere un método especial para determinar cuál era la tasa de interés en una fecha específica. Este método se denomina **getPrimeRate**. La firma para **getPrimeRate** se muestra aquí:

doble getPrimeRatepúblico(fechade fecha final)

En primer lugar, se define una variable para contener la última tasa encontrada, que es la tasa actual.

Esta variable se denomina **currentRate**.

doble corrienteRate = 0;

A continuación, el método recorre todos los tipos de interés. Estas tasas de interés se almacenan en una lista ordenada, por lo que primero se producen las tasas de interés de las fechas más tempranas.

para (tasa de interés final : this.rates) {

Tan pronto como se encuentra la primera tasa interest con una fecha más allá de la fecha de interés, entonces la tasa almacenada en **currentRate** es la tasa de interés para la fecha especificada. Si aún no se ha establecido la variable **currentRate,** la fecha especificada es anterior a las fechas de nuestros datos. Si este es el caso, entonces no tenemos datos de tasa de interés para la fecha especificada y se devuelve un valor de **null.**

if (rate.getEffectiveDate().after(date)) { return currentRate;

} else {

De lo contrario, aún no se ha alcanzado la fecha especificada, por lo que currentRate

variable se actualiza.

currentRate = rate.getRate();

}

}

Si llegamos al final de la lista, entonces la tasa de interés final simplemente se devuelve.

asumir que la tasa no ha cambiado desde nuestro último valor de datos y fecha especificada. Mientras nuestro archivo de tasas de interés esté actualizado, y la fecha especificada no sea en el futuro, esta es una suposición válida.

devolver currentRate;

}

Dado que el método **getPrimeRate** debe iterar para encontrar la tasa de interés, llamarla es algo cara; por lo tanto, cada muestra de S&P 500 debe ser "cosida" a la tasa de interés correcta.

## Coser las tarifas a rangos

Se llama a la función **stitchInterestRates** para encontrar la tasa de entrada y entrada adecuada para cada uno de los objetos **FinancialSample.** La firma para el método **stitchInterestRates** se muestra aquí:

punto vacío públicoInterestRates()

Comenzamos recorriendo todos los objetos **de FinancialSample.**

para (ejemplo final de FinancialSample : this.samples) {

Para cada objeto **FinancialSample,** obtenemos la tasa de interés preferencial.

doble velocidad final = getPrimeRate(sample.getDate()); sample.setRate(velocidad);

}

Este proceso se continúa hasta que todos los objetos **de FinancialSample** hayan sido procesado.

Para entrenar la red neuronal, se deben crear datos de entrada e ideales. Las dos secciones siguientes describen cómo se hace esto.

## Creación de los datos de entrada

Para crear datos de entrada para la red neuronal, se utiliza el método **getInputDate** de la clase **SP500Actual.** La firma para el método **getInputData** se muestra aquí:

valor void público getInputData(desplazamientofinal int, entrada final double[])

Se pasan dos argumentos al método **getInputData.** El **argumento de desfase** - ment especifica el índice basado en cero en el que se extraerán los datos de entrada. El argumento **de entrada** proporciona una **matriz doble** en la que se copiarán los ejemplos financieros. Esta matriz también especifica el número de objetos **FinancialSample** a pro- cess. Se procesarán suficientes objetos **FinancialSample** para rellenar la matriz.

En primer lugar, se obtiene una matriz de referencias a las muestras.

objeto final[] samplesArray = this.samples.toArray();

A continuación, avanzamos en bucle, según el tamaño de la matriz de **entrada.**

para (int i = 0; i < this.inputSize; i++) {

A continuación, se obtiene cada objeto **FinancialSample.**

ejemplo final de FinancialSample = (FinancialSample)

samplesArray[offset

+ i];

Se copian tanto el porcentaje de cambio como la velocidad de cada muestra. A continuación, la red neuronal utiliza estos dos valores para realizar una predicción.

entrada[i] = sample.getPercent();

entrada[i + this.outputSize] = sample.getRate();

}

Como puede ver, la entrada a la red neuronal consiste en cambios porcentuales y

el nivel actual de la tasa de interés preferencial. El uso de los cambios porcentuales es diferente de cómo se manejó la entrada para la red neuronal presentada en el capítulo 9. En el capítulo 9, las numbers reales seagregaron a la red neuronal. El programa de este capítulo, en su lugar, realizará un seguimiento de los movimientos porcentuales. En general, el S&P 500 ha aumentado a lo largo de su historia, y no ha revisado a menudo los rangos. Por lo tanto, se pueden encontrar más patrones mediante el seguimiento de los cambios de percent, en lugar de los valores de punto reales.

