

# Redes de Neuronas Artificiales

## Un Enfoque Práctico

**Redes de Neuronas Artificiales**



Isasi  
Galván

Las Redes de Neuronas Artificiales tienen una historia de alrededor de sesenta años, pero no fue sino a partir de los años ochenta cuando la disciplina alcanzó la madurez para poder ser incluida dentro de las materias de estudio dentro de la computación. Es, por tanto, una disciplina en plena pubertad, con todo lo que eso significa. Un período difícil, de mucho cambio, donde aparecen y desaparecen tendencias y corrientes, donde hay pasos hacia delante y hacia atrás, donde en algunos aspectos no se ha alcanzado aún la madurez.

En este libro se ha hecho un esfuerzo por tratar de explicar en detalle sólo aquellos aspectos de la disciplina que ya han alcanzado suficiente madurez.

El espíritu con el que se ha escrito este libro trata de contar, de forma sencilla, pero rigurosa y profunda, la disciplina de las Redes de Neuronas Artificiales. Es por esto que todos los capítulos teóricos del libro concluyen con ejemplos prácticos sencillos, que ayudan en la comprensión de los conceptos introducidos a lo largo del capítulo. Además, cuenta con tres capítulos de aplicaciones prácticas de las Redes de Neuronas a problemas reales. Estos capítulos incluyen problemas difíciles, donde se realiza una descripción de los mismos, la estructura de los datos que los definen, en qué consiste la complejidad de cada uno de ellos y, por supuesto, la solución a los mismos mediante Redes de Neuronas Artificiales. Estas soluciones incluyen la descripción de los modelos, los parámetros, cómo dichos parámetros influyen en la solución y las distintas soluciones encontradas.

El enfoque del libro es el del paradigma matemático para la resolución de problemas, más que las relaciones entre las Redes Neuronales y sus analogías computacionales.

ISBN 84-205-4025-0



PEARSON

# **REDES DE NEURONAS ARTIFICIALES**

## **Un enfoque práctico**

**Pedro Isasi Viñuela  
Inés M. Galván León**

Departamento de Informática  
*Universidad Carlos III de Madrid*



Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima • Montevideo  
San Juan • San José • Santiago • São Paulo • White Plains

Datos de catalogación bibliográfica

ISASI VIÑUELA, P.; GALVÁN LEÓN, I. M.  
REDES DE NEURONAS ARTIFICIALES.  
UN ENFOQUE PRÁCTICO

PEARSON EDUCACIÓN, S.A., Madrid, 2004

ISBN: 84-205-4025-0

Materia: Informática 681.3

Formato 170 × 240

Páginas: 248

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (*arts. 270 y sgts. Código Penal*).

**DERECHOS RESERVADOS**

© 2004 por PEARSON EDUCACIÓN, S.A.  
Ribera del Loira, 28  
28042 MADRID (España)

**REDES DE NEURONAS ARTIFICIALES. UN ENFOQUE PRÁCTICO**

**ISASI VIÑUELA, P.; GALVÁN LEÓN, I. M.**

**ISBN: 84-205-4025-0**

Depósito legal: M. 52.362-2003

PEARSON PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S.A.

**Equipo editorial:**

Editor: David Fayerman Aragón

Técnico editorial: Ana Isabel García Borro

**Equipo de producción:**

Director: José Antonio Clares

Técnico: José Antonio Hernán

**Diseño de cubierta:** Equipo de diseño de Pearson Educación, S.A.

**Impreso por:** Lavel, S.A.

**IMPRESO EN ESPAÑA - PRINTED IN SPAIN**

Este libro ha sido impreso con papel y tintas ecológicos

# Prólogo

Todos los libros están escritos siempre con un ánimo. Este libro no es una excepción. Es en parte el resultado de varios años de trabajo en la enseñanza de estas materias, y tiene la finalidad de enseñar. Afortunadamente, a pesar de la era tecnológica que nos ha tocado presenciar, los libros son aún el pilar fundamental sobre el que descansa la educación. Los libros son como los profesores: los malos pueden hacer que odiemos la materia que nos enseñan; los buenos, por el contrario, pueden conseguir que nos apasionemos por algo, que comprendamos el código oculto que hay detrás de las disciplinas, ya sean científicas, sociales o artísticas, que veamos la belleza que recubre los conocimientos de un determinado campo.

Éste es el objetivo fundamental de este libro. No se trata de hacer un libro de referencia para la comunidad científica, ni siquiera de hacer una obra que contenga nuevos elementos científicos; se trata de intentar acercar las Redes de Neuronas Artificiales a aquellos lectores que estén interesados en saber de qué tratan. Es un libro que nace de la ilusión de los autores por transmitir, de una manera sencilla, todo un campo de conocimiento que resulta, a primera vista, extraño, extravagante y reservado sólo a un grupo de iniciados; pero que, sin embargo, está siendo aplicado para solucionar problemas y forma parte de elementos que, cada vez más, se están incorporando a nuestra vida cotidiana.

Algunas veces los autores se empeñan, en una incomprensible creencia de que lo bueno debe ser ininteligible, en recubrir sus escritos de una estructura barroca, artificialmente sofisticada. También es cierto que otras veces aligerar una materia, sin dejarla desprovista de sus elementos fundamentales y necesarios, es complicado, porque hacer fácil lo difícil no es una tarea sencilla, pero es el objetivo deseable de todo docente. Éste es el espíritu con el que se ha escrito este libro; tratar de contar, de forma sencilla, pero rigurosa y profunda, la disciplina de las Redes de Neuronas Artificiales. Por esto todos los capítulos teóricos del libro concluyen con ejemplos prácticos sencillos, que ayudan en la comprensión de los conceptos introducidos a lo largo del capítulo. Además, cuenta con tres capítulos de aplicaciones prácticas de las Redes de Neuronas Artificiales a problemas reales. Estos capítulos incluyen problemas difíciles, donde se realiza una descripción de los mismos: la estructura de los datos que los definen, en qué consiste la complejidad de cada uno de ellos y, por supuesto, la solución mediante Redes de Neuronas Artificiales. Estas soluciones incluyen la descripción de los modelos, los parámetros, cómo dichos parámetros influyen en la solución y las

distintas soluciones encontradas.

Las Redes de Neuronas Artificiales tienen una historia de alrededor de sesenta años, pero no fue sino a partir de los años ochenta cuando la disciplina alcanzó la madurez para poder ser incluida dentro de las materias de estudio dentro de la computación. Es, por tanto, una disciplina en plena pubertad, con todo lo que eso significa. Un periodo difícil, de mucho cambio, donde aparecen y desaparecen tendencias y corrientes, donde hay pasos hacia adelante y hacia atrás, donde en algunos aspectos no se ha alcanzado aún la madurez. En este libro se ha hecho un esfuerzo por tratar de explicar en detalle sólo aquellos aspectos de la disciplina que ya han alcanzado suficiente madurez. Está ubicada dentro de la Inteligencia Artificial debido a que es un paradigma de los que se conocen como resolutorios de problemas. Más concretamente se la encuadra dentro del Aprendizaje Automático, ya que es un sistema que adquiere habilidades a partir de ejemplos, pero también una vía para entender un poco mejor cómo funciona nuestro cerebro. El enfoque del libro es, sin embargo, el del paradigma matemático para la resolución de problemas, más que las relaciones entre las Redes Neuronales y sus analogías computacionales.

El libro está dividido en tres grandes partes. Por un lado se sigue la división clásica en dos grandes paradigmas: los modelos con aprendizaje supervisado y los modelos autoorganizados, los cuales ocupan las dos terceras partes del libro. Por otra lado están los capítulos de aplicaciones. En las dos primeras partes, de descripción de los modelos, se ha optado por la inclusión únicamente de los modelos más habituales, a pesar de lo tentadora que resulte la idea de mencionar modelos más recientes o variaciones y mejoras de los modelos que han demostrado ser de utilidad. Se ha preferido hacer hincapié en unos pocos modelos con la idea de dejar lo más claro posible los conceptos fundamentales, de manera que el lector, al final del libro, esté capacitado para acercarse de forma más eficaz a otras lecturas más avanzadas. De este modo, se han incluido los primeros modelos existentes (**PERCEPTRON** y **ADALINE**), el **RETROPROPAGACIÓN**, las Redes de Base Radial, los Mapas Autoorganizados de *Kohonen*, los modelos **ART** y los modelos más sencillos de Redes Recurrentes.

Pedro Isasi e Inés Galván. Madrid, octubre de 2003.

A la memoria de José Luis Isasi  
A Juan Carlos y Blanca

# Índice general

<b>1. Introducción a las Redes de Neuronas Artificiales</b>	<b>1</b>
1.1. Fundamentos biológicos de las Redes Neuronales . . . . .	2
1.2. Modelo computacional . . . . .	5
1.2.1. La neurona artificial . . . . .	5
1.2.2. Estructura básica de la red . . . . .	7
1.2.3. Aprendizaje . . . . .	10
1.3. Características de las Redes de Neuronas Artificiales . . . . .	14
1.3.1. Redes Neuronales frente a Redes de Neuronas Artificiales	14
1.3.2. Redes de Neuronas Artificiales frente a Computación convencional . . . . .	16
1.4. Historia de las Redes de Neuronas Artificiales . . . . .	18
1.4.1. La década de los 40 y 50 . . . . .	18
1.4.2. La década de los 60 . . . . .	19
1.4.3. La década de los 70 y 80 . . . . .	21
<b>2. Primeros modelos computacionales</b>	<b>23</b>
2.1. Células de McCulloch-Pitts . . . . .	23
2.1.1. Función lógica NOT . . . . .	24
2.1.2. Función lógica AND . . . . .	25
2.1.3. Función lógica OR . . . . .	25
2.2. PERCEPTRON . . . . .	26
2.2.1. Descripción del modelo . . . . .	27
2.2.2. Función lógica AND . . . . .	30
2.3. ADALINE . . . . .	34
2.3.1. Descripción del modelo . . . . .	35
2.3.2. Descodificador de binario a decimal . . . . .	37
2.3.3. Problema del OR exclusivo . . . . .	40
<b>3. Perceptrón multicapa</b>	<b>45</b>
3.1. Introducción . . . . .	45
3.2. Arquitectura del perceptrón multicapa . . . . .	46
3.2.1. Propagación de los patrones de entrada . . . . .	48
3.2.2. Diseño de la arquitectura del perceptrón multicapa . . . . .	51
3.3. Algoritmo de RETROPROPAGACIÓN . . . . .	52

3.3.1. Obtención de la regla delta generalizada . . . . .	54
3.3.2. Resumen de la regla delta generalizada . . . . .	60
3.3.3. Razón de aprendizaje. Inclusión del momento en la ley de aprendizaje . . . . .	62
3.4. Proceso de aprendizaje del Perceptrón multicapa . . . . .	63
3.4.1. Capacidad de generalización . . . . .	65
3.4.2. Deficiencias del algoritmo de aprendizaje . . . . .	68
3.5. Ejemplo de funcionamiento del perceptrón multicapa . . . . .	69
<b>4. Redes de neuronas de base radial</b>	<b>75</b>
4.1. Introducción . . . . .	75
4.2. Arquitectura de las redes de base radial . . . . .	76
4.2.1. Activaciones de las neuronas de la red de base radial . . . . .	77
4.2.2. Carácter local de las redes de base radial . . . . .	80
4.2.3. Diseño de la arquitectura de las redes de base radial . . . . .	81
4.3. Aprendizaje de las redes de base radial . . . . .	82
4.3.1. Método de aprendizaje híbrido . . . . .	83
4.3.2. Método de aprendizaje totalmente supervisado . . . . .	89
4.3.3. Combinando ambos métodos de aprendizaje: híbrido y totalmente supervisado . . . . .	95
4.4. Ejemplo de funcionamiento de las redes de base radial . . . . .	96
4.5. Redes de base radial frente a PERCEPTRON multicapa . . . . .	99
<b>5. Redes de neuronas recurrentes</b>	<b>103</b>
5.1. Introducción . . . . .	103
5.2. Red de Hopfield . . . . .	105
5.2.1. Aprendizaje y mecanismo de actuación de la red de Hopfield	107
5.2.2. Función energía . . . . .	108
5.3. Redes parcialmente recurrentes . . . . .	109
5.4. Redes totalmente recurrentes . . . . .	113
5.4.1. Algoritmo de retropropagación a través del tiempo . . . . .	114
5.4.2. Algoritmo de aprendizaje recurrente en tiempo real . . . . .	117
<b>6. Aprendizaje no supervisado</b>	<b>123</b>
6.1. Características básicas . . . . .	123
6.1.1. Categorización en redes neuronales biológicas . . . . .	125
6.1.2. Regla de Hebb . . . . .	127
6.1.3. Modelo de interacción lateral . . . . .	128
6.1.4. Aprendizaje competitivo . . . . .	131
6.2. Mapas autoorganizativos de Kohonen . . . . .	136
6.2.1. Descripción del modelo . . . . .	136
6.2.2. Ejemplos de aplicaciones . . . . .	145
6.3. Teoría de la resonancia adaptativa . . . . .	152
6.3.1. Arquitectura ART 1 . . . . .	152
6.3.2. Funcionamiento del modelo . . . . .	155
6.3.3. Ecuaciones del modelo ART 1 . . . . .	158

6.3.4. Características de los modelos ART . . . . .	163
6.3.5. Ejemplo, reconocedor de caracteres . . . . .	164
<b>7. Predicción de series temporales</b>	<b>171</b>
7.1. Introducción . . . . .	171
7.2. Problema de predicción . . . . .	173
7.3. Modelos neuronales para la predicción de series temporales . . . . .	174
7.3.1. Predicción en un paso de tiempo . . . . .	175
7.3.2. Predicción en múltiples pasos de tiempo . . . . .	176
7.4. Predicción del nivel del agua en la laguna de Venecia . . . . .	180
<b>8. Control de procesos dinámicos</b>	<b>185</b>
8.1. Introducción . . . . .	185
8.2. Modelización y control de procesos dinámicos . . . . .	186
8.3. Diferentes esquemas de control utilizando redes de neuronas . . . . .	187
8.4. Control de la temperatura en un reactor químico . . . . .	192
<b>9. Clasificación</b>	<b>197</b>
9.1. Tarea de clasificación . . . . .	197
9.1.1. Características . . . . .	198
9.1.2. Tipos de clasificadores . . . . .	201
9.2. Métodos de clasificación . . . . .	204
9.2.1. Redes de cuantización vectorial . . . . .	205
9.2.2. K medias . . . . .	206
9.3. Clasificación de enfermos de diabetes . . . . .	208
<b>Bibliografía</b>	<b>213</b>
<b>Índice alfabético</b>	<b>227</b>

# Capítulo 1

## INTRODUCCIÓN A LAS REDES DE NEURONAS ARTIFICIALES

Uno de los grandes enigmas que ha preocupado al hombre desde tiempos ancestrales es el de su propia naturaleza. Cuáles son las características que nos hacen humanos, qué tiene el hombre que no tienen el resto de los animales, qué nos hace únicos entre todos los seres vivos. Este enigma ha venido asociado con el de la inteligencia, pues dentro de la naturaleza humana está el ser inteligente, y ésta es una característica que discrimina absolutamente a nuestra especie. Es por esto por lo que el estudio de la inteligencia ha fascinado a filósofos y científicos desde siempre y ha sido un tema recurrente en tratados y libros, y sin embargo no se han producido avances significativos.

A medida que la ciencia y la tecnología han ido avanzando, el objetivo se ha ido perfilando: uno de los retos más importantes a los que se enfrenta el ser humano de nuestra generación es el de la construcción de sistemas inteligentes. Aquí, sistema puede ser entendido como cualquier dispositivo físico o lógico capaz de realizar la tarea requerida. Éste es precisamente el objetivo de la disciplina científica conocida con el nombre de Inteligencia Artificial.

Dentro de la Inteligencia Artificial se pueden distinguir dos grandes áreas. Una se ocupa de la construcción de sistemas con características que se puedan definir como inteligentes. A este campo se lo denomina Inteligencia Artificial Simbólica. En este caso, se define el problema a resolver y se diseña el sistema capaz de resolverlo siguiendo esquemas prefijados por la disciplina. Los Sistemas Expertos siguen este esquema: se introducen una serie de reglas lógicas que recogen el conocimiento de un experto sobre una materia, y mediante mecanismos de inferencia parecidos a los que empleamos al razonar, se sacan conclusiones. En la Inteligencia Artificial Simbólica se dice que los sistemas siguen un esquema de arriba hacia abajo (en inglés *top – down*) ya que es necesario disponer

de una aproximación a la solución del problema y diseñarla completamente.

Frente a esta perspectiva se encuentra la otra gran área de la Inteligencia Artificial, la Subsimbólica. En este caso no se realizan diseños a alto nivel de sistemas capaces de resolver los problemas utilizando las técnicas de la disciplina, sino que se parte de sistemas genéricos que van adaptándose y construyéndose hasta formar por sí mismos un sistema capaz de resolver el problema. Esto quedará más claro con un ejemplo. Una perspectiva simbólica consiste en el estudio de los mecanismos de razonamiento humano a alto nivel, cómo nos enfrentamos a un problema, cómo lo abordamos y resolvemos; y se elaboran programas que realizan las mismas tareas. Cuanto mejor haya podido entenderse la forma de razonar humana, más eficiente será el sistema producido a la hora de resolver los problemas planteados. La perspectiva subsimbólica trata de estudiar los mecanismos físicos que nos capacitan como seres inteligentes, frente a los programas de computador clásicos que son simples autómatas que obedecen órdenes muy concretas. El mecanismo fundamental que capacita a los seres vivos para la realización de tareas sofisticadas no preprogramadas directamente es el sistema nervioso. Desde este punto de vista la perspectiva subsimbólica trata de estudiar los mecanismos de los sistemas nerviosos, del cerebro, así como su estructura, funcionamiento y características lógicas, con la intención de diseñar programas basados en dichas características que se adapten y generen sistemas capaces de resolver problemas. En este caso el diseño es de abajo hacia arriba (*bottom – up*), ya que los sistemas diseñados son simples e idénticos, recogen las características físicas de los sistemas que tratan de imitar, y se van generando cálculos cada vez más complejos, de forma automática, mediante mecanismos prefijados de aprendizaje. Es en este campo donde se encuadran las Redes de Neuronas Artificiales.

Idealmente, el objetivo de las Redes de Neuronas Artificiales es llegar a diseñar máquinas con elementos neuronales de procesamiento paralelo, de modo que el comportamiento global de esa red “emule”, de la forma más fiel posible, los sistemas neuronales de los animales. Esto hace imprescindible el estudio profundo de los mecanismos que rigen el comportamiento de los sistemas neuronales. En la siguiente sección se describirán los fundamentos biológicos de los sistemas neuronales. Gran parte de las características que serán descritas son realizadas en los modelos artificiales de los que se va a hablar en este libro.