## Creación de los datos de salida ideales

Para el entrenamiento supervisado, las salidas ideales también deben calcularse a partir de datos conocidos. Mientras que los insumos incluyen datos de tasa de interés y cotización, las salidas solo contaminan los datos de porcentaje de cotización. Estamos tratando de predecir el movimiento porcentual en el S&P 500; no estamos tratando de predecir fluctuaciones en la tasa de interés preferencial.

Los datos de salida ideales se crean llamando al método **getOutputData.** La firma para el método **getOutputData** se muestra aquí:

valor void público getOutputData(desplazamientofinal int, salida final double[])

Se pasan dos argumentos a **getInputData**. El argumento **offset** especifica el índice basado en cero en el que se extraerán los datos de salida. El argumento **output** proporciona una matriz **doble** en la que se copiarán los ejemplos financieros. Esta matriz también especifica el número de objetos **FinancialSample** que se van a processed. Se procesarán suficientes objetos **FinancialSample** para rellenar la matriz.

En primer lugar, se obtiene una matriz de referencias a las muestras.

objeto final[] samplesArray = this.samples.toArray();

A continuación, recorremos las muestras.

para (int i = 0; i < this.outputSize; i++) {

Para cada ejemplo, copiamos solo el cambio porcentual en la matriz de salida.

ejemplo final de FinancialSample = (FinancialSample)

samplesArray[offset

+ this.inputSize + i]; salida[i] = sample.getPercent();

}

La tasa de interés preferencial no se copia, porque la red neuronal no está tratando de

predecir fluctuaciones en la tasa de interés preferencial. La red neuronal sólo predice fluctuaciones en el índice S&P 500.

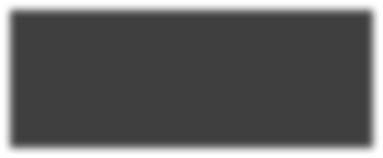
# Formación de la Red S&P 500

Se utiliza una combinación de retropropagación y recocido simulado para entrenar esta red neuronal. Una red compleja como esta a menudo se beneficia de un enfoque de entrenamiento híbrido.

## Minima local

El error de un algoritmo de entrenamiento de backpropagation tiene una tendencia a encontrar el problema del mínimo local. Idealmente, debe entrenar hasta que se alcance el mínimo mundial. Sin embargo, el algoritmo de backpropagation a veces puede confundir un mínimo local con el mínimo global. Considere la Figura 10.3.

### Figura 10.3: Minima global y local



Como se puede ver en la Figura 10.3, hay cuatro minima locales. Sólo uno de estos minimas locales es el mínimo mundial. Para evitar el problema del mínimo local, se utiliza un algoritmo de entrenamiento híbrido.

## Entrenamiento híbrido

La clase principal para el ejemplo de predicción S&P 500 es **PredictSP500**. Esta clase debe ejecutarse para ejecutar este ejemplo. La clase **PredictSP500** se muestra en el listado 10.7:

### Listado 10.7: Intente predecir el S&P 500 (PredictSP500.java)

paquete com.heatonresearch.book.introneuralnet.ch10.sp500;

importar java.io.IOException; importar java.text.NumberFormat; importar java.util.Date;

importar com.heatonresearch.book.introneuralnet.common.ReadCSV; importar com.heatonresearch.book.introneuralnet.neural.activation. ActivationFunction;

importación com.heatonresearch. book.introneuralnet.neural.activation.

ActivationTANH;

importar com.heatonresearch.book.introneuralnet.neural.feedforward.

FeedforwardLayer;

importar com.heatonresearch.book.introneuralnet.neural.feedforward.

FeedforwardNetwork;

importar com.heatonresearch.book.introneuralnet.neural.feedforward.

tren. Tren;

importar com.heatonresearch.book.introneuralnet.neural.feedforward.

train.anneal. NeuralSimulatedAnnealing;

importar com.heatonresearch.book.introneuralnet.neural.feedforward.

train.backpropagation. Backpropagation;

importar com.heatonresearch.book.introneuralnet.neural.util.

ErrorCalculación;

importar com.heatonresearch.book.introneuralnet.neural.util.

SerializeObject;

/\*\*

* Capítulo 10: Aplicación a los mercados financieros

\*

* PredictSP500: Intente predecir el SP500 usando un predictivo
* red neuronal de avance. La palabra clave es "intento". El
* red neuronal puede adivinar algunas tendencias básicas en el SP500, y es
* significaba sólo como un punto de partida.

\*

* ¡Esta clase no es de ninguna manera asesoramiento de inversión!

\*

* @author Jeff Heaton
* @version 2,1

\*/

clase pública PredictSP500 {

int estático final público TRAINING\_SIZE = 500; int estático final público INPUT\_SIZE = 10; int estático final público OUTPUT\_SIZE = 1; int estática final pública NEURONS\_HIDDEN\_1 = 20; int estático final público NEURONS\_HIDDEN\_2 = 0; doble MAX\_ERROR estático final público = 0,02; PREDICT\_FROM de fecha estática final pública =

ReadCSV.parseDate("2007-01-01"); LEARN\_FROM de fecha estática final pública =

ReadCSV.parseDate("1980-01-01");

vacío estático público main(final String args[]) {

predicto finalSP500 = nuevo PredictSP500(); if (args.length > 0 &&

args[0]. equalsIgnoreCase("completo"))

más

}

predict.run(true); predict.run(false);

entrada doble privada[][];

doble ideal privado[][];

red privada feedforwardNetwork; SP500Actual privado real;

public void createNetwork() {

umbral final de activaciónFunction = nueva ActivationTANH();

this.network = nuevo FeedforwardNetwork(); this.network.addLayer(nuevo FeedforwardLayer(umbral,

PredictSP500.INPUT\_SIZE \* 2));

this.network.addLayer(nuevo FeedforwardLayer(umbral, PredictSP500.NEURONS\_HIDDEN\_1));

si (PredictSP500.NEURONS\_HIDDEN\_2 > 0) {

this.network.addLayer(

nuevo FeedforwardLayer(umbral, PredictSP500.NEURONS\_HIDDEN\_2));

}

this.network.addLayer(nuevo FeedforwardLayer(umbral, PredictSP500.OUTPUT\_SIZE));

this.network.reset();

}

exhibición pública del vacío( ) {

numberformat final percentFormat = NumberFormat.getPercentInstance();

percentFormat.setMinimumFractionDigits(2);

doble final[] presente = nuevo doble[INPUT\_SIZE \* 2]; doble[] predecir = nuevo doble[OUTPUT\_SIZE];

doble final[] realOutput = nuevo doble[OUTPUT\_SIZE];

int index = 0;

para (ejemplo final de FinancialSample: this.real.getSamples()) { if (sample.getDate().after(

PredictSP500.PREDICT\_FROM)) {

final StringBuilder str = nuevo StringBuilder();

str.append(ReadCSV.displayDate( sample.getDate()));

str.append(":Start="); str.append(sample.getAmount());

this.real.getInputData(índice - INPUT\_SIZE, presente);

this.real.getOutputData(índice - INPUT\_SIZE, actualOutput);

predict = this.network.computeOutputs(presente); str.append(,"Real % Change="); str.append( percentFormat.format(actualOutput[0])); str.append(,Predicted % Change= "); str.append( percentFormat.format(predict[0]));

str.append(":D ifference=");

error finalCalculat error iónico = nuevo ErrorCalculation();

error.updateError(predict, realOutput); str.append(percentFormat.format( error.calculateRMS()));