## 1.1. Fundamentos biológicos de las Redes Neuronales

El aparato de comunicación neuronal de los animales y del hombre, formado por el sistema nervioso y hormonal, en conexión con los órganos de los sentidos y los órganos efectores (músculos, glándulas), tiene la misión de recoger informaciones, transmitirlas y elaborarlas, en parte también almacenarlas y enviarlas de nuevo en forma elaborada. El sistema de comunicación neuronal se compone de tres partes:

1. Los receptores, que están en las células sensoriales, recogen las informaciones en forma de estímulos, bien del ambiente, bien del interior del organismo.
2. El sistema nervioso, que recibe las informaciones, las elabora, en parte las almacena y las envía en forma elaborada a los órganos efectores y a otras zonas del sistema nervioso.
3. Órganos diana o efectores (por ejemplo, músculos y glándulas), que reciben la información y la interpretan en forma de acciones motoras, hormonales, etc.

El elemento estructural y funcional más esencial, en el sistema de comunicación neuronal, es la célula nerviosa o neurona. La mayoría de las neuronas utilizan sus productos de secreción como señales químicas (transmisores) para la transmisión de la información. Dicha información se envía, entre las distintas neuronas, a través de prolongaciones, formando redes, en las cuales se elabora y almacena información. Además, una parte de las neuronas está en relación con receptores, a través de los cuales llegan comunicaciones procedentes del exterior o el interior del organismo hasta las redes neuronales.

Otra parte conduce las informaciones, elaboradas en forma de órdenes, hacia los efectores. Una de las prolongaciones es la encargada de la conducción de impulsos; se denomina axón.

A una distancia más o menos grande a partir de su origen, se ramifica y forma los botones terminales, que se ponen en contacto con otras neuronas o con células efectoras, pero sin llegar a fusionarse con ellas. A esta zona de contacto se la denomina sinapsis.

La misión de las neuronas comprende generalmente cinco funciones parciales:

- Las neuronas recogen la información que llega a ellas en forma de impulsos procedentes de otras neuronas o de receptores.
- La integran en un código de activación propio de la célula.
- La transmiten codificada en forma de frecuencia de impulsos a través de su axón.
- A través de sus ramificaciones el axón efectúa la distribución espacial de los mensajes.
- En sus terminales transmite los impulsos a las neuronas subsiguientes o a las células efectoras.

Un diagrama de una célula nerviosa típica es el que se muestra en la Figura 1.1.

En este esquema se aprecia que la neurona consta de un cuerpo celular y un núcleo, como el resto de las células del organismo, pero cuenta también con algunos elementos específicos. En primer lugar está el axón, que es una ramifications de salida de la neurona. A través de él se propagan una serie de impulsos

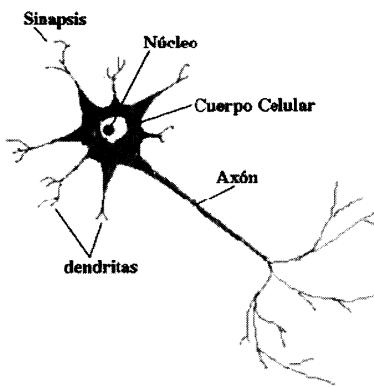


Figura 1.1: Descripción de una célula nerviosa típica

electro-químicos. Además, la neurona cuenta con un gran número de ramificaciones de entrada, las dendritas, que propagan la señal al interior de la neurona. El mecanismo es el siguiente. Las sinapsis recogen información electro-química procedente de las células vecinas a las que la célula en cuestión está conectada; esta información llega al núcleo donde es procesada hasta generar una respuesta que es propagada por el axón. Más tarde, la señal única propagada por el axón se ramifica y llega a dendritas de otras células a través de lo que se denomina sinapsis. Las sinapsis son los elementos de unión entre axón y dendritas. Es un espacio líquido donde existen determinadas concentraciones de elementos ionizados, normalmente iones de sodio y potasio. Estos iones hacen que el espacio intersináptico posea ciertas propiedades de conductividad que activen o impidan, en cierto grado, el paso del impulso eléctrico. De esta forma las sinapsis se convierten en potenciadores o inhibidores de la señal procedente de los axones, actuando como aislantes o amplificadores a conveniencia.

El funcionamiento en general será el de una enorme malla que propaga señales electro-químicas de unas células a otras y que va modificando sucesivamente la concentración de iones de las sinapsis. Esta concentración iónica es muy importante, ya que las neuronas no son elementos lineales; no se encargan simplemente de proyectar la acumulación de las señales recibidas por sus dendritas, sino que funcionan a saturación; producen una señal de activación si la señal recibida supera un cierto umbral, permaneciendo inhibidas mientras tanto.

La conectividad entre las células del cerebro es muy elevada. Se calcula que en cada cerebro existen alrededor de 100.000 millones de neuronas, conectadas cada una de ellas con alrededor de 10.000, es decir que cada actividad neuronal afecta a otras 10.000 neuronas, lo cual forma una red de un tamaño enorme. Un dispositivo de características similares es imposible de fabricar con la tecnología actual. Los intentos más cercanos que se han llevado a cabo han sido con redes de sólo un millón de procesadores en las cuales cada procesador se conectaba

únicamente con sus ocho adyacentes, lo cual queda varios órdenes de magnitud por debajo de lo deseable. Ésta es una de las grandes limitaciones que tienen los sistemas artificiales existentes en la actualidad.

Algunas de las células nerviosas, los receptores, reciben información directamente del exterior; a esta información se le da el nombre de estímulo. Los estímulos son el mecanismo de contacto de un organismo con el mundo exterior. Existen terminaciones nerviosas en casi todas las partes del organismo que se encargan de recibir la información visual, auditiva, táctil, etc. Ésta información, una vez elaborada, pasa a ser tratada como el resto de la información del sistema nervioso y convertida en impulsos electro-químicos. Son estos impulsos los que, básicamente, ponen en funcionamiento la red neuronal del sistema nervioso, la cual propaga todas las señales asincrónicamente, como se ha descrito, hasta que llega a los efectores, los órganos, glándulas, músculos; que son capaces de transformar la información recibida en acciones motoras, hormonales, etc. Así es como un organismo recibe, procesa y reacciona ante la información recibida del exterior.

Nuestro comportamiento, inteligente o no, y el del resto de los animales superiores, siguen un esquema de este tipo. Y son estos mecanismos los que tratan de incorporar los modelos artificiales que han ido apareciendo a lo largo de la historia de las Redes de Neuronas Artificiales.

## 1.2. Modelo computacional

La gran diferencia entre una máquina conexiónista, es decir, una máquina neuronal y los programas de computador convencionales es que éstas “elaboran”, en cierta medida, la información de entrada para obtener una salida o respuesta. No se trata de la aplicación ciega y automática de un algoritmo. De hecho, el proceso de elaboración de la información recibida depende de las distintas características, tanto estructurales como funcionales, de la red.

Existen modelos muy diversos de redes de neuronas en los cuales se siguen filosofías de diseño, reglas de aprendizaje y funciones de construcción de las respuestas muy distintas. Una primera clasificación se hace en función del recorrido que sigue la información dentro de la red, y así se distinguen redes alimentadas hacia adelante y redes con retro-alimentación.

A continuación se va a describir el modelo computacional general utilizado para desarrollar los diferentes sistemas de Redes de Neuronas Artificiales.

### 1.2.1. La neurona artificial

La neurona artificial, célula o autómata, es un elemento que posee un estado interno, llamado nivel de activación, y recibe señales que le permiten, en su caso, cambiar de estado.

Si se denomina  $S$  al conjunto de estados posibles de la neurona,  $S$  podrá ser, por ejemplo,  $S = \{0, 1\}$ , siendo 0 el estado inactivo y 1 el activo.  $S$  también podrá tomar un mayor número de valores,  $S = \{0, 1, 2, \dots, n\}$  para representar,

por ejemplo, una imagen con  $n+1$  niveles de gris, o incluso un intervalo continuo de valores, por ejemplo  $S = [0, 1]$ .

Las neuronas poseen una función que les permite cambiar de nivel de activación a partir de las señales que reciben; a dicha función se la denomina función de transición de estado o función de activación. Las señales que recibe cada neurona pueden provenir del exterior o de las neuronas a las cuales está conectada.

El nivel de activación de una célula depende de las entradas recibidas y de los valores sinápticos, pero no de anteriores valores de estados de activación. Para calcular el estado de activación se ha de calcular en primer lugar la entrada total a la célula,  $E_i$ . Este valor se calcula como la suma de todas las entradas ponderadas por ciertos valores.

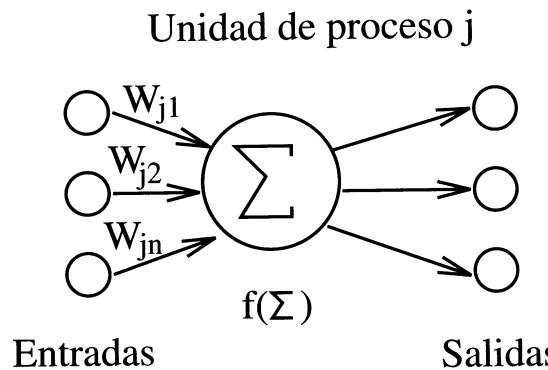


Figura 1.2: Esquema de una unidad de proceso típica

La Figura 1.2 muestra un modelo que representa esta idea. Aquí un grupo de entradas  $x_1, x_2, \dots, x_n$  son introducidas en una neurona artificial. Éstas entradas, definidas por un vector  $\bar{X}$ , corresponden a las señales de la sinapsis de una neurona biológica. Cada señal se multiplica por un peso asociado  $w_1, w_2, \dots, w_n$  antes de ser aplicado el sumatorio etiquetado por  $\Sigma$ . Cada peso corresponde a la fuerza de una conexión sináptica, es decir el nivel de concentración iónica de la sinapsis, y se representa por un vector  $\bar{W}$ .

El sumatorio, que corresponde al cuerpo de la neurona, suma todas las entradas ponderadas algebraicamente, produciendo una salida que se denomina  $E$ , así:

$$E = x_1w_1 + x_2w_2 + \dots + x_nw_n$$

Esto puede ser definido en forma vectorial como sigue:

$$E = X^T W$$

Las señales  $E$  son procesadas además por una función llamada función de activación o de salida  $\mathcal{F}$ , que produce la señal de salida de la neurona  $S$ . Dependiendo de la función  $\mathcal{F}$ , habrá distintos modelos de autómatas; por ejemplo:

- Lineal:  $S = KE$  con  $K$  constante.
- Umbral:  $S = 1$  si  $E \geq \theta$ ,  $S = 0$  si  $E < \theta$ ; siendo  $\theta$  el umbral constante.
- Cualquier función:  $S = \mathcal{F}(I)$ ; siendo  $\mathcal{F}$  una función cualquiera.

Más adelante se hará una descripción más detallada de los tipos de funciones de activación y de su importancia en un modelo de Red de Neuronas Artificial.

### 1.2.2. Estructura básica de la red

En la Figura 1.2 se muestra un ejemplo de una unidad típica de proceso de una Red de Neuronas Artificial. A la izquierda se ve una serie de entradas a la neurona; cada una llega de la salida de otra neurona de la red. Una vez calculada la salida de una neurona, como se ha explicado en el apartado anterior, ésta se propaga, vía conexiones de salida, a las células destino. Todas las conexiones de salida reciben el mismo valor de salida.

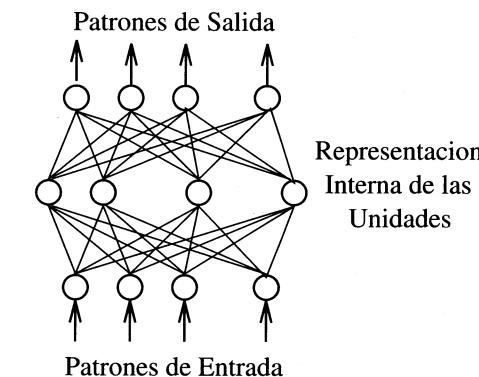


Figura 1.3: Esquema de una red de tres capas totalmente interconectadas

A la manera en que las células se conectan entre sí se la denomina patrón de conectividad o arquitectura de la red. La estructura básica de interconexión entre células es la de la red multicapa, mostrada en la Figura 1.3. Se trata de una estructura típica de implementación del paradigma conocido como RETRO-PROPAGACIÓN, que será descrito en el capítulo 3. El primer nivel lo constituyen las células de entrada; estas unidades reciben los valores de unos patrones representados como vectores que sirven de entrada a la red. A continuación hay una serie de capas intermedias, llamadas ocultas, cuyas unidades responden a rasgos particulares que pueden aparecer en los patrones de entrada. Puede haber uno o varios niveles ocultos. El último nivel es el de salida. La salida de estas unidades sirve como salida de toda la red.

Cada interconexión entre unidades de proceso actúa como una ruta de comunicación: a través de estas interconexiones viajan valores numéricos de una

célula a otra. Estos valores son evaluados por los pesos de las conexiones. Los pesos de las conexiones se ajustan durante la fase de aprendizaje para producir una Red de Neuronas Artificial final.

Así pues, una Red de Neuronas Artificial podría definirse como un grafo cuyos nodos están constituidos por unidades de proceso idénticas, y que propagan información a través de los arcos. En este grafo se distinguen tres tipos de nodos: los de entrada, los de salida y los intermedios.

El funcionamiento de la red es simple. Para cada vector de entrada, éste es introducido en la red copiando cada valor de dicho vector en la célula de entrada correspondiente. Cada célula de la red, una vez recibida la totalidad de sus entradas, las procesa y genera una salida que es propagada a través de las conexiones entre células, llegando como entrada a la célula destino. Una vez que la entrada ha sido completamente propagada por toda la red, se producirá un vector de salida, cuyos componentes son cada uno de los valores de salida de las células de salida.

Así pues, el esquema de funcionamiento de una Red de Neuronas por capas como la de la Figura 1.3 puede describirse mediante la ecuación:

$$\vec{S} = F(F(\vec{X} \cdot W_1) \cdot W_2) \quad (1.1)$$

donde  $W_1$  y  $W_2$  son los pesos de la primera y segunda capa, respectivamente;  $F$  es la función de activación idéntica en todas las neuronas;  $\vec{X}$  es el vector de entrada a la red, y  $\vec{S}$  es el vector de salida que la red produce.  $W_1$  y  $W_2$  son matrices de conexiones entre las capas de la red, y por lo tanto se trata de multiplicaciones de matrices. La función de activación  $F$  desempeña un papel importante en un esquema de Red de Neuronas. Supóngase que se utiliza una función lineal del tipo comentado anteriormente:  $F(x) = k \cdot x$ . En este caso, si sustituimos dicha función en la Ecuación 1.1 quedará como sigue:

$$\vec{S} = k \cdot (K \cdot \vec{X} \cdot W_1) \cdot W_2$$

Lo cual es equivalente a una red con una sola capa de conexiones  $W_t$  donde:

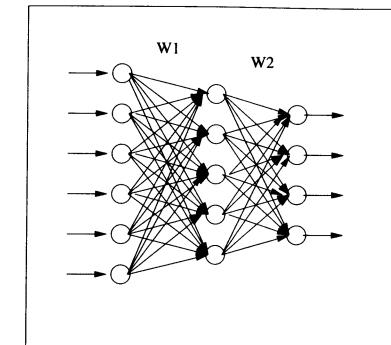
$$W_t = k^2 \cdot W_1 \cdot W_2$$

Se va a ilustrar esto con un ejemplo. Sea la red de la Figura 1.5, donde:

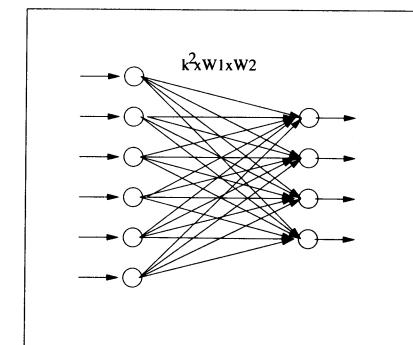
$$W_1 = \begin{pmatrix} 3 & 2 & 3 \\ 4 & 1 & 2 \\ 5 & 2 & 1 \\ 6 & 1 & 1 \end{pmatrix} \quad W_2 = \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 1 & 1 \end{pmatrix}$$

Si la función de activación es  $F(x) = 2x$  y se introduce el vector  $\vec{x} = \{-3, 1, 4, -2\}$ , según la Ecuación 1.1 la salida será:

$$\vec{A} = \vec{X} \cdot W_1 = (-3 \ 1 \ 4 \ -2) \cdot \begin{pmatrix} 3 & 2 & 3 \\ 4 & 1 & 2 \\ 5 & 2 & 1 \\ 6 & 1 & 1 \end{pmatrix} = (3 \ 1 \ -5)$$



Red de dos capas



Red de una capa equivalente

Figura 1.4: Equivalencia entre redes al utilizar funciones de activación lineales

$$\vec{S} = F(F(\vec{A}) \cdot W_2) = (6 \ 2 \ -10) \cdot \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 1 & 1 \end{pmatrix} = F((4 \ 12)) = (8 \ 24)$$

La misma salida se obtendría con una sola capa con una matriz de conexiones calculada de la siguiente manera:

$$W_t = k^2 \cdot W_1 \cdot W_2 = 4 \cdot \begin{pmatrix} 3 & 2 & 3 \\ 4 & 1 & 2 \\ 5 & 2 & 1 \\ 6 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 44 & 64 \\ 44 & 64 \\ 52 & 80 \\ 56 & 84 \end{pmatrix}$$

Con una sola capa la salida, para el vector de entrada anterior, se calcularía simplemente como:

$$\vec{S} = \vec{X} \cdot W_t = (-3 \ 1 \ 4 \ -2) \cdot \begin{pmatrix} 44 & 64 \\ 44 & 64 \\ 52 & 80 \\ 56 & 84 \end{pmatrix} = (8 \ 24)$$

Como se ve, se obtiene la misma salida que anteriormente. Esto quiere decir que si la función de activación es lineal, el introducir más capas en la red es irrelevante; existirá siempre una red con una sola capa equivalente a cualquier otra con un número de capas arbitrario, y su cálculo es trivial. Más adelante se demostrarán las limitaciones de las redes de una sola capa; por lo tanto, el incluir funciones de activación lineales hará que la red pierda gran parte de sus capacidades. Todas las redes cuentan con funciones de activación no lineales, que hacen que el potencial de la red para solucionar problemas de forma genérica sea elevado.