//

System.out.println(str.toString());

}

índice++;

}

}

private void generateTrainingSets() {

this.input = nuevo double[TRAINING\_SIZE][INPUT\_SIZE \* 2]; this.ideal = nuevo doble[TRAINING\_SIZE][OUTPUT\_SIZE];

encontrar donde estamos empezando desde int startIndex = 0;

para (ejemplo final de FinancialSample : this.real. getSamples()) {

if (sample.getDate().after(LEARN\_FROM)) { break;

}

startIndex++;

}

crear un factor de muestra en todo el área de entrenamiento final int eligibleSamples = TRAINING\_SIZE - startIndex; if (eligibleSamples == 0) {

System.out

. println(

"Necesita una fecha anterior para LEARN\_FROM o un "

+"número más pequeño para TRAINING\_SIZE."); System.exit(0);

}

factor int final = aptoSamples / TRAINING\_SIZE;

agarrar los datos de entrenamiento reales desde ese punto para (int i = 0; i < TRAINING\_SIZE; i++) {

this.real.getInputData(startIndex + (i \* factor), this.input[i]); this.real.getOutputData(startIndex + (i \* factor), this.ideal[i]);

}

}

public void loadNeuralNetwork() produce IOException, ClassNotFoundException {

this.network = (FeedforwardNetwork) SerializeObject.load("sp500.net");

}

public void run(boolean full) { try {

this.actual = nuevo SP500Actual( INPUT\_SIZE, OUTPUT\_SIZE);

this.real.load("sp500.csv", "prime.csv");

System.out.println("Ejemplos leídos: "

+ this.real.size());

si (lleno) {

createNetwork(); generateTrainingSets();

trainNetworkBackprop(); saveNeuralNetwork();

} else {

loadNeuralNetwork();

}

display();

} catch (excepción final e) { e.printStackTrace();

}

}

public void saveNeuralNetwork() produce IOException { SerializeObject.save("sp500.net", this.network);

}

tren vacío privadoNetworkBackprop() {

tren final = nueva backpropagation(

this.network, this.input,

this.ideal, 0.00001, 0.1); double lastError = Double.MAX\_VALUE; int epoch = 1;

int lastAnneal = 0;

hacer {

train.iteration();

doble error = train.getError();

System.out.println("Iteración (Backprop) #"

+ epoch + " Error:"

+ error);

if( error>0.05 )

{

if( (últimoanneal>100)

&& (error>último || Math.abs(error-lastError)<0.0001) )

{

trainNetworkAnneal(); lastAnneal = 0;

}

}

lastError = train.getError(); epoch++;

lastAnneal++;

} mientras que (train.getError() > MAX\_ERROR);

}

tren vacío privadoNetworkAnneal() { System.out.println(

"Formación con recocido simulado para 5 iteraciones");

entrenar la red neuronal

último tren NeuralSimulatedAnnealing = nuevo NeuralSimulatedAnnealing(

this.network, this.input, this.ideal,

10, 2, 100);

int epoch = 1;

for(int i=1;i<=5;i++) { train.iteration();

System.out.println("Iteración (Anneal) #"

+ epoch + " Error:"

+ train.getError()); epoch++;

}

}

}

La clase **PredictSP500** implementa el entrenamiento híbrido, así como los intentos de predecir futuros valores S&P 500 con la red neuronal recién entrenada. El entrenamiento híbrido funciona alternando entre el entrenamiento de backpropagation y el an- nealing simulado. El entrenamiento comienza con la retropropagación y los interruptores a recocido simulado cuando la retropropagación ya no está entrenando eficientemente la red. Una vez que simu- recocido latado se ha utilizado para una serie de ciclos, el programa cambia de nuevo a backpropagation.

La lista 10.8 muestra la salida del entrenamiento híbrido.

### Listado 10.8: Salida de entrenamiento híbrido

Iteración (Backprop) #1511 Error:0.1023912889542664

Iteración(Backprop) #1512 Error:0.10237590385794164

Iteración (Backprop) #1513 Error:0.10236112842990429

Iteración (Backprop) #1514 Error:0.10234696743296834

Iteración (Backprop) #1515 Error:0.1023334256570161

Iteración (Backprop) #1516 Error:0.10232050792236635

Iteración (Backprop) #1517 Error:0.10230821907285384

...

Iteración (Backprop) #1518 Error:0.10229656398261411

Iteración (Backprop) #1519 Error:0.10228554755257999

Iteración (Backprop) #1520 Error:0.10227517471108645 Iteración(Backprop) #1521 Error:0.10226545041378378 Train-

ing con recocido simulado para 5 iteraciones Iteración (Anneal)

#1 Error:0.042124954651261835 Iteration(Anneal) #2 Er- ror:0.042124954651261835 Iteration(Anneal) #3 Er- ror:0.042124954651261835 Iteration(Anneal) #4 Error:0.042124954651261835 Iteration(Anneal) #5 Er- ror:0.042124954651261835 Iteration(Backprop) #1522 Er- ror:0.04137291563937421 Iteration(Backprop) #1523 Error:0.04079595880076687 Iteration(Backprop) #1524 Er- ror:0.04031691771522145 Iteration(Backprop) #1525 Er- ror:0.03987729279067175 Iteration(Backprop) #1526 Error:0.03945727030649545 Iteration(Backprop) #1527 Er- ror:0.03905037926667383 Iteration(Backprop) #1528 Er- ror:0.038654238218587864 Iteration(Backprop) #1529 Error:0.038267815849145556 Iteration(Backprop) #1530 Er- ror:0.037890572580577805 Iteration(Backprop) #1531 Er- ror:0.03752216093273823 Iteration(Backprop) #1532 Error:0.03716231245175569 Iteration(Backprop) #1533 Er- ror:0.036810793350444744

...

Como se puede ver en el código anterior, backpropagation se utiliza a través de iteración 1.521. La mejora entre las iteraciones 1.520 y 1.521 no fue suficiente, por lo que el recocido sim-ulated se empleó para cinco iteraciones. Antes de que se utilizara el recocido simulado, la tasa de error rondaba el 10%. Después del recocido simulado, la tasa de error seextendió rápidamente a alrededor del 4%. El recocido simulado logró evitar el mínimo lo-cal al que se acercaba la sesión de entrenamiento anterior.

El algoritmo de entrenamiento híbrido se implementa en el método **trainNeuralNetworkHybrid.** La firma para el método **trainNeuralNetworkHybrid** se muestra aquí:

tren vacío privadoNetworkHybrid()

El entrenamiento híbrido comienza como una sesión regular de entrenamiento de backpropagation. Se implementa un entrenador de backpropagation con una baja tasa de entrenamiento y un momenbajo- tum.

tren final = nueva backpropagation (this.network, this.input, this.ideal, 0.00001, 0.1);

Realizamos un seguimiento del último error, para que podamos medir el rendimiento del algoritmo de entrenamiento. Inicialmente, este último valor de error se establece muy alto para que se ini correctamente - tialized durante la primera iteración.

double lastError = Double.MAX\_VALUE; int epoch = 1;

Sólo usamos recocido simulado cada 100 iteraciones; oa medida que las mejoras se vuelven muy pequeñas hacia el final del entrenamiento, el recocido simulado sería invocado constantemente. Utilizamos la última variable **deAnneal** para realizar un seguimiento de cuántas épocas ha sido desde el último intento de recocido simulado.

int lastAnneal = 0;

hacer { tren.iteration();

doble error = train.getError();

Para cada época, actualizamos el progreso.