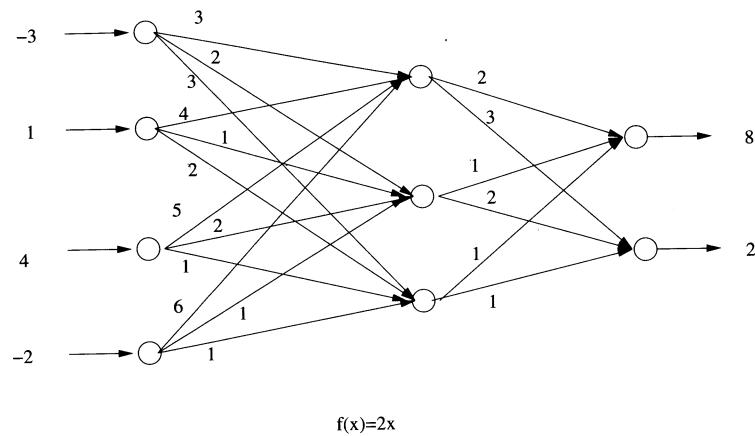


Figura 1.5: Red de dos capas

### 1.2.3. Aprendizaje

La parte más importante de una Red de Neuronas Artificial es el aprendizaje. El esquema de aprendizaje de una red es lo que determina el tipo de problemas que será capaz de resolver. Las Redes de Neuronas Artificiales son sistemas de aprendizaje basados en ejemplos. La capacidad de una red para resolver un problema estará ligada de forma fundamental al tipo de ejemplos de que dispone en el proceso de aprendizaje. Desde el punto de vista de los ejemplos, el conjunto de aprendizaje debe poseer las siguientes características:

- Ser significativo. Debe haber un número suficiente de ejemplos. Si el conjunto de aprendizaje es reducido, la red no será capaz de adaptar sus pesos de forma eficaz.
- Ser representativo. Los componentes del conjunto de aprendizaje deberán ser diversos. Si un conjunto de aprendizaje tiene muchos más ejemplos de un tipo que del resto, la red se especializará en dicho subconjunto de datos y no será de aplicación general. Es importante que todas las regiones significativas del espacio de estados estén suficientemente representadas en el conjunto de aprendizaje.

El aprendizaje en una Red de Neuronas Artificial consiste en la determinación de los valores precisos de los pesos para todas sus conexiones, que la capacite para la resolución eficiente de un problema. El proceso general de aprendizaje consiste en ir introduciendo paulatinamente todos los ejemplos del conjunto de aprendizaje, y modificar los pesos de las conexiones siguiendo un determinado esquema de aprendizaje. Una vez introducidos todos los ejemplos se comprueba si se ha cumplido cierto criterio de convergencia; de no ser así se repite el

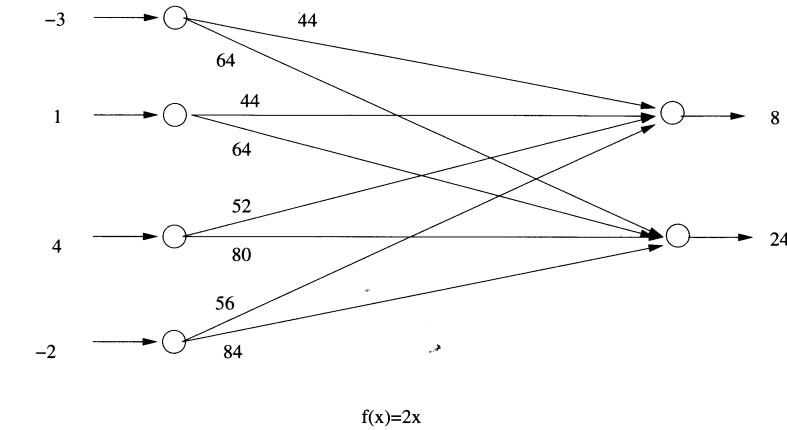


Figura 1.6: Red de una capa

proceso y todos los ejemplos del conjunto vuelven a ser introducidos. La modificación de los pesos puede hacerse después de la introducción de cada ejemplo del conjunto, o una vez introducidos todos ellos.

El criterio de convergencia depende del tipo de red utilizado o del tipo de problema a resolver. La finalización del periodo de aprendizaje se puede determinar:

- Mediante un número fijo de ciclos. Se decide a priori cuántas veces será introducido todo el conjunto, y una vez superado dicho número se detiene el proceso y se da por aceptada la red resultante.
- Cuando el error descienda por debajo de una cantidad preestablecida. En este caso habrá que definir en primer lugar una función de error, bien a nivel de patrón individual, bien a nivel de la totalidad del conjunto de entrenamiento. Se decide a priori un valor aceptable para dicho error, y sólo se para el proceso de aprendizaje cuando la red produzca un valor de error por debajo del prefijado. Para este criterio puede suceder que la red jamás consiga bajar por debajo del nivel prefijado, en cuyo caso se debe disponer de un criterio adicional de parada, por ejemplo un número de ciclos, que de utilizarse por la red significará que ésta no ha convergido. En este caso la red se dice que no ha sido capaz de obtener una solución. Será necesario probar cambiando alguno de los parámetros.
- Cuando la modificación de los pesos sea irrelevante. En algunos modelos se define un esquema de aprendizaje que hace que las conexiones vayan modificándose cada vez con menor intensidad. Si el proceso de aprendizaje continúa, llegará un momento en que ya no se producirán variaciones de los valores de los pesos de ninguna conexión; en ese momento se dice que la red ha convergido y se detiene el proceso de aprendizaje.

Dependiendo del esquema de aprendizaje y del problema a resolver, se pueden distinguir tres tipos de esquemas de aprendizaje:

- Aprendizaje supervisado. En este tipo de esquemas, los datos del conjunto de aprendizaje tienen dos tipos de atributos: los datos propiamente dichos y cierta información relativa a la solución del problema. Por ejemplo, si se trata de definir un clasificador para un conjunto de datos, un sistema capaz de distinguir entre caras de diferentes personas, los ejemplos contendrán datos del individuo, una imagen de su cara, e información de la solución, de qué persona se trata (una etiqueta que la distinga). El esquema de aprendizaje supervisado utilizará esta información para modificar las conexiones. La manera más habitual de modificar los valores de los pesos de las conexiones es la representada en la Figura 1.7. Cada vez que un ejemplo es introducido y se procesa para obtener una salida, dicha salida se compara con la salida que debería haber producido, y de la que se dispone al estar incluida dicha información en el conjunto de aprendizaje. La diferencia entre ambas influirá en cómo se modificarán los pesos. Si los dos datos son muy diferentes, se modificarán mucho los pesos si son parecidos la modificación será menor. Para este tipo de aprendizaje, se dice que hay un profesor externo encargado de determinar si la red se está comportando de forma adecuada, mediante la comparación entre la salida producida y la esperada, y de actuar en consecuencia modificando apropiadamente los valores de los pesos.

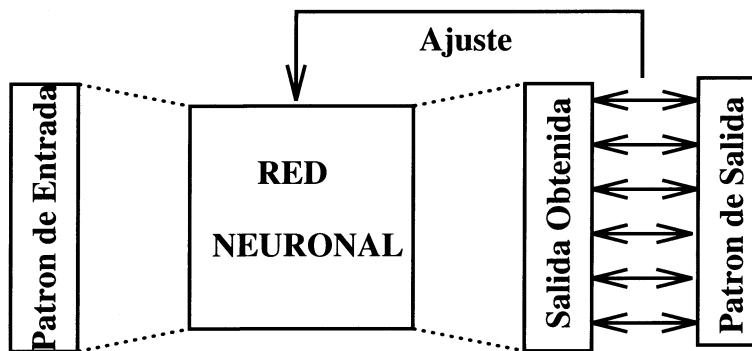


Figura 1.7: Aprendizaje supervisado

- Aprendizaje no supervisado. En este aprendizaje los datos del conjunto de aprendizaje sólo tienen información de los ejemplos, y no hay nada que permita guiar en el proceso de aprendizaje. En este caso, no existe profesor externo que determine el aprendizaje. En la Figura 1.8 se representa este tipo de aprendizaje. La red modificará los valores de los pesos a partir de información interna. Cuando se utiliza aprendizaje no supervisado, la red trata de determinar características de los datos del conjunto de entrenamiento: rasgos significativos, regularidades o redundancias. A este tipo de

modelos se los conoce también como sistemas autoorganizados, debido a que la red se ajusta dependiendo únicamente de los valores recibidos como entrada.

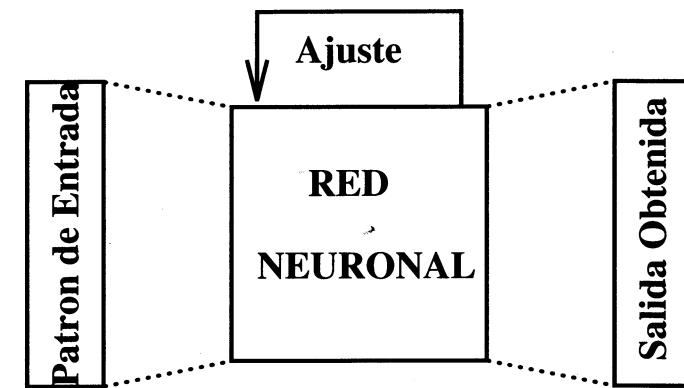


Figura 1.8: Aprendizaje no supervisado

- Aprendizaje por refuerzo. Es una variante del aprendizaje supervisado en el que no se dispone de información concreta del error cometido por la red para cada ejemplo de aprendizaje, sino que simplemente se determina si la salida producida para dicho patrón es o no adecuada.

Para el caso del aprendizaje por refuerzo, el aprendizaje tiene una serie de características específicas que es importante resaltar. En este caso el conjunto de aprendizaje está compuesto por ejemplos que contienen los datos y sus salidas deseadas. El proceso consiste en modificar los pesos de la red hasta que para todos los ejemplos del conjunto de entrenamiento, la salida producida sea lo más parecida posible a la deseada. Sin embargo, esto no siempre indica que la red será capaz de solucionar el problema, pues lo importante no es que el sistema dé buenas salidas sobre el conjunto de aprendizaje, que ya son conocidas, sino sobre los datos que puedan presentarse en el futuro, y cuyas salidas se desconocen.

Supóngase que se necesita entrenar una red neuronal para determinar la dirección del frente principal de fuego de un incendio. Se dispone de información pasada sobre incendios. Dicha información constará de un registro por incendio y hora, del que se sabe la humedad del ambiente, la dirección del viento, la temperatura, el tipo de terreno en cada unidad de superficie, la intensidad del viento, el tipo de vegetación, etc. y además, en qué dirección se desplazó el foco principal del incendio dos horas después. Si se entrena una red con estos datos, lo importante no es que ajuste adecuadamente los datos de entrenamiento, sino que sea capaz de predecir dónde se situará el foco de un incendio futuro dos horas después del momento de realizar la predicción, y esto es algo que se desconoce, por lo cual no puede ser utilizado en el proceso de aprendizaje. Lo que ocurre en

muchos casos es que un ajuste muy bueno del conjunto de aprendizaje lleva a malas predicciones. En estos casos se dice que la red ha padecido un sobreajuste de los datos de entrenamiento, y que su capacidad de generalización se ha visto reducida. Es decir, que el error producido por los datos de entrenamiento no es una buena medida de la capacidad de predicción, o de generalización, de la red.

Para poder determinar si la red produce salidas adecuadas, se divide el conjunto de entrenamiento en dos conjuntos que se llamarán de entrenamiento y de validación. El conjunto de entrenamiento se utiliza para aprender los valores de los pesos, como se ha descrito anteriormente. La diferencia es que en vez de medirse el error en el conjunto de entrenamiento, se utiliza el de validación. De esta manera, para medir la eficacia de la red para resolver el problema, se utilizarán datos que no han sido utilizados para su aprendizaje. Si el error sobre el conjunto de validación es pequeño, entonces quedará garantizada la capacidad de generalización de la red.

Para que este proceso sea eficaz los conjuntos de entrenamiento y validación deben tener las siguientes características:

- El conjunto de validación debe ser independiente del de aprendizaje. No puede haber ningún tipo de sesgo en el proceso de selección de los datos de validación.
- El conjunto de validación debe cumplir las propiedades de un conjunto de entrenamiento, descritas anteriormente.

Además, el conjunto de validación puede utilizarse durante el aprendizaje para guiarlo en conjunción con el de entrenamiento. En este caso el proceso sería el siguiente:

1. Asignar a los pesos valores aleatorios.
2. Introducir todos los ejemplos del conjunto de entrenamiento, modificando los pesos de acuerdo con el esquema de aprendizaje supervisado elegido.
3. Introducir todos los ejemplos del conjunto de validación. Obtener el error producido al predecir dichos ejemplos.
4. Si el error calculado en el paso anterior está por encima de cierto valor umbral, ir a (2).
5. Acabar el proceso de aprendizaje y dar como salida la red obtenida.

### 1.3. Características de las Redes de Neuronas Artificiales

#### 1.3.1. Redes Neuronales frente a Redes de Neuronas Artificiales

Una de las diferencias fundamentales entre los Sistemas Biológicos y las Redes de Neuronas Artificiales es la complejidad de las sinapsis. En los Sistemas

Biológicos estos puntos de interconexión tienen miles de componentes y de procesos activos de propagación de los impulsos electro-químicos. En cambio, las Redes de Neuronas Artificiales tienen conexiones relativamente simples, en las que, por lo general, se realiza una suma ponderada de las entradas, a la que se le aplica una función de umbral, lo que hace casi ridícula la comparación con la complejidad de las sinapsis biológicas.

Por otro lado, en los Sistemas Biológicos la información se propaga por medio de impulsos electro-químicos que al llegar a las células, dependiendo de su intensidad y de una serie de factores fisiológicos de las neuronas, producirán una serie de reacciones en estas. Éstos impulsos se reciben en cualquier momento, de forma que el funcionamiento de los Sistemas Biológicos se puede calificar como de totalmente asincrónico, y permite la reconfiguración de la red en cualquier instante de tiempo, por lo que producen patrones temporales de forma continua. En cambio, en las Redes de Neuronas Artificiales, los parámetros se actualizan de forma periódica, en intervalos de tiempo discretos, y, por lo general, todos a la vez; con lo cual se le presentan los patrones en intervalos de tiempo prefijados, y no los podrá recoger de forma esporádica de su entorno.

Además, las Redes de Neuronas tienen la propiedad de poder aprender, a partir de unas pocas presentaciones de patrones (un pequeño período de entrenamiento). Por el contrario, las Redes de Neuronas Artificiales normalmente convergen muy lentamente, y pueden necesitar de cientos o miles de presentaciones de patrones para llegar a realizar una generalización aceptable.

En cuanto a las interconexiones entre unidades de proceso, las Redes de Neuronas Artificiales suelen tener unas arquitecturas simples (por capas, interconectadas todas con todas, autoasociativas,etc.) y fijas, mientras que las Redes Neuronales están estructuradas por niveles: córtex, circunvoluciones, cerebelo, hipocampo; todos ellos con estructuras que no son simples filas de unidades interconectadas, sino que forman una malla de conexiones muy densa, sin estructura aparente, y variable con el tiempo.

Otra de las limitaciones importantes de los sistemas automáticos estriba en que centran su aprendizaje y funcionamiento en una única tarea en concreto –no existe una red universal para cualquier tipo de problema–, al contrario de los Sistemas Biológicos, que pueden aprender simultáneamente un gran número de tareas de tipos muy diversos.

Por su parte, las Redes de Neuronas Artificiales tienen la ventaja de que los valores transferidos de unidad a unidad pueden ser calculados de forma relativamente precisa, al contrario de lo que ocurre en las biológicas, cuyos impulsos no se prestan a una medida precisa de los valores implicados en las reacciones electro-químicas.

A pesar de estar muy lejos de conseguir representar todas las características de las Redes Neuronales, las Redes de Neuronas Artificiales tienen en común con ellas algunas de esas características, como son las de representación y procesamiento distribuido de la información, o la de redundancia intrínseca de la información representada, que permite un perfecto funcionamiento del sistema aun estando dañado alguno de sus componentes (tolerancia a fallos). Esto será comentado a continuación con más detalle.

### 1.3.2. Redes de Neuronas Artificiales frente a Computación convencional

Las máquinas Von Neumann tradicionales tienen una única CPU que realiza secuencialmente todas sus computaciones. Una CPU clásica es capaz de realizar una centena o más de comandos básicos, incluidas sumas, restas, cargas y desplazamientos, y otras. Estos comandos son ejecutados uno cada vez, en pasos sucesivos de reloj. Por el contrario, una unidad de proceso en Redes de Neuronas Artificiales puede realizar sólo una o unas pocas operaciones diferentes.

El poder de proceso de una Red de Neuronas Artificial viene dado por el número de actualizaciones por segundo de sus conexiones; por el contrario, en las máquinas Von Neumann se mide por el número de instrucciones realizadas por segundo, secuencialmente, por un procesador. En las Redes de Neuronas Artificiales, durante la fase de aprendizaje se produce el ajuste de los parámetros asociados a las interconexiones entre neuronas. Por tanto, la medida del aprendizaje vendrá asociada a la medida de las interconexiones actualizadas.

Las Redes de Neuronas Artificiales también se diferencian de las clásicas arquitecturas paralelas. Primero, los procesadores en una Red de Neuronas Artificial están masivamente interconectados. Como resultado hay más conexiones que unidades de proceso, en contra de lo que ocurre en las máquinas paralelas clásicas. Por otra parte, las arquitecturas paralelas clásicas tratan de incorporar procesadores comparables en complejidad con los de las máquinas Von Neumann; las Redes de Neuronas Artificiales, en cambio, trasladan el poder computacional a las conexiones, no se necesitan poderosos procesadores.

Los computadores tradicionales operan secuencialmente porque sólo tienen una unidad de proceso. Construir computadores con varias unidades de proceso es muy costoso y complejo. A su vez, la complejidad del diseño global de la arquitectura del computador aumenta muchísimo. Por otro lado, para utilizar la capacidad de proceso de estos computadores los problemas deben poderse dividir fácilmente en tareas independientes capaces de ejecutarse simultáneamente en cada una de las unidades de proceso del computador paralelo. Sin embargo, el ámbito de aplicación de los computadores es mucho más general, y este tipo de problemas no se presentan con frecuencia.

Las Redes de Neuronas Artificiales no son programadas, aprenden a partir de ejemplos. Normalmente, a una Red de Neuronas Artificial le presentamos una serie de patrones ejemplo de los cuales ella debe aprender. Estos ejemplos, o patrones de entrenamiento, vienen representados por vectores, y pueden ser obtenidos por fuentes de imágenes, sensores, movimientos del brazo de un robot, etc.

Debido a que el aprendizaje es por ejemplos, las Redes de Neuronas Artificiales tienen un gran potencial para la construcción de sistemas de computación que no necesitan ser programados. Esto supone un enfoque radicalmente distinto de los sistemas de computación, comparados con los métodos tradicionales, los cuales necesitan siempre programas de desarrollo. En un programa de computador, cada paso que el computador ejecuta debe ser anteriormente especificado por el programador, un proceso que absorbe tiempo y recursos humanos. Las

Redes de Neuronas Artificiales, por el contrario, comienzan con ejemplos de entradas y salidas y aprenden a producir la salida correcta para cada entrada.

El enfoque de Redes de Neuronas Artificiales no requiere identificación de características, ni el desarrollo de algoritmos y programas para la resolución de problemas particulares; aun así, las Redes de Neuronas Artificiales tienen dos grandes inconvenientes:

- El tiempo de aprendizaje de la red no puede ser conocido a priori. No se puede determinar el número de veces que será necesario introducir todo el conjunto de datos para que la red aprenda. En muchos problemas de tiempo real esto puede ser un inconveniente importante. Además, este tiempo no tiene por qué estar en relación con el tamaño del conjunto de aprendizaje. Los conjuntos pequeños pueden requerir muchas pasadas, lo que redundaría en un mayor tiempo de aprendizaje. Por el contrario, los conjuntos grandes pueden contener elementos no significativos y redundantes que hagan que el aprendizaje se produzca rápidamente.
- El diseño de una red para resolver un problema con éxito puede ser una tarea muy compleja y larga. El método común para diseñar una buena red de neuronas es mediante prueba y error. Esto quiere decir que hasta que una red no ha sido aprendida no puede ser descartada y, lo que es peor, no tiene por qué dar indicios del camino a seguir, lo cual hace del procedimiento algo tedioso y no sistemático.