System.out.println("Iteración (Backprop) #" + epoch + " Error:"

+ error);

Si el error es superior al 5%, entonces consideraremos el uso derecocido simulado de ed. Una vez que el error es inferior al 5%, por lo general es mejor dejar que la backpropagation termine el entrenamiento.

if( error>0.05 )

{

Ahora debemos considerar si nos gustaría usar recocido simulado. Si han sido 100 iteraciones desde la última vez que usamos recocido simulado y la tasa deerror no ha demostrado en una centésima parte de un por ciento, entonces intentaremos entrenar la red neuronal usando recocidoesmulado esmulado.

if((últimoanneal>100) && (error>lastError || Math.abs(error-lastError)<0.0001) )

{

Para entrenar usando recocido simulado, llamamos al **trenNetworkAnneal** meth-od. Este método funciona muy parecido a nuestros ejemplos anteriores de recocido simulado, por lo que no se repetirá aquí. Una vez completado el entrenamiento simulado de recocido, la última variable **deAnneal** se establece en cero para que podamos volver a realizar un seguimiento de cuántas épocas ha sido desde que se utilizó por última vez el entrenamiento simulado de recocido.

trainNetworkAnneal(); lastAnneal = 0;

}

}

Realizamos un seguimiento del último error y del número de época.

lastError = train.getError(); epoch++;

lastAnneal++;

} mientras que (train.getError() > MAX\_ERROR);

La constante **MAX\_ERROR** para este ejemplo se establece en 2%. Es posible entrenar este ejemplo a menos del 1%, pero se necesitan casi un millón de épocas y varios días de entrenamiento.

# Tratando de predecir el S&P 500

En este ejemplo se utiliza el método **predict** para intentar predecir el S&P 500. La firma para el método **predict** se muestra aquí:

predicción del vacío público()

En primer lugar, creamos un objeto **NumberFormat** diseñado para dar formato a porcentajes.

final NumberFormat percentFormat = NumberFormat.getPercentIn- stance();

percentFormat.setMinimumFractionDigits(2);

Se crean tres matrices. La **matriz actual** contiene los "valores actuales" en los que se basará la predicción. La matriz **predicta** mantendrá los valores S&P 500 predichos. La matriz **actualOutput** contendrá los valores reales de los datos s&p 500 tórricos. La matriz **realOutput** se comparará con la matriz **predicha** para determinar la eficacia de la red neuronal.

doble final[] presente = nuevo doble[INPUT\_SIZE \* 2]; doble[] predecir = nuevo doble[OUTPUT\_SIZE];

doble final[] realOutput = nuevo doble[OUTPUT\_SIZE];

Recorremos todos los objetos **de FinancialSample** que caen en el rango para el que se va a hacer esta predicción.

int index = 0;

para (ejemplo final de FinancialSample: this.real.getSamples()) {

Si este objeto **FinancialSample** cae en el rango después de la **PREDICT\_FROM**

constante, entonces debemos intentar predecir en base a ella.

if (sample.getDate().after(PredictSP500.PREDICT\_FROM)) {

Creamos un **StringBuilder** que creará la línea de texto que se mostrará y anexa la fecha en la que se tomó el ejemplo.

final StringBuilder str = nuevo StringBuilder(); str.append(ReadCSV.displayDate(sample.getDate()));

A continuación, se muestra el valor inicial del S&P 500 para este sector de tiempo.

str.append(":Start="); str.append(sample.getAmount());

Se obtienen los valores de entrada para la red neural, así como los valores de salida ideales.

this.real.getInputData(índice - INPUT\_SIZE, presente); this.real.getOutputData(índice - INPUT\_SIZE, actualOutput);

A continuación, las salidas se calculan utilizando los valores actuales. Esta es la predicción del trabajo de la red neuronal.

predict = this.network.computeOutputs(presente);

Se muestra el cambio real en el porcentaje para el S&P 500.

str.append(,"Real % Change="); str.append(percentFormat.format(actualOutput[0]));

A continuación, se muestra el cambio previsto en el porcentaje.

str.append(,Predicted % Change= "); str.append(percentFormat.format(predict[0]));

A continuación, se muestra la diferencia entre los valores reales y predichos.

str.append(":D ifference=");

A continuación, el error entre el real y la diferencia se calcula utilizando el cuadrado medio raíz.

error finalCalculation = nuevo ErrorCalculation(); error.updateError(predict, realOutput); str.append(percentFormat.formen(error.calculateRMS()));

Por último, se muestra **stringbuffer.**

System.out.println(str.toString());

}

índice++;

}

Este mismo procedimiento se sigue para cada objeto **FinancialSample** proporcionado.

Este ejemplo sirve como una introducción básica a la predicción financiera con redes neuronales. Un libro entero podría escribirse fácilmente sobre cómo utilizar las redes neuronales con los mercados financieros. Hay muchas opciones disponibles que le permitirán cre- comió redes neuronales financieras más avanzadas. Por ejemplo, se pueden proporcionar insumos adicionales; se pueden utilizar acciones individuales y sus relaciones con otras poblaciones; y se pueden utilizar enfoques híbridos utilizando redes neuronales y otras formas de análisis estadísticos. Este ejemplo sirve como punto de partida.

# Resumen del capítulo

Predecir el movimiento de los mercados financieros es un área de interés muy común para las redes neuronales predictivas. La aplicación de redes neuronales a la previsión financiera podría llenar fácilmente un libro. Este libro proporciona una breve introducción presentando los conceptos básicos de cómo construir una red neuronal que intenta predecir el movimiento de precios en el índice S&P 500.

Para intentar predecir el índice S&P 500, se utilizan tanto la tasa de interés preferencial como los valores anteriores del S&P 500. Esto intenta encontrar tendencias en los datos S&P 500 que podrían utilizarse para predecir el movimiento futuro de los precios.

Este capítulo alasí introdujo el entrenamiento híbrido. El algoritmo de entrenamiento híbrido utilizado en este capítulo hizo uso tanto de la backpropgation como del recocido simulado. La backpropagation se utiliza hasta que la backpropagation ya no produce una reducción satisfactoria en la tasa de error. En este momento, el recocido simulado se utiliza para ayudar a liberar la red neuronal de lo que podría ser un mínimo local. Un mínimo local es un punto bajo en la tabla de entrenamiento, pero no necesariamente el punto más bajo. La contrapropagación tiene una dency de diez para quedar atascado como mínimo local.

Aunque la red neuronal feedforward es una de las formas más comunes de redes neuronales, hay otras arquitecturas de red neuronal que también vale la pena considerar- ing. En el siguiente capítulo,se le introducirá en un mapa autoorganizado. Un mapa auto-orga- nizing se utiliza a menudo para clasificar la entrada en grupos.

# vocabulario

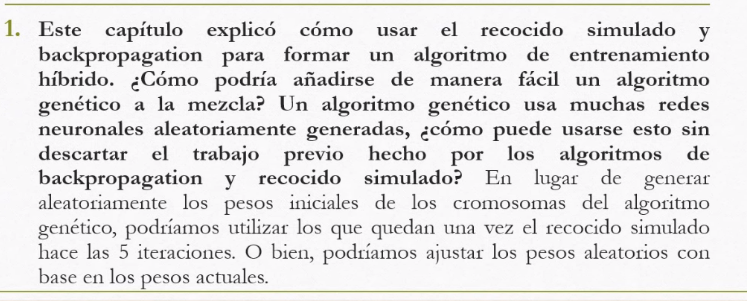
Mínimo global del archivo de valores separados por comas (CSV)