Las Redes de Neuronas Artificiales codifican la información de manera distribuida. Normalmente la información que es almacenada en una Red de Neuronas Artificial es compartida por varias de sus unidades de proceso. Este tipo de almacenamiento está en evidente contraste con los esquemas tradicionales de memoria, donde las piezas de información son almacenadas en lugares específicos de memoria. Los sistemas tradicionales de reconocimiento de lenguaje, por ejemplo, contienen una tabla de patrones de voz, que son comparados uno a uno por los vocablos de entrada. Las tablas están almacenadas en unas posiciones fijas de memoria. Las Redes de Neuronas Artificiales, por el contrario, reconocen sílabas utilizando varias unidades de proceso al mismo tiempo. La representación interna, en este caso, se distribuye a través de toda o parte de la red. Por tanto, una misma zona de la red puede almacenar la misma sílaba o patrón.

Este tipo de esquemas de almacenamiento distribuido tiene muchas ventajas; la más importante radica en que la información es almacenada de forma redundante, lo que permite un correcto funcionamiento de la red, aun cuando el sistema pueda haber sufrido una parcial destrucción. Aunque la redundancia puede ser implementada en otros tipos de sistemas, las Redes de Neuronas Artificiales disponen de una vía natural de organizar e implementar esta redundancia; el resultado es un sistema que resulta tolerante a fallos, por sí mismo. Esto está en concordancia con la tolerancia a fallos de nuestros propios sistemas neuronales. Las células nerviosas son las únicas células del organismo que no se regeneran; a partir de alrededor de los 25 años los humanos empiezan a perder

neuronas, que jamás son recuperadas. Sin embargo, esta pérdida no merma en absoluto ninguna de las capacidades de nuestros sistemas neuronales. En los sistemas artificiales, en general, esta característica se conserva, y si la red es suficientemente grande, la eliminación de alguna de sus unidades no afectará a su funcionamiento.

Las redes pueden ser entrenadas para resolver problemas de forma genérica, y no sólo para memorizar los patrones de entrenamiento, siempre que estos patrones representen adecuadamente al problema, como se ha comentado en el apartado de aprendizaje. Esta característica es importante y hay que tenerla en cuenta, ya que existen algunos problemas que no pueden ser descritos de forma exhaustiva mediante ejemplos, y no es posible programar procedimientos para su resolución.

## 1.4. Historia de las Redes de Neuronas Artificiales

Las primeras investigaciones en Redes de Neuronas Artificiales datan de principios del siglo XIX, con algunos de los trabajos realizados por **Freud** en el periodo del presicoanálisis [Strachey, 1966]. La primera implementación de Redes de Neuronas Artificiales fue un dispositivo hidráulico descrito por **Russell** [Russell, 1913]. Pero no fue hasta la década de los 40, ya en el siglo XX, cuando el estudio en Redes de Neuronas Artificiales cobró una fuerza que se ha ido incrementando hasta la actualidad, gracias al trabajo de varios científicos brillantes y de los increíbles avances del hardware. Cabe destacar varios científicos que han conseguido que las Redes de Neuronas ocupen el lugar relevante del que gozan actualmente. En los siguientes apartados se describirán los primeros pasos y más importantes de la historia de las Redes de Neuronas Artificiales.

### 1.4.1. La década de los 40 y 50

**Warren McCulloch y Walter Pitts** [McCulloch and Pitts, 1943] realizaron el primer modelo matemático de unas Redes de Neuronas Artificiales. El modelo de McCulloch-Pitts) está basado en la idea de que las neuronas operan mediante impulsos binarios. Este modelo introduce la idea de una función de paso por umbral utilizada posteriormente por muchos modelos como las Redes de Neuronas Artificiales discretas de Hopfield [Hopfield, 1982] y la memoria asociativa bidireccional discreta [Kosko, 1988]. Aunque este modelo generó gran interés al proporcionar medidas de comportamiento sofisticadas a través de cálculos sencillos, el factor clave del modelo está en la capacidad de aprendizaje. Este trabajo y otros fascinantes experimentos se describen con detalle en el libro *Embodiments of Mind* [McCulloch, 1965].

**Donald Hebb** desarrolló posteriormente un procedimiento matemático de aprendizaje. Los estudios de Hebb sobre las neuronas y las condiciones clásicas de aprendizaje se describen en su libro *Organization of Behavior* [Hebb, 1949],

## Historia de las RNA

en el que desarrolla un paradigma de aprendizaje que ahora lleva su nombre –aprendizaje hebbiano–.

En 1951, **Marvin Minsky** obtuvo los primeros resultados prácticos en Redes de Neuronas Artificiales [Minsky, 1954]. Inspirados por los trabajos de McCulloch y Pitts, Minsky y Edmons diseñaron una máquina con 40 neuronas cuyas conexiones se ajustaban de acuerdo con una serie de sucesos que ocurrían al realizar ciertas tareas<sup>1</sup>. La máquina estaba construida con tubos, motores y relés, y pudo modelizar con éxito el comportamiento de una rata buscando comida en un laberinto.

Más tarde, **Albert Uttley** [Uttley, 1956a, Uttley, 1956b] comenzó a desarrollar nuevos paradigmas de Redes de Neuronas Artificiales, creando una máquina teórica compuesta de informes (elementos de proceso). El informe era un separador lineal que ajustaba sus parámetros de entrada utilizando la medida de entropía de Shannon. Éstas máquinas han sido utilizadas para simular fenómenos atmosféricos [Uttley, 1966, Uttley, 1976a], así como para el reconocimiento adaptativo de patrones [Uttley, 1975, Uttley, 1976b].

En 1957, **Frank Rosenblatt** generalizó el modelo de células de McCulloch-Pitts añadiéndole aprendizaje [Rosenblatt, 1957] [Rosenblatt, 1958]; llamó a este modelo el PERCEPTRON. Primero desarrolló un modelo de dos niveles, que ajustaba los pesos de las conexiones entre los niveles de entrada y salida, en proporción al error entre la salida deseada y la salida obtenida. Rosenblatt intentó extender su procedimiento de aprendizaje a un PERCEPTRON de tres niveles, pero no encontró un método matemático sólido para entrenar la capa de conexiones oculta. Algunos de los trabajos que Rosenblatt realizó en torno al PERCEPTRON están recogidos en su libro *Principles of Neurodynamics* [Rosenblatt, 1962].

Dos años después, **Bernard Widrow** [Widrow, 1959, Widrow, 1960] diseñó una Red Neural Artificial muy similar al PERCEPTRON, llamada *Adaptive Linear Element* o ADALINE. El ADALINE de dos niveles, muy parecido al PERCEPTRON, ajusta los pesos entre los niveles de entrada y salida en función del error entre el valor esperado de salida y el obtenido. La diferencia entre los dos modelos es muy pequeña, pero las aplicaciones a las que van dirigidas son muy distintas. En 1960, Widrow y Marcian Hoff probaron matemáticamente que en determinadas circunstancias el error entre la salida deseada y la obtenida puede ser minimizado hasta el límite que queramos. Tanto el PERCEPTRON como el ADALINE mantienen el problema de la separabilidad lineal. El ADALINE ha sido utilizado para procesamiento adaptativo de señales [Widrow et al., 1963, Widrow and Stearns, 1985, Widrow et al., 1975], sistemas de control [Widrow and Smith, 1964, Widrow, 1988] y sistemas adaptativos de antenas [Widrow et al., 1967].

### 1.4.2. La década de los 60

**Steinbuch** fue uno de los primeros investigadores en desarrollo de métodos de codificación de información en Redes de Neuronas [Steinbuch, 1961]. Las

<sup>1</sup> Aprendizaje hebbiano.

redes de Steinbuch se aplicaron al reconocimiento de escritura a mano distorsionada, mecanismos de diagnóstico de fallos de maquinarias y control de múltiples procesos en producción [Steinbuch and Piske, 1963].

**Stephen Grossberg** es el más influyente y formal de todos los investigadores en Redes de Neuronas Artificiales. Grossberg realizó importantes estudios sobre procesos y fenómenos psicológicos (mente) y biológicos (cerebro) de procesamiento humano de la información e intentó juntar los dos (mente y cerebro) en una teoría unificada [Grossberg, 1964]. Los trabajos de Grossberg incluyen estrictos análisis matemáticos, que permitieron la realización de nuevos paradigmas de Redes de Neuronas. Éstos permitían tener un acceso directo a la información mientras se operaba en tiempo real.

Grossberg formó, también, un grupo de investigación en la Universidad de Boston llamado Center for Adaptive Systems. Con este grupo, Grossberg investigó todas las facetas del procesamiento de información por parte de los humanos. Las investigaciones realizadas por Grossberg y sus colegas cubren todo el espectro de las Redes de Neuronas Artificiales.

A finales de la década de los 60 y principio de la de los 70, apareció una serie de investigaciones de interés en este campo, algunas de las cuales merecen ser destacadas.

**Shun-Ichi Amari** combinó la actividad de las redes neuronales biológicas con rigurosos modelos matemáticos de Redes de Neuronas Artificiales. Uno de éstos es una solución al famoso problema, durante mucho tiempo irresoluble, de la asignación de créditos. Sus estudios incluyen el tratamiento de Redes de Neuronas Artificiales dinámicas y aleatoriamente conectadas [Amari, 1971, Amari, 1972, Amari, 1974], estudios de aprendizaje competitivo [Amari, 1977, Amari, 1983, Amari and Takeuchi, 1978], así como el análisis matemático de memorias asociativas [Amari, 1982].

**James Anderson**, que trabajó con un modelo de memoria basado en la asociación de activaciones de las sinapsis de una neurona [Anderson, 1968], realizó un modelo de memoria asociativa lineal [Anderson, 1973], siguiendo el planteamiento de Hebb. Empleó un nuevo método de corrección de error, y sustituyó la función umbral lineal por otra en rampa, creando un nuevo modelo llamado *Brain-state-in-a-box* (BSB) [Anderson and Murphy, 1986].

También en el año 1968, investigadores del Departamento de Máquinas Inteligentes de la Universidad de Edimburgo fueron los primeros en descubrir la relación entre los hologramas y las memorias asociativas. Ambos mecanismos son capaces de obtener un patrón a partir de unas pocas pistas. **Longuet y Higgins** crearon un sistema de ecuaciones codificadas para almacenar y recuperar una secuencia de señales [Longuet-Higgins, 1968]. Más tarde, **Willshaw y Buneman**, junto con Longuet y Higgins, introdujeron los principios holográficos, como un posible mecanismo en los procesos de memoria humanos [Willshaw et al., 1969, Willshaw and Longuet-Higgins, 1969]. De este trabajo de la Universidad de Edimburgo surgió un modelo temporal de Red Neuronal Artificial llamado HOLOPHONE [Willshaw and Longuet-Higgins, 1970], un paradigma que almacena señales de entrada y puede obtener una señal completa, partiendo únicamente de una parte de ella.

**Kunihiko Fukushima** empezó a trabajar sobre Redes de Neuronas Artificiales a finales de los 60 [Fukushima, 1969], estudiando modelos espaciales y espacio-temporales para sistemas de visión [Fukushima, 1970] y el cerebro. Su trabajo más notable ha sido la creación de un paradigma de Red Neuronal Artificial multicapa para visión, que ha sido mejorado con el tiempo. Fukushima llamó a su primer trabajo el COGNITRON [Fukushima, 1975, Fukushima, 1979], y realizó una versión mejorada llamada NEOCOGNITRON [Fukushima, 1980].

**A. Harry Klopff** estudió las relaciones entre la psicología de la mente y la biología del cerebro desde 1969 [Klopff and Gose, 1969]. Teorizó [Klopff, 1972, Klopff, 1979, Klopff, 1982] que la neurona es un componente hedonístico del cerebro se mueve por búsqueda de metas. Es un sistema adaptativo que aumenta la eficacia de sus sinapsis excitadoras cuando se despolariza, y aumenta la eficacia de sus sinapsis inhibidoras cuando se hiperpolariza.

#### 1.4.3. La década de los 70 y 80

**Teuvo Kohonen** comenzó sus investigaciones sobre Redes de Neuronas Artificiales con paradigmas de conexiones aleatorias en 1971. Los trabajos de Kohonen [Kohonen, 1972, Kohonen, 1974] se centraron en memorias asociativas y matrices de correlación, de manera semejante a los trabajos de Anderson [Anderson, 1968, Anderson, 1970] y Steinbuch y Piske [Steinbuch, 1961, Steinbuch and Piske, 1963]. Más tarde, Kohonen y Ruohonen extendieron el modelo de memoria asociativa lineal, que requería vectores linealmente independientes para un buen rendimiento, en uno que buscaba las óptimas entre vectores linealmente dependientes, llamado *Asociador Óptimo de Memoria Lineal* (OLAM) [Kohonen, 1977]. Más tarde realizó investigaciones en métodos de aprendizaje y desarrolló el LVQ (Learning Vector Quantization), un sistema de aprendizaje competitivo.

El premio Nobel **Leon Cooper** y su colega **Charles Elbaum** comenzaron a trabajar en Redes de Neuronas Artificiales a principios de los 70 [Cooper, 1973, Cooper et al., 1979]. Formaron un grupo para el desarrollo, la patente y la explotación comercial de Redes de Neuronas Artificiales, llamado Nestor Associates. Este pequeño grupo tuvo bastante éxito en el desarrollo comercial de algunos sistemas de Red Neuronal, como por ejemplo la red RCE (Reduced Coulomb Energy) [Reilly et al., 1982], con su propio sistema de aprendizaje NSL (Nestor Learning System).

**Terence Sejnowski** es uno de los pocos investigadores que trabajaron con modelos matemáticos y biológicos. Una de sus más importantes contribuciones en este campo es el descubrimiento, junto con Geoff Hinton, del algoritmo de la máquina de Boltzmann [Hinton et al., 1984], y su extensión de mayor orden, la primera Red de Neuronas Artificial que reconocía un algoritmo de aprendizaje para una red de tres niveles [Sejnowski and Hinton, 1986]. Aplicaron la máquina de Boltzmann a distintas áreas de visión [Kienker et al., 1986, Sejnowski, 1986]. Más recientemente son conocidas sus contribuciones a la aplicación del algoritmo del RETROPROPAGACIÓN, sobre todo para el reconocimiento de voz.

**McClelland y Rumelhart** son psicólogos interesados en la utilización de

modelos de Redes de Neuronas Artificiales para la ayuda a la comprensión de las funciones psicológicas de la mente. David Rumelhart [Rumelhart, 1977] empezó a interesarse por las Redes de Neuronas a través de sus trabajos del modelo HEARSAY de reconocimiento del lenguaje hablado. Por su parte, James McClelland, inspirado por los trabajos de Anderson [Anderson, 1968], comenzó sus investigaciones formulando un modelo semiparalelo de procesos mentales [McClelland, 1979]. Más tarde [McClelland and Rumelhart, 1981], ambos aunaron sus esfuerzos para elaborar un paradigma de reconocimiento de voz (Interactive Activation Model) [Rumelhart and McClelland, 1982]. A partir de sus estudios, seguidos con mucho interés por otros investigadores, nació el grupo de investigación PDP (Parallel Distributed Processing). La culminación de los estudios de los investigadores del grupo PDP fue la publicación, en 1986, de un libro en dos volúmenes titulado *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Entre los muchos autores de este libro cabe destacar a los siguientes: McClelland, Rumelhart, Hinton, Zipser, Smolensky, Sejnowsky, Williams, Jordan, Stone, Rabin, Elman, Kawamoto, Crick, Asanuma, Munro.

**Jerome Feldman y Dana Ballard**, pertenecientes a la Universidad de Rochester, fueron dos de los primeros investigadores en este campo. Ambos, junto con un grupo de asociados, formaron el grupo conexionista de su Universidad. Sus primeros trabajos fueron sobre visión [Feldman, 1980, Feldman, 1981a, Feldman, 1981b, Ballard, 1981], basándose en modelos de memoria visual. Otras áreas de investigación del grupo son: visión por computador [Ballard, 1982], lenguaje natural [Cottrell and Small, 1984] y representación de conceptos abstractos [Feldman, 1986b, Feldman, 1986a].

**Robert Hecht-Nielsen** fue el principal diseñador de uno de los primeros computadores neuronales, dedicado al procesamiento de paradigmas de Redes Neuronales. Este neuro-computador, el TRW MARK III, está soportado por un computador VAX de DIGITAL, y estuvo comercialmente disponible en 1986.

En 1982, **John Hopfield** describió un método de análisis del estado estable en una red autoasociativa [Hopfield, 1982]. Introdujo una función de energía en sus estudios sobre sistemas de ecuaciones no lineales. Hopfield demuestra que se puede construir una ecuación de energía que describa la actividad de una Red Neuronal monocapa, en tiempo discreto, y que esta ecuación de energía puede ir disipándose y el sistema converger a un valor mínimo local. Este análisis hizo resurgir el interés por aplicar los paradigmas de Redes de Neuronas Artificiales para problemas difíciles que los computadores convencionales no pueden resolver. Hopfield extendió su modelo para considerar tiempos continuos [Hopfield, 1984].

**Bart Kosko** creó una familia de paradigmas de Redes de Neuronas Artificiales [Kosko, 1987a, Kosko, 1987b] (llamados memorias asociativas bidimensionales, BAMs) que extienden a las autoasociativas de Hebb de un nivel, a dos niveles utilizando aprendizaje sin supervisión; y son capaces de converger a una solución mínima dada una matriz arbitraria.

## Capítulo 2

### PRIMEROS MODELOS COMPUTACIONALES

#### 2.1. Células de McCulloch-Pitts

El primer ejemplo que puede considerarse como una red de neuronas artificial, al menos estructuralmente, son las células de McCulloch-Pitts. Este primer modelo de neurona fue propuesto por Warren McCulloch y Walter Pitts en 1943 en el artículo "A Logical Calculus of the Ideas Immanent in Nervous Activity" [McCulloch and Pitts, 1943]. En él modelizaban una estructura y un funcionamiento simplificado de las neuronas del cerebro, considerándolas como dispositivos con sólo dos estados posibles: apagado (0) y encendido (1).