Entrenamiento híbrido Minima Local Tasa de Interés Preferencia S&P 500

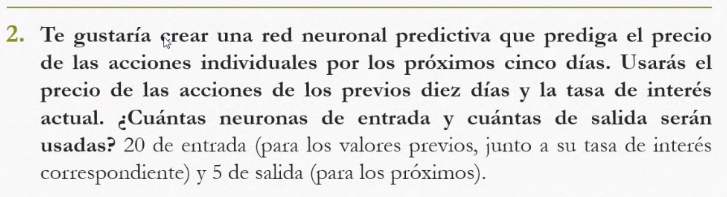
muestra

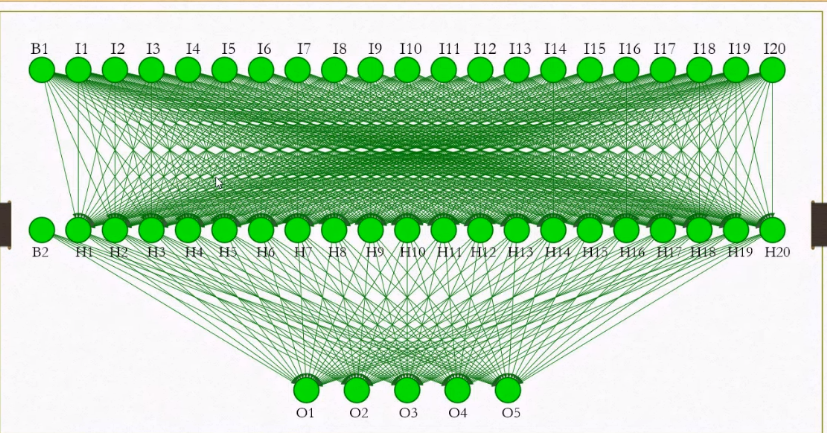
# Preguntas para revisión

1. En este capítulo se explica cómo utilizar el recocido simulado y la retropropagación para formar un algoritmo de entrenamiento híbrido. ¿Cómo se puede añadir fácilmente un algoritmo genético a la mezcla? Un algoritmo genético utiliza muchas redes neuronales generadas aleatoriamente. ¿Cómo se puede utilizar esto sin descartar el trabajo anterior realizado por los algoritmos de retroapropiación y recocido simulado?

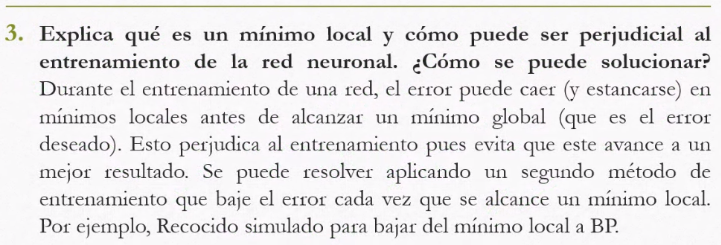


1. Le gustaría crear una red neuronal predictiva que prediga el precio de un stock individual para los próximos cinco días. Utilizará los precios de las acciones a partir de los diez días previos al vious y la tasa de interés preferencial actual. ¿Cuántas neuronas de entrada y cuántas neuronas de salida se utilizarán?

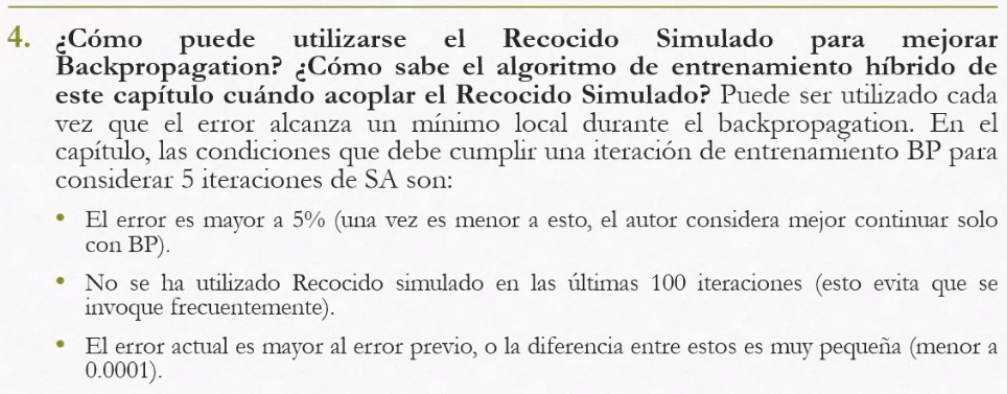




1. Explicar lo que es un mínimo local y cómo puede ser perjudicial para el entrenamiento de trabajo neto neuronal. ¿Cómo se puede superar esto?



1. ¿Cómo se puede utilizar el recocido simulado para aumentar la retropropagación? ¿Cómo sabe el algoritmo de entrenamiento híbrido presentado en este capítulo cuándo participar en el recocidode lata?



1. ¿Por qué es preferible introducir cambios porcentuales en los cambios de la red neuronal financiera en lugar de los precios reales de las acciones? ¿Qué función de activación funcionaría bien para los cambios porcentuales? ¿Por qué?