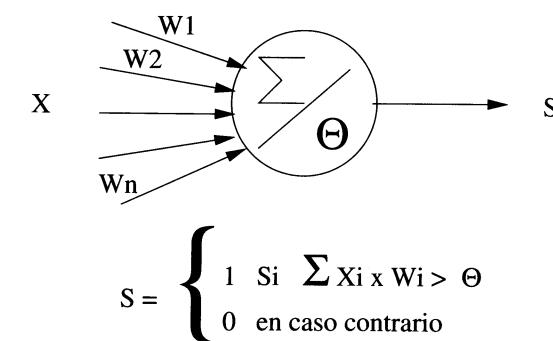


Figura 2.1: Esquema general de una célula de McCulloch-Pitts

La célula de McCulloch-Pitts recibe como entrada un conjunto de  $n$  valores

binarios,  $X = \{x_1, x_2, \dots, x_n\}$  procedente de las salidas de otras células, o de la entrada a la red; y produce una única salida también binaria,  $s$ . Cada célula se caracteriza por  $n + 1$  valores reales, de los cuales  $n$  son los pesos de las conexiones ( $w_i$ ) correspondientes a las entradas  $x_i$ , y el otro es un valor de umbral  $\theta$ , que puede ser distinto para cada célula. La célula opera en lapsos discretos. La forma de procesar la entrada es la siguiente: la célula se activará y, por lo tanto, producirá un valor 1, si la suma de las entradas multiplicadas por los pesos supera al umbral  $\theta$ .

$$S(t+1) = \begin{cases} 1 & \sum_i w_i x_i(t) > \theta \\ 0 & \text{en caso contrario} \end{cases}$$

A partir del modelo de neurona de McCulloch-Pitts se define el primer modelo de red neuronal:

Una red neuronal es una colección de neuronas de McCulloch y Pitts, todas con las mismas escalas de tiempos, donde sus salidas están conectadas a las entradas de otras neuronas.

De este modo, una salida puede actuar sobre varias entradas, pero una entrada viene a lo sumo de una salida. La red tiene contacto con el exterior a través de líneas de entrada y de salida. Las líneas de entrada de la red formarán parte de la entrada de alguna o de todas las neuronas de la red. Asimismo, las líneas de salida procederán de algunas o de todas las neuronas de la red.

Ésta formalización matemática de la red neuronal no es y no pretende ser una modelización del cerebro, pero sí un punto de partida para iniciar los estudios sobre el mismo.

Una red neuronal de células de McCulloch-Pitts tiene la capacidad de computación universal. Es decir, cualquier estructura que pueda ser programada en un computador, puede ser modelizada mediante una red de células de McCulloch-Pitts. Esto se puede intuir mediante las modelizaciones de funciones lógicas. Los computadores están constituidos de elementos de cálculo simples. Si cada uno de estos elementos puede ser modelizado mediante una estructura de células de McCulloch-Pitts, los cálculos realizados por éste podrían ser sustituidos por su correspondiente red de células.

A continuación se van a representar las funciones lógicas AND, OR y NOT, mediante células de McCulloch-Pitts.

### 2.1.1. Función lógica NOT

Es una célula con una entrada y una salida. El único peso que posee tiene valor  $-1$ , y su umbral valor  $-1$  (Figura 2.2).

Si la entrada a la célula es cero, la salida será:  $0x - 1 = 0$ ; como es menor que el umbral, que es  $-1$ , la salida será 1. Para una entrada de uno la salida será:  $1 \cdot x - 1 = -1$ , que al no ser mayor que el umbral  $-1$  producirá una salida de 0. Éste es el comportamiento de una función lógica NOT.

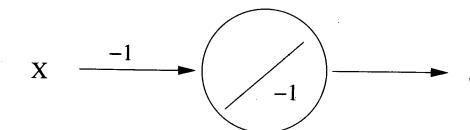


Figura 2.2: Función lógica NOT mediante una célula de McCulloch-Pitts

### 2.1.2. Función lógica AND

Para la función AND, la célula tendrá dos entradas y una salida. El valor del umbral es 1, y el de las dos conexiones también 1 (Figura 2.3).

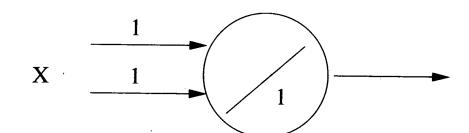


Figura 2.3: Función lógica AND mediante una célula de McCulloch-Pitts

En cuanto a la función lógica AND, se va incluir la tabla de la función para las distintas entradas:

$x_1$	$x_2$	$S$
0	0	0
0	1	0
1	0	0
1	1	1

La siguiente tabla muestra la salida de la célula para cada una de las entradas:

$x_1$	$x_2$	$\Sigma x_i w_i$	$S$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

La única vez que el sumatorio supera el umbral de 1 es en la última entrada, y será sólo en este caso cuando la célula dé una salida de 1. Se aprecia cómo ambas tablas coinciden.

### 2.1.3. Función lógica OR

La célula que representa la función OR es igual que la de la función lógica AND, cambiando el valor del umbral, que en este caso será cero (Figura 2.4).

La tabla de la función OR es:

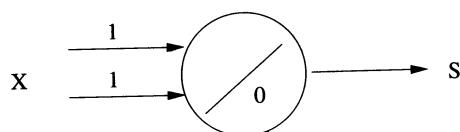


Figura 2.4: Función lógica OR mediante una célula de McCulloch-Pitts

$x_1$	$x_2$	$S$
0	0	0
0	1	1
1	0	1
1	1	1

Mientras que la tabla de la célula de McCulloch-Pitts anterior es:

$x_1$	$x_2$	$\Sigma x_i w_i$	$S$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

En este caso el sumatorio supera al umbral 0 en todos los casos menos en el primero, que será en el único en el que la salida da cero.

Aparentemente, el diseño de células de McCulloch-Pitts es sencillo; sin embargo, cuando se trata de funciones más complejas, o de sistemas de computación sofisticados, el número de células necesarias será muy elevado y los parámetros involucrados (conexiones entre células, umbrales, pesos) tan numerosos que la realización de células de McCulloch-Pitts para estos problemas es imposible en la práctica. Sería necesario un mecanismo de asignación de los parámetros de forma automática. Es lo que se conoce como mecanismo de aprendizaje.

## 2.2. Perceptron

Como se ha señalado, los primeros estudios sobre Redes de Neuronas Artificiales datan de los años 50, sobre todo con la aparición del modelo PERCEPTRON. Este modelo se concibió como un sistema capaz de realizar tareas de clasificación de forma automática. La idea era disponer de un sistema que, a partir de un conjunto de ejemplos de clases diferentes, fuera capaz de determinar las ecuaciones de las superficies que hacían de frontera de dichas clases. La información sobre la que se basaba el sistema estaba constituida por los ejemplos existentes de las diferentes clases. A esto lo llamaremos a partir de ahora patrones o ejemplos de entrenamiento, indistintamente. Son dichos patrones de entrenamiento los que aportaban la información necesaria para que el sistema construyera las superficies discriminantes, y además actuara como un discriminador para ejemplos

nuevos desconocidos. El sistema, al final del proceso, era capaz de determinar, para cualquier ejemplo nuevo, a qué clase pertenecía.

### 2.2.1. Descripción del modelo

La arquitectura de la red es muy simple. Se trata de una estructura monocapa, en la que hay un conjunto de células de entrada, tantas como sea necesario, según los términos del problema; y una o varias células de salida. Cada una de las células de entrada tiene conexiones con todas las células de salida, y son estas conexiones las que determinan las superficies de discriminación del sistema.

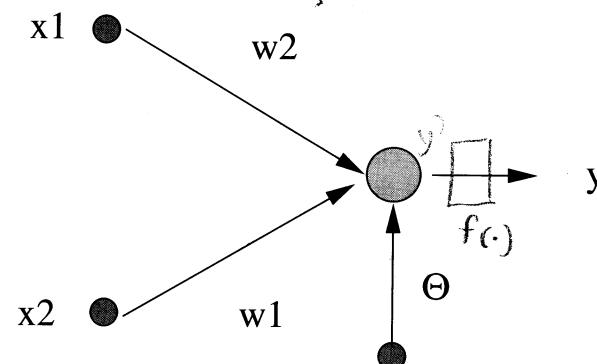


Figura 2.5: Arquitectura de un PERCEPTRON con dos entradas y una salida

En el ejemplo de la Figura 2.5 las entradas son  $x_1$  y  $x_2$ , y la salida,  $y$ . Los pesos son  $w_1$  y  $w_2$ . Además, existe un parámetro adicional llamado umbral y denotado por  $\theta$ . El umbral se utiliza como factor de comparación para producir la salida, y habrá tantos como células de salida existan en la red, uno por cada una.

En este esquema la salida de la red se obtiene de la siguiente forma. Primero se calcula la activación de la célula de salida mediante la suma ponderada por los pesos de todas las entradas:

$$y' = \sum_{i=1}^n w_i x_i$$

La salida definitiva se produce al aplicarle una función de salida al nivel de activación de la célula. En un PERCEPTRON la función de salida es una función escalón que depende del umbral:

$$y = \mathcal{F}(y', \theta)$$

$$\mathcal{F}(s, \theta) = \begin{cases} 1 & \text{si } s > \theta \\ -1 & \text{en caso contrario} \end{cases}$$

Simplemente pasando el término  $\theta$  al otro lado de la ecuación, la salida se puede escribir en una sola ecuación:

$$y = \mathcal{F} \left( \sum_{i=1}^n w_i x_i + \theta \right) \quad (2.1)$$

donde  $\mathcal{F}$  ya no depende de ningún parámetro:

$$\mathcal{F}(s) = \begin{cases} 1 & \text{si } s > 0 \\ -1 & \text{en caso contrario} \end{cases} \quad (2.2)$$

Esta ecuación equivale a introducir artificialmente en la salida un nuevo peso  $\theta$  que no está conectado a ninguna entrada, sino a una ficticia con un valor constante de  $-1$ .

La función de salida  $\mathcal{F}$  es binaria y de gran utilidad en este modelo ya que al tratarse de un discriminante de clases, una salida binaria puede ser fácilmente traducible a una clasificación en dos categorías de la siguiente forma:

- Si la red produce salida 1, la entrada pertenece a la categoría A.
- Si la red produce salida  $-1$ , la entrada pertenece a la categoría B.

En el caso de dos dimensiones, la Ecuación 2.1 se transforma en:

$$w_1 x_1 + w_2 x_2 + \theta = 0$$

que es la ecuación de una recta de pendiente  $-\frac{w_1}{w_2}$  y que en el origen de ordenadas pasa por  $-\frac{\theta}{w_2}$ . En este caso, con dos células en la entrada, los patrones de entrenamiento pueden representarse como puntos en un espacio bidimensional. Si además dichos puntos pertenecen a una de dos categorías, A o B, la separación de dichas categorías podrá hacerse mediante la recta anterior.

Es decir, la red define una recta, que en el caso de ser solución al problema discriminará entre las clases existentes en los datos de entrenamiento. Gráficamente podría representarse como se muestra en la Figura 2.6.

Si las clases están separadas, puede demostrarse que dicha recta discriminante existe; el problema es cómo determinar la ecuación de tal recta, a partir de los datos de entrenamiento. Esto es lo que hace precisamente el PERCEPTRON en su proceso de aprendizaje. El problema puede complicarse si en vez de dos dimensiones hay muchas más. En este caso no habrá que determinar rectas, sino hiperplanos. El proceso puede definirse en los siguientes términos:

Dado Se dispone de un conjunto de ejemplos de entrenamiento, distribuidos en un espacio multidimensional, y de los que se conoce a qué categoría pertenecen. Se va a describir el caso en que hay únicamente dos clases, A y B. Más adelante se comentarán las modificaciones del modelo para extenderlo a un caso genérico de  $n$  clases.

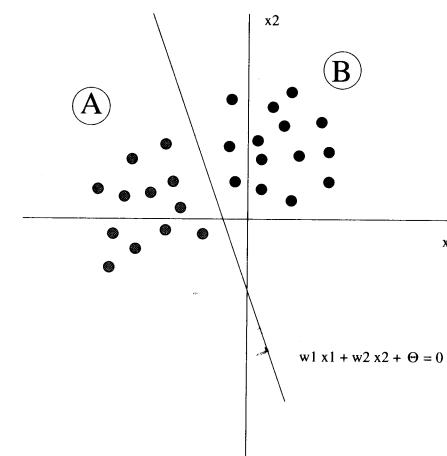


Figura 2.6: Separación en dos clases mediante un PERCEPTRON

Obtener Hay que determinar la ecuación del hiperplano que deja a un lado los ejemplos de un tipo y a otro los del otro. La ecuación del hiperplano se deduce a partir de los ejemplos.

Más formalmente sería: dados conjuntos de puntos en  $\mathbb{R}^n$ :  $A = (\vec{a}_1, \dots, \vec{a}_{na})$  y  $B = (\vec{b}_1, \dots, \vec{b}_{nb})$ , obtener el hiperplano:  $w_1 x_1 + \dots + w_n x_n + \theta = 0$ , de tal forma que:

$$\forall \vec{a} \in A : w_1 a_1 + \dots + w_n a_n + \theta > 0$$

y

$$\forall \vec{b} \in B : w_1 b_1 + \dots + w_n b_n + \theta < 0$$

Este proceso constituye el aprendizaje del PERCEPTRON y se realiza mediante un proceso iterativo en el que paulatinamente se van modificando los valores de los pesos de las conexiones, hasta encontrar los valores que determinan las ecuaciones discriminantes. En función de la red neuronal esto sería equivalente a encontrar los valores de las conexiones que hagan:

$$\forall \vec{a} \in A : y(\vec{a}) = 1$$

y

$$\forall \vec{b} \in B : y(\vec{b}) = -1$$

El proceso de aprendizaje consiste en lo siguiente. Se introduce un patrón de los del conjunto de **aprendizaje**, perteneciente, por ejemplo, a la clase A. Se

obtiene la salida que genera la red para dicho patrón. Si la salida producida es 1, la respuesta de la red para dicho patrón es correcta, y no se realizará ninguna acción; se procederá simplemente a introducir un nuevo patrón. Si por el contrario la salida producida es  $-1$ , entonces la respuesta de la red es incorrecta; la red categoriza el patrón como de la clase B. Esto es un error de clasificación, y es en este caso cuando se produce el aprendizaje. El aprendizaje consiste en modificar los valores de las conexiones. En este caso, dado que la salida producida es inferior a la que se debería haber obtenido, los pesos son incrementados para que en la próxima presentación del mismo patrón pueda superar el umbral y producir la salida deseada, 1. Si el patrón que se introduce es de la clase B, y también se produce un error de clasificación, el proceso se invierte; los pesos se decrementan por la misma razón.

\* Si se llama  $\chi$  a cada uno de los patrones de entrenamiento y  $d(\chi)$  a su clase asociada, tomando valores en  $(1, -1)$ , el proceso de aprendizaje se puede describir de la siguiente forma:

1. Empezar con valores aleatorios para los pesos y el umbral.
2. Seleccionar un vector de entrada  $\chi$  del conjunto de ejemplos de entrenamiento.
3. Si  $y \neq d(\chi)$ , la red da una respuesta incorrecta. Modificar  $w_i$  de acuerdo con:

$$\Delta w_i = d(\chi)x_i \quad (2.3)$$

4. Si no se ha cumplido el criterio de finalización, volver a 2.

En el paso tres se aprecia que si la salida de la red para un patrón es  $y(\chi) = 1$ , pero su clase es  $d(\chi) = -1$ , entonces el incremento es negativo,  $\Delta w_i = d(\chi)x_i = -x_i$ , mientras que si ocurre lo contrario, es positivo, como se describió anteriormente.

Puesto que el umbral es equivalente a un peso adicional, al que se denota por  $w_0$ , cuya entrada es siempre 1 ( $x_0 = 1$ ), la ecuación anterior se puede extender para el umbral de la siguiente manera:

$$\Delta w_i = d(\chi)x_i, \quad i = 0, \dots, n$$

### 2.2.2. Función lógica AND

El proceso de aprendizaje se ilustrará con un ejemplo. Sea el problema de aprender una función lógica AND<sup>1</sup>, a partir de los ejemplos que la definen:

<sup>1</sup>Es el mismo problema que se mencionó anteriormente para las células de McCulloch-Pitts, pero en este caso es la red la que tiene que aprender los valores precisos de los pesos

$x_1$	$x_2$	AND
-1	-1	-1 (A)
+1	-1	-1 (A)
-1	+1	-1 (A)
+1	+1	+1 (B)

En este caso hay cuatro ejemplos; tres de ellos pertenecen a la clase A (salida  $-1$ ) y el restante a la clase B (salida 1). Gráficamente se representa mediante la Figura 2.7.

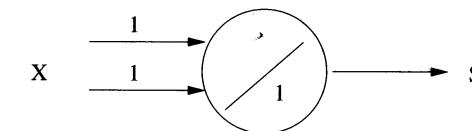


Figura 2.7: Representación de la función lógica AND

La inicialización de la red es aleatoria. Supóngase que los pesos son inicializados a 1, y el umbral a 0,5 (Figura 2.8).

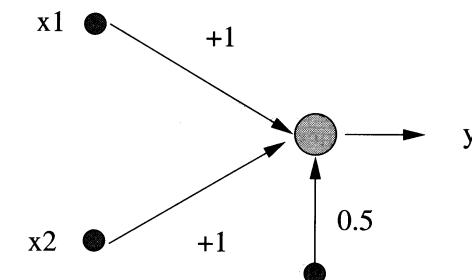


Figura 2.8: Inicialización del PERCEPTRON para el ejemplo de la función AND

Con esta inicialización, la recta discriminante definida por la red es  $x_1 + x_2 + 0,5 = 0$ , cuya representación gráfica es la que aparece en la Figura 2.9. En este caso se ve cómo la recta clasifica bien un patrón de la clase A, el de la parte inferior izquierda, y el patrón de la clase B. Luego habrá que proceder a realizar aprendizaje.

A continuación se introducen los patrones de entrenamiento y se efectúa el aprendizaje para aquellos que produzcan una salida errónea.

Patrón	Salida	Clasifica
$(-1, -1 -1)$	$f(-1 + (-1) + 0,5) = f(-1,5) = -1$	Bien $[-1 = -1]$
$(+1, -1 -1)$	$f(1 + (-1) + 0,5) = f(0,5) = +1$	Mal $[+1 \neq -1]$

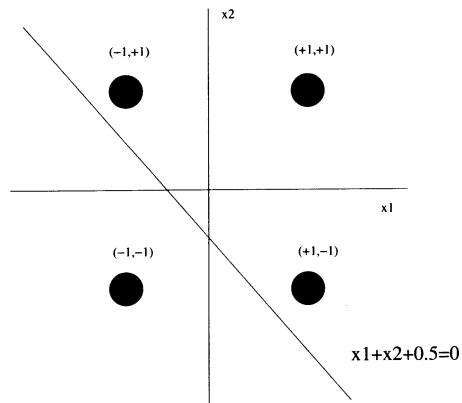


Figura 2.9: Recta determinada por el PERCEPTRON después de la inicialización

El primer patrón produce una salida correcta  $y(-1, -1) = -1 = d(-1, -1)$ , el segundo produce un error  $y(1, -1) = 1 \neq d(1, -1) = -1$ . En este caso se produce el aprendizaje tal y como se refleja en la tabla:

Aprendizaje:		
Antes	Incremento	Después
$w_1 = 1$	$d(\chi) \cdot x_1 \equiv -1 \cdot 1 = -1$	$w_1 = 0$
$w_2 = 1$	$d(\chi) \cdot x_2 \equiv -1 \cdot -1 = 1$	$w_2 = 2$
$\theta = 0,5$	$d(\chi) \equiv -1$	$w_1 = -0,5$

Después del aprendizaje los pesos son  $W = \{-0,5, 0, 2\}$ , que se corresponden con la recta:  $2x_2 - 0,5 = 0$  de la Figura 2.10.

Se puede apreciar cómo la recta se ha desplazado y ahora el patrón de abajo a la izquierda también está bien clasificado. El único error de clasificación que existe es en el patrón de arriba a la izquierda.

Se introduce ahora el siguiente patrón de entrenamiento:

Patrón	Salida	Clasifica
$(-1, +1  -1)$	$f(0 + 2 - 0,5) = f(-1,5) = +1$	Mal $[+1 \neq -1]$

Este patrón produce una salida errónea y se procede al aprendizaje como en el caso anterior:

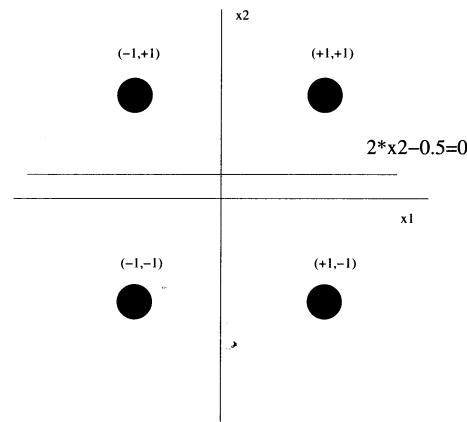


Figura 2.10: Recta determinada por el PERCEPTRON después de la primera iteración

Aprendizaje:		
Antes	Incremento	Después
$w_1 = 0$	$d(\chi) \cdot x_1 \equiv -1 \cdot -1 = 1$	$w_1 = 1$
$w_2 = 2$	$d(\chi) \cdot x_2 \equiv -1 \cdot 1 = -1$	$w_2 = 1$
$\theta = -0,5$	$d(\chi) \equiv -1$	$w_0 = -1,5$

Después de éste ciclo de aprendizaje el vector de pesos queda así:  $W = \{-1,5, 1, 1\}$ , que se corresponde con la recta:  $x_1 + x_2 - 1,5 = 0$  de la Figura 2.11.

Vemos que esta recta es un discriminador perfecto de los ejemplos, y que después de tan sólo dos ciclos de aprendizaje la red ha sido capaz de resolver este sencillo problema de clasificación. Sin embargo, esto sólo se demuestra si al introducir todos los ejemplos de entrenamiento la red los clasifica correctamente:

Patrón	Salida	Clasifica
$(+1, +1  +1)$	$f(1 + 1 - 1,5) = f(0,5) = +1$	Bien $[+1 = +1]$
	Iteración 2	
Patrón	Salida	Clasifica
$(-1, -1  -1)$	$f(-1 + (-1) - 1,5) = f(-3,5) = -1$	Bien $[-1 = -1]$
$(+1, -1  -1)$	$f(1 + (-1) - 1,5) = f(-1,5) = -1$	Bien $[-1 = -1]$
$(-1, +1  -1)$	$f(-1 + 1 - 1,5) = f(-1,5) = -1$	Bien $[-1 = -1]$
$(+1, +1  +1)$	$f(1 + 1 - 1,5) = f(0,5) = +1$	Bien $[+1 = +1]$

Se aprecia que éste es el caso. En general, en un PERCEPTRON, los patrones deben ser introducidos una y otra vez hasta que exista un ciclo en el que se hayan introducido todos los patrones y hayan sido correctamente clasificados.

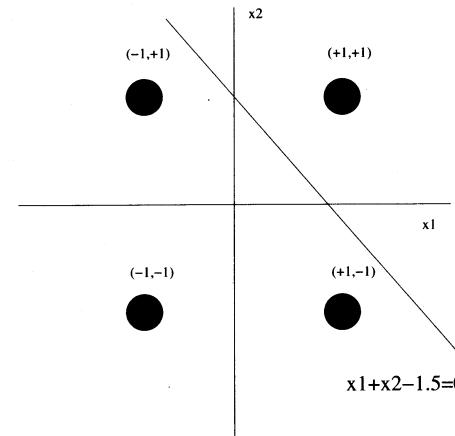


Figura 2.11: Recta determinada por el PERCEPTRON después de la segunda iteración

El modelo del PERCEPTRON descrito hasta el momento maneja entradas binarias y clasifica únicamente en dos categorías. La extensión a entradas no binarias es inmediata. En la regla de aprendizaje el valor de cada conexión se incrementa o decrementa.

### 2.3. Adaline

El PERCEPTRON es un sistema de aprendizaje basado en ejemplos capaz de realizar tareas de clasificación. Sin embargo, existen un gran número de problemas abordables desde la perspectiva del aprendizaje basado en ejemplos que no se reducen a tareas de clasificación. La característica de clasificador del PERCEPTRON viene dada por la naturaleza de sus salidas. En la capa de salida las células codifican una función de salida en escalón, función 2.2, que transforma la suma ponderada de las entradas que es un valor real, en una salida binaria. Al ser binarias, sólo pueden codificar un conjunto discreto de estados. Si las salidas fueran números reales, podrían codificar cualquier tipo de salida y se convertirían en sistemas de resolución de problemas generales. En este caso, podrían resolver problemas a partir de ejemplos en los que es necesario aproximar una función cualquiera ( $\mathcal{F}(x)$ ), definida por un conjunto de datos de entrada.

Los datos de entrenamiento en este caso son conjuntos de valores enteros, compuestos por un vector de entrada y su salida asociada:

$$\mathcal{P} = \{(\vec{x}_1, y_1), \dots, (\vec{x}_m, y_m)\}$$

En este caso la función a aproximar sería:

$$\mathcal{F}(\vec{x}_i) = y_i, \forall p \in \mathcal{P}$$

En otros términos, habrá que buscar la función  $\mathcal{F}(x)$  tal que aplicada a cada una de las entradas  $x_i$  del conjunto de aprendizaje  $\mathcal{P}$  produzca la salida  $y_i$  correspondiente en dicho conjunto de aprendizaje.

Sin embargo, la naturaleza de la regla de aprendizaje del PERCEPTRON no permite producir salidas reales. Esta regla es fundamentalmente una regla de aprendizaje por refuerzo en la que se potencian las salidas correctas y no se tienen en cuenta las incorrectas. No existe ninguna graduación en la regla que indique en qué medida resulta errónea la salida producida, y refuerce proporcionalmente a dicha medida de error.

#### 2.3.1. Descripción del modelo

En el año 1960, Widrow y Hoff [Widrow, 1960] propusieron un sistema de aprendizaje que sí tuviera en cuenta el error producido, y diseñaron lo que denominaron ADaptive LInear NEuron, o en su acrónimo, ADALINE. Ésta es una estructura prácticamente idéntica a la del PERCEPTRON, pero es un mecanismo físico, capaz de realizar aprendizaje. Es un elemento combinador adaptativo, que recibe un conjunto de entradas y las combina para producir una salida. Esta salida puede transformarse en binaria mediante un conmutador bipolar que produce un 1 si la salida es positiva y un -1 si es negativa:

$$\hat{y} = \sum_{i=1}^n w_i \cdot x_i + \theta$$

El aprendizaje en este caso incluye la diferencia entre el valor real producido en la capa de salida para un patrón de entrada  $x^p$  y el que debería haber producido dicho patrón, es decir, su salida esperada  $d^p$ , que está en el conjunto de aprendizaje ( $|d^p - y^p|$ ). A esta regla de aprendizaje se la conoce con el nombre de *regla Delta*.

La diferencia con respecto a la regla de aprendizaje del PERCEPTRON es la manera de utilizar la salida, una diferencia fundamental entre ambos sistemas. El PERCEPTRON utiliza la salida de la función umbral para el aprendizaje; sin embargo, la regla Delta utiliza directamente la salida de la red, sin pasarlala por ninguna función umbral. El objetivo es obtener un valor determinado  $y = d^p$  cuando el conjunto de valores  $X_i^p, i = 1, \dots, n$  se introduce a la entrada. Por lo tanto, el problema será obtener los valores de  $w_i, i = 1, \dots, n$  que permitan realizar lo anterior para un número arbitrario de patrones de entrada.

No es posible conseguir una salida exacta, pero sí minimizar la desviación cometida por la red, esto es, minimizar el error cometido por la red para la totalidad del conjunto de patrones de ejemplo. En otros términos, hay que evaluar globalmente el error cometido por la red para todos los patrones, o sea elegir una medida de error global. Habitualmente, la medida de error global utilizada es el

error cuadrático medio, Ecuación 2.4, pero otros errores pueden ser utilizados en el modelo.

$$E = \sum_{p=1}^m E^p = \frac{1}{2} \sum_{p=1}^m (d^p - y^p)^2 \quad (2.4)$$

Al ser ésta una medida de error global, la regla intentará minimizar este valor para todos los elementos del conjunto de patrones de aprendizaje. La manera de minimizar este error es recurrir a un proceso iterativo en el que se van presentando los patrones uno a uno, y modificando los parámetros de la red, pesos de las conexiones, mediante la *regla del descenso del gradiente*.

La idea es realizar un cambio en cada peso proporcional a la derivada del error, medida en el patrón actual, respecto del peso:

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j}$$

Para el cálculo de la derivada anterior se utiliza la regla de la cadena:

$$\frac{\partial A}{\partial x} = \frac{\partial A}{\partial y} \frac{\partial y}{\partial x}$$

Aplicando la regla de la cadena a la expresión anterior queda como sigue:

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p} \frac{\partial y^p}{\partial w_j}$$

Al ser unidades lineales, sin función de activación en la capa de salida, se cumple:

$$\frac{\partial y^p}{\partial w_j} = x_j \quad \frac{\partial E^p}{\partial y^p} = -(d^p - y^p)$$

que sustituyendo queda como sigue:

$$\Delta_p w_j = \gamma(d^p - y^p)x_j \quad (2.5)$$

Si se compara la expresión de la regla Delta (Ecuación 2.5), con la regla de aprendizaje del PERCEPTRON (Ecuación 2.3), se aprecia que la diferencia es precisamente la introducción de la diferencia entre la salida deseada y la obtenida en la regla de aprendizaje. Si la salida del ADALINE fuese binaria, el conjunto de patrones estaría constituido por  $\mathcal{P} = \{(x_1^p, 0 \text{ o } 1), \dots, (x_m^p, 0 \text{ o } 1)\}$ , es decir,  $d^p \in \{0, 1\}, \forall p$ . Si se incluye a la salida del ADALINE el acoplador bipolar comentado con anterioridad para "binarizar" la salida, la regla Delta (Ecuación 2.5) quedaría así:

$$\Delta_p w_j = \begin{cases} \gamma x_j & \text{si } d^p > y^p \\ -\gamma x_j & \text{si } d^p < y^p \\ 0 & \text{en caso contrario} \end{cases} \quad (2.6)$$

que para un  $\gamma = 1$  se convierte en la regla del PERCEPTRON. Así pues, la regla Delta es una extensión de la regla del PERCEPTRON a valores de salida reales.

El procedimiento de aprendizaje definido por la regla Delta será:

1. Inicializar los pesos de forma aleatoria.
2. Introducir un patrón de entrada.
3. Calcular la salida de la red, compararla con la deseada y obtener la diferencia:  $(d^p - y^p)$ .
4. Para todos los pesos, multiplicar dicha diferencia por la entrada correspondiente, y ponderarla por una tasa de aprendizaje  $\gamma$ .
5. Modificar el peso restando del valor antiguo la cantidad obtenida en 4.
6. Si no se ha cumplido el criterio de convergencia, regresar a 2; si se han acabado todos los patrones, empezar de nuevo a introducir patrones.

En resumen, las diferencias entre ambos modelos de Redes de Neuronas Artificiales serán las siguientes:

- En el PERCEPTRON la salida es binaria, en el ADALINE es real.
- En el PERCEPTRON la diferencia entre entrada y salida es 0 si ambas pertenecen a la misma categoría y  $\pm 1$  si por el contrario pertenece a categorías diferentes. En el ADALINE se calcula la diferencia real entre entrada y salida.
- En el ADALINE existe una medida de cuánto se ha equivocado la red; en el PERCEPTRON sólo se determina si se ha equivocado o no.
- En el ADALINE hay una razón de aprendizaje ( $\gamma$ ) para regular cuánto va a afectar cada equivocación a la modificación de los pesos. Es siempre un valor entre 0 y 1 para ponderar el aprendizaje.

### 2.3.2. Descodificador de binario a decimal

Como ejemplo para ilustrar el funcionamiento del ADALINE se ha elegido un descodificador de binario a decimal. El descodificador recibe una entrada en binario y produce como salida su valor en decimal. Evidentemente, éste es un problema trivial, ya que existe una expresión analítica capaz de realizar la descodificación:

$$d = \sum_{i=1}^n x_i^i \quad (2.7)$$

En el ejemplo se trata de comprobar si una red de tipo ADALINE es capaz de aproximar la expresión de la Ecuación 2.7 por sí sola, a partir del conjunto de ejemplos de entrenamiento.

En primer lugar habrá que producir dichos ejemplos. Se determina un tamaño de la entrada; por razones prácticas se ha elegido una dimensión de 3 en la entrada. Como se trata de pasar de binario a decimal, las entradas estarán en binario, y como el tamaño prefijado es tres, el número máximo de entradas que se pueden generar es de  $2^3 = 8$ . En la siguiente tabla se muestra el conjunto de datos de entrenamiento:

0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

A continuación se va a proceder con el algoritmo para la red ADALINE. Primero se realiza la inicialización de los pesos de forma aleatoria y se eligen los parámetros de la red. En este caso el único parámetro que hay que elegir es la razón de aprendizaje  $\alpha$  a la que se le ha asignado un valor de 0,3. Después de la inicialización los pesos de la red tienen los siguientes valores:

$w_{11}$	$w_{21}$	$w_{31}$
0,84	0,39	0,78

Se introduce el primer patrón en la red y se calcula su salida:

$\vec{x}$	0	0	1	d = 1
$\vec{w}_1$	0,84	0,394	0,783	
$\vec{x} \cdot \vec{w}_1$	0,0	0,0	0,783	y = 0,783

La salida producida no es idéntica a la deseada ( $e = 1 - 0,783 = 0,217$ ) y, por lo tanto, se producirá un ciclo de aprendizaje, utilizando la Ecuación 2.5. En la tabla siguiente se reflejan los valores de los pesos antes del aprendizaje, el incremento que se producirá según la fórmula de la regla Delta, y los valores después de aplicar dicho incremento:

Antes	$w_{11}$	$w_{21}$	$w_{31}$
	0,840	0,394	0,783
$\Delta w_j = \alpha(d - y)x_j$	$0,3 \cdot 0,217 \cdot 0 = 0$	$0,3 \cdot 0,217 \cdot 0 = 0$	$0,3 \cdot 0,217 \cdot 1 = 0,065$
Después	0,840	0,394	0,848

Este proceso habrá de repetirse para todos los elementos del conjunto de patrones de entrada. A continuación se muestra el proceso para los patrones restantes:

Patrón número 2			
$\vec{x}$	0	1	0
$\vec{w}_1$	0,84	0,394	0,848
$\vec{x} \cdot \vec{w}_1$	0,0	0,394	0,0
			y = 0,394

Aprendizaje para patrón número 2			
Antes	$w_{11}$	$w_{21}$	$w_{31}$
	0,840	0,394	0,848
$\Delta w_j = \alpha(d - y)x_j$	$0,3 \cdot 1,61 \cdot 0 = 0$	$0,3 \cdot 1,61 \cdot 1 = 0,482$	$0,3 \cdot 1,61 \cdot 0 = 0$
Después	0,840	0,876	0,848

El proceso continúa de la misma forma hasta el patrón número 7:

Patrón número 7			
$\vec{x}$	1	1	1
$\vec{w}_1$	3,090	1,966	1,825
$\vec{x} \cdot \vec{w}_1$	3,090	1,966	1,825
			y = 6,881

Aprendizaje para patrón número 7			
Antes	$w_{11}$	$w_{21}$	$w_{31}$
	3,090	1,966	1,825
$\Delta w_j = \alpha(d - y)x_j$	$0,3 \cdot 0,12 \cdot 1 = 0,036$	$0,3 \cdot 0,12 \cdot 1 = 0,036$	$0,3 \cdot 0,12 \cdot 1 = 0,036$
Después	3,126	2,002	1,861

Se puede apreciar cómo después de una única iteración la red produce una salida muy adecuada para este último patrón. Sin embargo, será necesario introducir todos los patrones de nuevo y repetir el proceso un número indefinido de veces si se desea reducir el error. A continuación se reflejarán los valores de los pesos y salidas producidas para 10 iteraciones de la red, al final de las cuales la salida producida es prácticamente exacta, y se puede afirmar que la red ha sido capaz de aprender el problema de decodificación:

Iteración	Pesos	Error por patrón
1	[3,12,2,00,1,86]	[0,21,1,65,1,27,3,16,1,98,2,35,0,12]
2	[3,61,1,98,1,42]	[-0,86,0,00,-0,60,0,87,0,19,0,73,-0,18]
3	[3,82,1,98,1,20]	[-0,42,0,01,-0,28,0,39,0,06,0,35,-0,01]
4	[3,92,1,98,1,10]	[-0,20,0,01,-0,13,0,17,0,02,0,17,-0,04]
5	[3,96,1,99,1,05]	[-0,09,0,01,-0,06,0,08,0,00,0,08,-0,02]
6	[3,98,1,99,1,02]	[-0,05,0,01,-0,03,0,03,0,00,0,04,-0,01]
7	[3,99,2,00,1,01]	[-0,02,0,00,-0,01,0,02,0,00,0,02,0,00]
8	[4,00,2,00,1,00]	[-0,01,0,00,0,00,0,01,0,00,0,01,0,00]
9	[4,00,2,00,1,00]	[0,00,0,00,0,00,0,00,0,00,0,00,0,00]
10	[4,00,2,00,1,00]	[0,00,0,00,0,00,0,00,0,00,0,00,0,00]

En este ejemplo es fácil determinar los valores de los pesos, dadas unas entradas binarias codificadas como [0,1] (las entradas también podrían codificarse en binario mediante [-1,1]), que realizan la operación de decodificación. Estos

valores para un tamaño de red de tres son 4, 2 y 1. En la tabla anterior se aprecia cómo la red va convergiendo paulatinamente a estos valores óptimos de pesos, y los alcanza en tan sólo 8 iteraciones. Además, también se puede apreciar cómo los errores cometidos en cada patrón van reduciéndose de iteración en iteración, hasta alcanzar un valor de cero para todos los patrones, en la iteración 9, que es la siguiente a cuando se han encontrado los valores óptimos de los pesos. Para algunos patrones la convergencia es más rápida; el segundo patrón produce valores de error muy próximos a cero ya desde la segunda iteración, mientras que otros tardan más en ser aprendidos. Esto dependerá en gran medida de la inicialización realizada, y del parámetro de aprendizaje  $\alpha$ . Valores grandes de este parámetro harán descender más rápidamente el error al principio, pero el ajuste definitivo de los pesos será más lento, y viceversa. En algunos casos se opta por cambiar el parámetro de aprendizaje a medida que se van procesando los patrones de aprendizaje. Se comienza con valores elevados de  $\alpha$ , ya que al principio la red estará muy alejada de la solución y se necesitará que se aproxime "a grandes pasos". A medida que iteran todos los patrones de aprendizaje, se va reduciendo el valor de  $\alpha$ , ya que al final del aprendizaje la red estará próxima a la solución y los ajustes han de ser más finos.

### 2.3.3. Problema del OR exclusivo

El ejemplo del decodificador binario decimal es un ejemplo del tipo de problemas que estos modelos supervisados son capaces de resolver. Tras su aparición en los años 50, en la comunidad científica se empezó a pensar en la posibilidad de que las Redes de Neuronas Artificiales fuesen verdaderos sistemas de resolución general de problemas. Se abría la posibilidad de aproximar soluciones a cualquier problema que estuviera bien descrito por un conjunto de ejemplos. Sin embargo, poco tiempo después se demostraron las serias limitaciones de los modelos existentes. El ejemplo más característico del tipo de problemas sencillos que los modelos descritos no resuelven es el de la función lógica OR exclusivo.

El problema del OR exclusivo se describe por su conjunto de ejemplos:

-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

Produce salida correcta o 1 cuando sus dos entradas coinciden, e incorrecta o -1 cuando no coinciden.

Si se introducen los ejemplos anteriores en una red ADALINE, la salida nunca llega a estabilizarse. A continuación se van a introducir los ejemplos del OR exclusivo en la misma red ADALINE que resolvió el problema de la decodificación. La inicialización en este caso es:

$w_{11}$	$w_{21}$
0,84	0,39

Se introduce el primer patrón, y se efectúan las siguientes operaciones para producir la salida y realizar el aprendizaje:

Patrón número 1			
$\vec{x}$	-1	-1	$d = 1$
$\vec{w}_1$	0,84	0,39	
$\vec{x} \times \vec{w}_1$	-0,84	-0,39	$y = -1,23$

Aprendizaje para patrón número 1			
	$w_{11}$	$w_{21}$	
Antes	0,84	0,39	
$\Delta w_j = 0,3(d - y)x_j$	$0,3 \cdot -2,23 \cdot -1$	$0,3 \cdot +2,23 \cdot -1$	
Después	0,17	-0,28	

Se han decrementado los dos pesos para tratar de aproximar las entradas. Lo mismo se cumpliría para los demás patrones:

Patrón número 2			
$\vec{x}$	-1	1	$d = -1$
$\vec{w}_1$	0,17	-0,28	
$\vec{x} \times \vec{w}_1$	-0,17	-0,28	$y = -0,45$

Aprendizaje para patrón número 2			
	$w_{11}$	$w_{21}$	
Antes	0,17	-0,28	
$\Delta w_j = 0,3(d - y)x_j$	$0,3 \cdot -0,55 \cdot -1$	$0,3 \cdot -0,55 \cdot 1$	
Después	0,34	-0,44	

Patrón número 3			
$\vec{x}$	1	-1	$d = -1$
$\vec{w}_1$	0,34	-0,44	
$\vec{x} \times \vec{w}_1$	0,34	0,44	$y = 0,78$

Aprendizaje para patrón número 3			
	$w_{11}$	$w_{21}$	
Antes	0,34	-0,44	
$\Delta w_j = 0,3(d - y)x_j$	$0,3 \cdot -1,78 \cdot 1$	$0,3 \cdot -1,78 \cdot -1$	
Después	-0,20	0,09	

Patrón número 4			
$\vec{x}$	1	1	$d = -1$
$\vec{w}_1$	-0,20	0,09	
$\vec{x} \times \vec{w}_1$	-0,20	0,09	$y = -0,11$

Aprendizaje para patrón número 4			
	$w_{11}$	$w_{21}$	
Antes	-0,20	0,09	
$\Delta w_j = 0,3(d - y)x_j$	$0,3 \cdot -1,11 \cdot 1$	$0,3 \cdot -1,11 \cdot -1$	
Después	0,13	0,42	

Se aprecia en este caso que los pesos no tienen tendencia a estabilizarse en un determinado valor como en el ejemplo anterior, sino que oscilan, con valores a veces positivos y a veces negativos. Para poder comprobar esta oscilación de una manera más clara, en la tabla siguiente se refleja la evolución en varias iteraciones de los pesos y los errores producidos por cada uno de los cuatro patrones de entrenamiento:

Iteración	Pesos	Error por patrón
1	[0,13,0,42]	[ 2,23,-0,55,-1,78,-1,11]
2	[0,02,0,43]	[ 1,56,-1,29,-1,48, 1,38]
3	[0,00,0,43]	[ 1,45,-1,41,-1,43, 1,42]
4	[0,00,0,43]	[ 1,43,-1,42,-1,43, 1,43]
5	[0,00,0,43]	[ 1,43,-1,43,-1,43, 1,43]
6	[0,00,0,43]	[ 1,43,-1,43,-1,43, 1,43]
7	[0,00,0,43]	[ 1,43,-1,43,-1,43, 1,43]

Se puede ver cómo a partir de la iteración 5 la red estabiliza sus pesos y su error; por más que se continúe iterando, no disminuye. Ningún patrón consigue producir errores pequeños. La red ha encontrado una solución intermedia, muy alejada de la óptima. En este caso, parece que los pesos están estables a un valor fijo; esto sólo es aparente debido a la información mostrada en la tabla. Si se mostrara la información del error para todos los patrones en cada iteración, se podría ver cómo los pesos están lejos de permanecer estables, oscilan constantemente entre:  $\{(-0,43,0), (0,-0,43), (-0,43,0), (0,0,43)\}$ , para cada patrón, respectivamente.

El problema de este tipo de modelos, y la razón por la que fueron descartados de forma definitiva como solucionadores generales de problemas, es que sólo resuelven problemas en los que los ejemplos son linealmente separables en términos de clasificación, o en los que las salidas son funciones lineales de las entradas. El caso del decodificador es un ejemplo. Sin embargo, la mayoría de los problemas, y sobre todo los más importantes, no son separables linealmente. Para los primeros existe un gran número de técnicas estadísticas que han demostrado su gran eficacia, y el ADALINE sería entonces un método más. En cambio, los problemas no lineales siguen siendo un reto para comunidad matemática.

Esto se ilustra mejor con una representación gráfica de varios problemas de clasificación con cuatro ejemplos de entrenamiento (Figura 2.12).

En los dos primeros casos, funciones lógicas AND y OR, se aprecia claramente que existe una recta capaz de separar los ejemplos de una clase y los de otra. Sin embargo, para la función OR exclusivo, los dos gráficos inferiores, dicha recta no existe; habría que definir otra estructura, tal vez una elipse, o bien mediante dos rectas. La primera solución es claramente no lineal y, por lo tanto, ni un PERCEPTRON ni un ADALINE serían capaces de representarla. La segunda solución es la composición de dos rectas. Esta solución tampoco es posible mediante un PERCEPTRON simple; sin embargo, se podría obtener mediante la concatenación de dos PERCEPTRONES. El primero discrimina una de las regiones y la otra se introduce en un nuevo PERCEPTRON que discrimina la otra región.

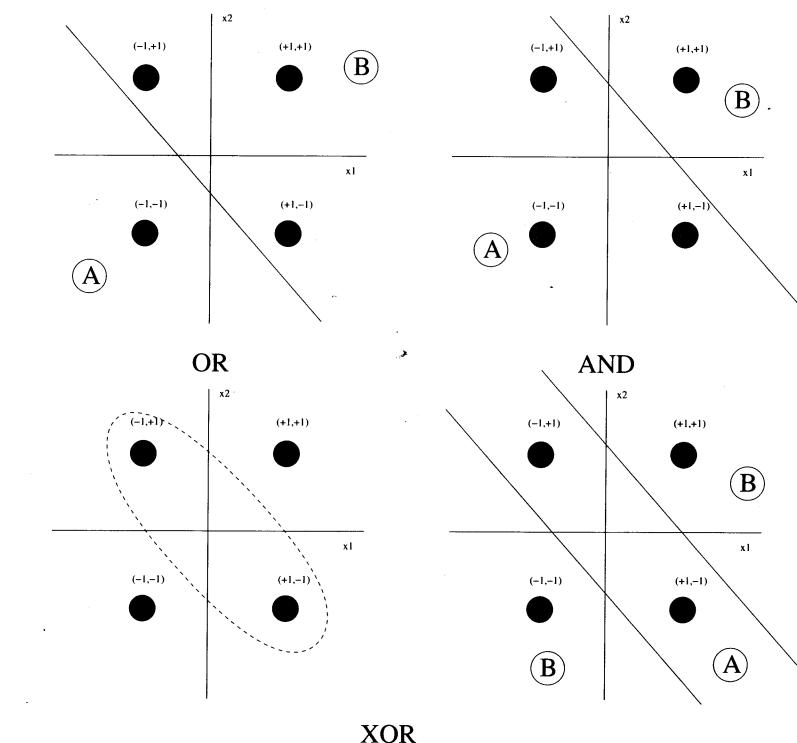


Figura 2.12: Regiones frontera en problemas de funciones lógicas

Esto se soluciona introduciendo una nueva capa, como en la Figura 2.13.

Al introducir una nueva capa, el efecto que produce es la transformación del espacio de entrada en una nueva dimensión, de modo que después de dicha transformación los ejemplos se encuentren distribuidos en el espacio y puedan ser linealmente separables, en este caso mediante un plano. En la figura 2.13 los ejemplos, después de la transformación, se han desplazado a las aristas de un cubo, dos en la cara posterior abajo, y los otros dos en diagonal. El plano que aparece en la figura sería un separador perfecto de las dos clases. Se deja como ejercicio al lector el comprobar cómo la red anterior produce salidas correctas cuando se introducen todos los ejemplos del OR exclusivo.

Sin embargo esta solución tiene varios problemas. En primer lugar es una solución a medida, es decir, no ha habido aprendizaje en función de los ejemplos de entrenamiento, sino que se han buscado los valores de los pesos que dan solución a este caso. Ahora bien, esto es inviable para problemas con un cierto grado de complejidad, aunque sea pequeño. La solución podría ser realizar un diseño genérico con dos capas (Figura 2.14).

Los pesos se inicializarían aleatoriamente, y después se aprenderían siguiendo el procedimiento del PERCEPTRON. El problema es que tanto en el PERCEPTRON

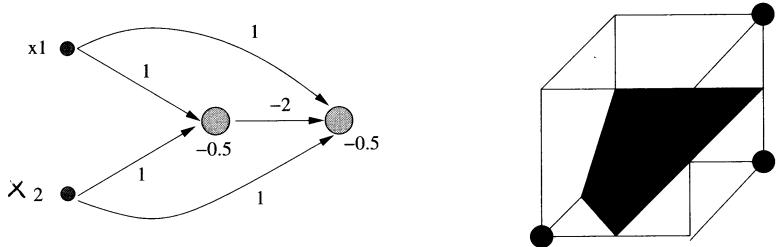


Figura 2.13: Regiones frontera en problemas de funciones lógicas

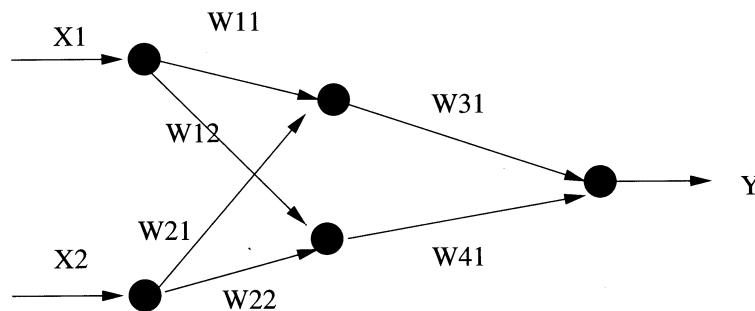


Figura 2.14: PERCEPTRON de dos capas para solucionar el XOR

como en el ADALINE, la regla de aprendizaje se basa en realizar incrementos en función del error cometido por las células. En una capa, esto es trivial, ya que sólo hay que modificar las conexiones de las células de salida, y su salida deseada está en el conjunto de patrones de entrenamiento. En el caso de la figura la modificación de las conexiones  $W_{31}$  y  $W_{41}$  se realiza de forma proporcional al error de la célula de salida. Sin embargo, en las capas intermedias no es posible saber qué valor deben tener los pesos para que la salida sea la correcta; además, habría muchos valores que cumplirían esta condición, por lo que el error cometido no podría calcularse y las conexiones  $W_{11}$ ,  $W_{12}$ ,  $W_{21}$ , y  $W_{22}$  no podrían ser modificadas. Será necesario introducir algún mecanismo adicional en la regla de aprendizaje para que estos pesos, u otros para un número arbitrario de capas, puedan ser modificados eficientemente. Hasta la década de los 80 no se descubrió el mecanismo de aprendizaje necesario, que pasaría a llamarse regla Delta generalizada. En el capítulo siguiente se describirán las redes que incluyen esta nueva regla de aprendizaje.

## Capítulo 3

# PERCEPTRON MULTICAPA

### 3.1. Introducción

En este capítulo se estudiará una de las clases de redes de neuronas, conocida como PERCEPTRON multicapa o red multicapa con conexiones hacia adelante. El PERCEPTRON multicapa es una generalización del PERCEPTRON simple presentado en el Capítulo 2, y surgió como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal. Como se ha discutido en el Capítulo 2, Minsky y Papert [Minsky and Papert, 1969] mostraron en 1969 que la combinación de varios PERCEPTRONES simples -inclusión de neuronas ocultas- podía resultar una solución adecuada para tratar ciertos problemas no lineales. Sin embargo, los autores no presentaron una solución al problema de cómo adaptar los pesos de la capa de entrada a la capa oculta, pues la regla de aprendizaje del PERCEPTRON simple no puede aplicarse en este escenario. No obstante, la idea de combinar varios PERCEPTRONES sirvió de base para estudios posteriores realizados por Rumelhart, Hinton y Willians en 1986 [Rumelhart et al., 1986b]. Estos autores presentaron una manera de retropropagar los errores medidos en la salida de la red hacia las neuronas ocultas, dando lugar a la llamada regla delta generalizada, que no es más que una generalización de la regla delta, descrita en el capítulo anterior, para funciones de activación no lineales y redes multicapa.

Diferentes autores ([Cybenko, 1989], [Hornik et al., 1989]) han demostrado independientemente que el PERCEPTRON multicapa es un aproximador universal, en el sentido de que cualquier función continua sobre un compacto de  $\mathbf{R}^n$  puede aproximarse con un PERCEPTRON multicapa, con al menos una capa oculta de neuronas. Este resultado sitúa al PERCEPTRON multicapa como una nueva clase de funciones -como pueden ser también los polinomios, las funciones trigonométricas, los splines, etc.- para aproximar o interpolar relaciones no lineales

entre datos de entrada y salida.

La habilidad del PERCEPTRON multicapa para aprender a partir de un conjunto de ejemplos, aproximar relaciones no lineales, filtrar ruido en los datos, etc. hace que sea un modelo adecuado para abordar problemas reales, sin que esto indique que sean los mejores aproximadores universales. Cada una de las clases de aproximadores tienen sus propias características; se conocen ciertas condiciones bajo las cuales un método es preferible a otro, pero en ningún caso se puede decir que un método sea absolutamente el mejor. Serán las consideraciones prácticas de cada problema las que determinen la elección de un aproximador u otro.

Dentro del marco de las redes de neuronas, el PERCEPTRON multicapa es en la actualidad una de las arquitecturas más utilizadas en la resolución de problemas. Esto es debido, fundamentalmente, a su capacidad como aproximador universal, como se comentó anteriormente, así como a su fácil uso y aplicabilidad. Estas redes han sido aplicadas con éxito para la resolución de problemas en una gran variedad de áreas diferentes, como reconocimiento de habla [Cohen et al., 1993], reconocimiento de caracteres ópticos [Sackinger et al., 1992], reconocimiento de caracteres escritos [Guyon, 1991], control de procesos [Werbos, 1989], modelización de sistemas dinámicos [Narendra and Parthasarathy, 1990], conducción de vehículos [Pomerleau, 1992], diagnósticos médicos [Baxt, 1992], predicción de series temporales [Weigend et al., 1990], etc.

Es necesario señalar, sin embargo, que aunque sea una de las redes más conocidas y utilizadas, esto no implica que sea una de las más potentes y con mejores resultados en las diferentes áreas de aplicación. De hecho, el PERCEPTRON multicapa posee también una serie de limitaciones, como el largo proceso de aprendizaje para problemas complejos dependientes de un gran número de variables; la dificultad en ocasiones de codificar problemas reales mediante valores numéricos; la dificultad para realizar un análisis teórico de la red debido a la presencia de componentes no lineales y a la alta conectividad. Por otra parte, es necesario señalar que el proceso de aprendizaje de la red busca en un espacio amplio de funciones, una posible función que relacione las variables de entrada y salida al problema, lo cual puede complicar su aprendizaje y reducir su efectividad en determinadas aplicaciones [Hinton, 1990].

### 3.2. Arquitectura del perceptron multicapa

La arquitectura del PERCEPTRON multicapa se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada, las capas ocultas y la capa de salida, como se observa en la Figura 3.1.

Las neuronas de la capa de entrada no actúan como neuronas propiamente dichas, sino que se encargan únicamente de recibir las señales o patrones que proceden del exterior y propagar dichas señales a todas las neuronas de la siguiente capa. La última capa actúa como salida de la red, proporcionando al exterior la respuesta de la red para cada uno de los patrones de entrada. Las

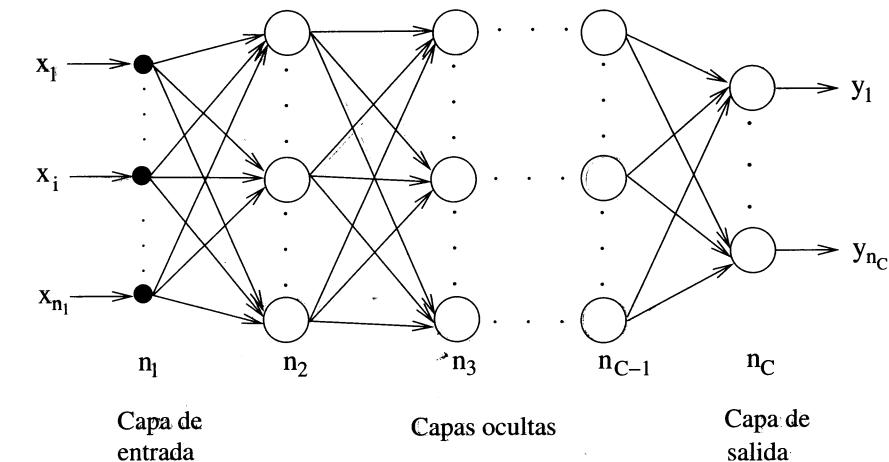


Figura 3.1: Arquitectura del PERCEPTRON multicapa

neuronas de las capas ocultas realizan un procesamiento no lineal de los patrones recibidos.

Como se observa en la Figura 3.1, las conexiones del PERCEPTRON multicapa siempre están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa, de ahí que reciban también el nombre de redes alimentadas hacia adelante o redes "feedforward". Las conexiones entre las neuronas llevan asociado un número real, llamado peso de la conexión. Todas las neuronas de la red llevan también asociado un umbral, que en el caso del PERCEPTRON multicapa suele tratarse como una conexión más a la neurona, cuya entrada es constante e igual a 1.

Generalmente, todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente capa. De este modo, las neuronas de la capa de entrada están conectadas a todas las neuronas de la primera capa oculta; las neuronas de la primera capa oculta se conectan a las neuronas de la siguiente capa, etc. Se dice entonces que existe conectividad total o que la red está totalmente conectada.

Aunque en la mayor parte de los casos la arquitectura del PERCEPTRON multicapa está asociada al esquema de la Figura 3.1, es posible también englobar dentro de este tipo de redes a arquitecturas con las siguientes características:

- Redes con conexiones de todas o ciertas neuronas de una determinada capa a neuronas de capas posteriores, aunque no inmediatamente posteriores.
- Redes en las que ciertas neuronas de ciertas capas no están conectadas a neuronas de la siguiente capa, es decir el peso de la conexión es constante e igual a cero.

Cuando se aborda un problema con el PERCEPTRON multicapa, en la mayoría

de los casos se parte de una arquitectura totalmente conectada, es decir, todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente capa. No es posible demostrar que si se utilizan arquitecturas en las que se eliminan conexiones o se añaden conexiones de una capa a capas no inmediatamente posteriores, se puedan obtener mejores resultados. Sin embargo, en ocasiones, y debido fundamentalmente a la naturaleza del problema, se pueden encontrar redes multicapa con estas características en sus conexiones.

### 3.2.1. Propagación de los patrones de entrada

El PERCEPTRON multicapa define una relación entre las variables de entrada y las variables de salida de la red. Esta relación se obtiene propagando hacia adelante los valores de las variables de entrada. Para ello, cada neurona de la red procesa la información recibida por sus entradas y produce una respuesta o activación que se propaga, a través de las conexiones correspondientes, hacia las neuronas de la siguiente capa. A continuación, se muestran las expresiones para calcular las activaciones de las neuronas de la red.

Sea un PERCEPTRON multicapa con  $C$  capas - $C - 2$  capas ocultas- y  $n_q$  neuronas en la capa  $q$ , para  $q = 1, 2, \dots, C$ . Sea  $W^q = (w_{ij}^q)$  la matriz de pesos asociada a las conexiones de la capa  $q$  a la capa  $q + 1$  para  $q = 1, 2, \dots, C - 1$ , donde  $w_{ij}^q$  representa el peso de la conexión de la neurona  $i$  de la capa  $c$  a la neurona  $j$  de la capa  $q + 1$ ; y sea  $U^q = (u_i^q)$  el vector de umbrales de las neuronas de la capa  $q$  para  $q = 2, \dots, C$ . Se denota  $a_i^q$  a la activación de la neurona  $i$  de la capa  $q$ ; estas activaciones se calculan del siguiente modo:

- Activación de las neuronas de la capa de entrada ( $a_i^1$ ). Las neuronas de la capa de entrada se encargan de transmitir hacia la red las señales recibidas del exterior. Por tanto:

$$a_i^1 = x_i \text{ para } i = 1, 2, \dots, n_1 \quad (3.1)$$

donde  $X = (x_1, x_2, \dots, x_{n_1})$  representa el vector o patrón de entrada a la red.

- Activación de las neuronas de la capa oculta  $q$  ( $a_i^q$ ). Las neuronas ocultas de la red procesan la información recibida aplicando la función de activación  $f$  a la suma de los productos de las activaciones que recibe por sus correspondientes pesos, es decir:

$$a_i^q = f\left(\sum_{j=1}^{n_{q-1}} w_{ji}^{q-1} a_j^{q-1} + u_i^q\right) \text{ para } i = 1, 2, \dots, n_q \text{ y } q = 2, 3, \dots, C - 1 \quad (3.2)$$

donde  $a_j^{q-1}$  son las activaciones de las neuronas de la capa  $q - 1$ .

- Activación de las neuronas de capa de salida ( $a_i^C$ ). Al igual que en el caso anterior, la activación de estas neuronas viene dada por la función

de activación  $f$  aplicada a la suma de los productos de las entradas que recibe por sus correspondientes pesos:

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_C-1} w_{ji}^{C-1} a_j^{C-1} + u_i^C\right) \text{ para } i = 1, 2, \dots, n_C \quad (3.3)$$

donde  $Y = (y_1, y_2, \dots, y_{n_C})$  es el vector salida de la red.

La función  $f$  es la llamada *función de activación*. Para el PERCEPTRON multicapa, las funciones de activación más utilizadas son la función sigmoidal y la función tangente hiperbólica. Dichas funciones poseen como imagen un rango continuo de valores dentro de los intervalos  $[0, 1]$  y  $[-1, 1]$ , respectivamente, y vienen dadas por las siguientes expresiones:

- Función sigmoidal:

$$f_1(x) = \frac{1}{1 + e^{-x}} \quad (3.4)$$

- Función tangente hiperbólica:

$$f_2(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (3.5)$$

Ambas son funciones crecientes con dos niveles de saturación: el máximo, que proporciona salida 1, y el mínimo, salida 0 para la función sigmoidal y salida -1 para la tangente hiperbólica, como se observa en la Figura 3.2.

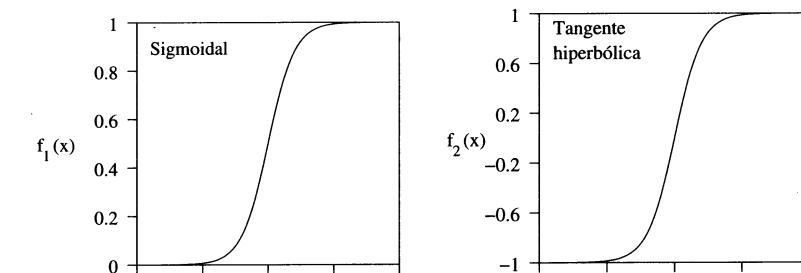


Figura 3.2: Funciones de activación del PERCEPTRON multicapa

Generalmente, la función de activación en el PERCEPTRON multicapa es común a todas las neuronas de red y es elegida por el diseñador, elección que se realiza únicamente basándose en los valores de activación que se desee que alcancen las neuronas. Ambas funciones están relacionadas mediante la expresión  $f_2(x) = 2f_1(x) - 1$ , por lo que la utilización de una u otra se elige únicamente en función del recorrido que interese.

En ocasiones, y dependiendo de la naturaleza del problema, las neuronas de salidas se distinguen del resto de las neuronas de la red, utilizando otro tipo de función de activación. En este caso, las más usadas son la función identidad y la función escalón.

De las Ecuaciones (1.1), (1.2) y (1.3), se observa que el PERCEPTRON multicapa define, a través de sus conexiones y neuronas, una función continua no lineal del espacio  $\mathbf{R}^{n_1}$  -espacio de los patrones de entrada- al espacio  $\mathbf{R}^{n_C}$  -espacio de los patrones de salida-. Se puede escribir, por tanto, que:

$$Y = F(X, W) \quad (3.6)$$

donde  $Y$  es el vector formado por las salidas de la red,  $X$  es el vector de entrada a la red,  $W$  es el conjunto de todos los parámetros de la red -pesos y umbrales- y  $F$  es una función continua no lineal dada por las Ecuaciones (1.1), (1.2) y (1.3).

### Cálculo de las activaciones para un Perceptrón multicapa con dos neuronas de entrada, dos ocultas y una salida

Sea el PERCEPTRON multicapa que se muestra en la Figura 3.3, con un única capa oculta. Si  $x_1$  y  $x_2$  denotan las variables de entrada a la red, las activaciones de las neuronas de la red vienen dadas por las siguientes expresiones:

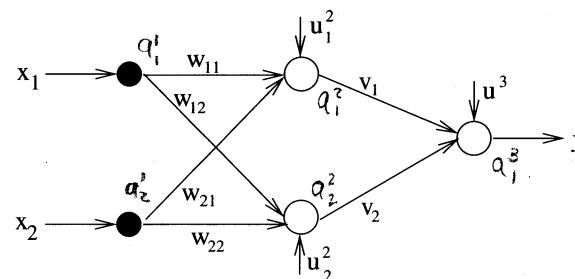


Figura 3.3: Ejemplo de arquitectura del PERCEPTRON multicapa

- Activaciones de las neuronas de entrada:

$$a_1^1 = x_1 \text{ y } a_2^1 = x_2$$

- Activaciones de las neuronas ocultas:

$$a_1^2 = f(w_{11}a_1^1 + w_{21}a_2^1 + u_1^2) \text{ y } a_2^2 = f(w_{12}a_1^1 + w_{22}a_2^1 + u_2^2)$$

- Activación de la neurona de salida:

$$y = a_3^3 = f(v_1a_1^2 + v_2a_2^2 + u^3)$$

Cuadro 3.1: Función XOR

Entrada	Salida
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	0

Cuadro 3.2: Activaciones del PERCEPTRON multicapa para la función XOR

Patrón de entrada	$a_1^1$	$a_2^1$	$a_1^2$	$a_2^2$	Salida de la red
(0, 0)	0	0	0,129781	0,015875	0,128883
(0, 1)	0	1	0,972618	0,204662	0,863310
(1, 0)	1	0	0,964023	0,202889	0,858615
(1, 1)	1	0	0,999843	0,802384	0,154993

Considérense los patrones que definen la función lógica XOR (cuadro 3.1). Si los pesos y umbrales de la red que se muestra en la Figura 3.3 toman los valores  $w_{11} = 5,191129$ ,  $w_{12} = 2,758669$ ,  $w_{21} = 5,473012$ ,  $w_{22} = 2,769596$ ,  $v_1 = 5,839709$ ,  $v_2 = -6,186834$ ,  $u_1^2 = -1,90289$ ,  $u_2^2 = -4,127002$  y  $u^3 = -2,570539$ , sustituyendo dichos parámetros en las expresiones anteriores y utilizando la función de activación sigmoidal (Ecuación 3.4), la salida de la red para los patrones de la función XOR toma los valores especificados en el cuadro 3.2.

### 3.2.2. Diseño de la arquitectura del perceptrón multicapa

Cuando se aborda un problema utilizando el PERCEPTRON multicapa, uno de los primeros pasos a realizar es el diseño de la arquitectura de la red. Este diseño implica la determinación de la función de activación a emplear, el número de neuronas y el número de capas en la red.

Como se ha comentado anteriormente, la elección de la función de activación se suele hacer basándose en el recorrido deseado, y el hecho de elegir una u otra, generalmente, no influye en la capacidad de la red para resolver el problema.

En lo que respecta al número de neuronas y capas, algunos de estos parámetros vienen dados por el problema y otros deben ser elegidos por el diseñador. Así, por ejemplo, tanto el número de neuronas en la capa de entrada, como el número de neuronas en la capa de salida, vienen dados por las variables que definen el problema. En algunas aplicaciones prácticas, no hay lugar a duda sobre el número de entradas y salidas. Sin embargo, existen problemas en los que el número de variables de entrada relevantes para el problema no se conoce con exactitud. En estos casos, se dispone de un gran número de variables, algunas de las cuales podrían no aportar información relevante a la red, y su utilización podría complicar el aprendizaje, pues implicaría arquitecturas

de gran tamaño y con alta conectividad. En estas situaciones, es conveniente realizar un análisis previo de las variables de entrada más relevantes al problema y descartar aquellas que no aportan información a la red. Este análisis puede llegar a ser una tarea complicada y requerir técnicas avanzadas, como técnicas basadas en análisis de correlación [Battini, 1994], análisis de componentes principales [Jolliffe, 1986], análisis de sensibilidad de redes de neuronas ([Mao and Jain, 1995], [Zurada et al., 1997]) y técnicas basadas en algoritmos genéticos [Guo and Uhrig, 1992], entre otras.

El número de capas ocultas y el número de neuronas en estas capas deben ser elegidos por el diseñador. No existe un método o regla que determine el número óptimo de neuronas ocultas para resolver un problema dado. En la mayor parte de las aplicaciones prácticas, estos parámetros se determinan por prueba y error. Partiendo de una arquitectura ya entrenada, se realizan cambios aumentando y disminuyendo el número de neuronas ocultas y el número de capas hasta conseguir una arquitectura adecuada para el problema a resolver, que pudiera no ser la óptima, pero que proporciona una solución.

Si bien el número de neuronas ocultas puede influir en el comportamiento de la red, como se verá más adelante -capacidad de generalización de la red-, es necesario indicar que en el caso del PERCEPTRON multicapa, generalmente, el número de neuronas ocultas no es parámetro significativo, pues dado un problema, pueden existir una gran cantidad de arquitecturas capaces de resolver de manera adecuada dicho problema. Además, añadir o eliminar una neurona oculta no influye, de manera significativa, en la capacidad de la red.

En la actualidad existen líneas de investigación abiertas centradas en la determinación automática del número óptimo de neuronas ocultas, así como de capas ocultas, para cada problema en particular. En el caso del PERCEPTRON multicapa, la mayor parte de estos trabajos se basan en la utilización de técnicas evolutivas, las cuales realizan una búsqueda en el espacio de las arquitecturas de redes guiada por la optimización del rendimiento de la red ([Miller et al., 1989], [Yao and Lin, 1997], [Gutiérrez et al., 2001]).

### 3.3. Algoritmo de RetroPropagación

La regla o algoritmo de aprendizaje es el mecanismo mediante cual se van adaptando y modificando todos los parámetros de la red. En el caso del PERCEPTRON multicapa se trata de un algoritmo de aprendizaje supervisado; es decir, la modificación de los parámetros se realiza para que la salida de la red sea lo más próxima posible a la salida proporcionada por el supervisor o salida deseada. Por tanto, para cada patrón de entrada a la red es necesario disponer de un patrón de salida deseada.

Puesto que el objetivo es que la salida de la red sea lo más próxima posible a la salida deseada, el aprendizaje de la red se formula como un problema de minimización del siguiente modo:

$$\text{Min}_W E \quad (3.7)$$

siendo  $W$  el conjunto de parámetros de la red -pesos y umbrales- y  $E$  una función error que evalúa la diferencia entre las salidas de la red y las salidas deseadas. En la mayor parte de los casos, la función error se define como:

$$E = \frac{1}{N} \sum_{n=1}^N e(n) \quad (3.8)$$

donde  $N$  es el número de patrones o muestras y  $e(n)$  es el error cometido por la red para el patrón  $n$ , dado por:

$$e(n) = \frac{1}{2} \sum_{i=1}^{n_C} (s_i(n) - y_i(n))^2 \quad (3.9)$$

siendo  $Y(n) = (y_1(n), \dots, y_{n_C}(n))$  y  $S(n) = (s_1(n), \dots, s_{n_C}(n))$  los vectores de salidas de la red y salidas deseadas para el patrón  $n$ , respectivamente.

De este modo, si  $W^*$  es un mínimo de la función error  $E$ , en dicho punto el error es próximo a cero, lo cual implica que la salida de la red es próxima a la salida deseada, alcanzando así la meta de la regla de aprendizaje.

Por tanto, el aprendizaje del PERCEPTRON multicapa es equivalente a encontrar un mínimo de la función error. La presencia de funciones de activación no lineales hace que la respuesta de la red sea no lineal respecto a los parámetros ajustables, por lo que el problema de minimización es un problema no lineal, y, como consecuencia, tienen que utilizarse técnicas de optimización no lineales para su resolución. Dichas técnicas están, generalmente, basadas en una adaptación de los parámetros siguiendo una cierta dirección de búsqueda. En el contexto de redes de neuronas, y en particular para el PERCEPTRON multicapa, la dirección de búsqueda más comúnmente usada es la dirección negativa del gradiente de la función  $E$  -método de descenso del gradiente-, pues conforme al cálculo de varias variables, ésta es la dirección en la que la función decrece. No obstante, se han desarrollado también métodos de búsqueda aleatoria para localizar el mínimo de dicha función [Solis and Wets, 1981] y métodos basados en técnicas evolutivas [Montana and Davis, 1989], en los que la búsqueda está guiada por una función de adecuación.

Aunque, estrictamente hablando, el aprendizaje de la red debe realizarse para minimizar el error total (Ecuación 3.8), el procedimiento más utilizado está basado en métodos del gradiente estocástico, los cuales consisten en una sucesiva minimización de los errores para cada patrón,  $e(n)$ , en lugar de minimizar el error total  $E$ . Por tanto, aplicando el método de descenso del gradiente estocástico, cada parámetro  $w$  de la red se modifica para cada patrón de entrada  $n$  de acuerdo con la siguiente ley de aprendizaje:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w} \quad (3.10)$$

donde  $e(n)$  es el error para el patrón  $n$  dado por la Ecuación (3.9) y  $\alpha$  es la razón o tasa de aprendizaje, parámetro que influye en la magnitud del desplazamiento en la superficie del error, como se analizará más adelante.

Debido a que las neuronas de la red están agrupadas en capas de distintos niveles, es posible aplicar el método del gradiente de forma eficiente, resultando el conocido algoritmo de RETROPROPAGACIÓN [Rumelhart et al., 1986b] o *regla delta generalizada*. El término de retropropagación se utiliza debido a la forma de implementar el método del gradiente en el PERCEPTRON multicapa, pues el error cometido en la salida de la red es propagado hacia atrás, transformándolo en un error para cada una de las neuronas ocultas de la red.

### 3.3.1. Obtención de la regla delta generalizada

A continuación se va a desarrollar la regla delta generalizada para el aprendizaje del PERCEPTRON multicapa, que, como ya se ha comentado, no es más que una forma eficiente de aplicar el método del gradiente a los parámetros -pesos y umbrales- de dicha arquitectura. Para el desarrollo de esta regla es necesario distinguir dos casos: uno para los pesos de la capa oculta  $C - 1$  a la capa de salida y para los umbrales de las neuronas de salida, y otro para el resto de los pesos y umbrales de la red, pues las reglas de modificación de estos parámetros son diferentes.

#### Pesos de la capa oculta $C - 1$ a la capa de salida y umbrales de la capa de salida

Sea  $w_{ji}^{C-1}$  el peso de la conexión de la neurona  $j$  de la capa  $C - 1$  a la neurona  $i$  de la capa de salida. Utilizando el método de descenso del gradiente (Ecuación 3.10), dicho parámetro se modifica siguiendo la dirección negativa del gradiente del error:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) - \alpha \frac{\partial e(n)}{\partial w_{ji}^{C-1}} \quad (3.11)$$

Por tanto, para la actualización de dicho parámetro es necesario evaluar la derivada del error  $e(n)$  en dicho punto. De acuerdo con la expresión del error (Ecuación 3.9) y teniendo en cuenta, por un lado, que las salidas deseadas  $s_i(n)$  para la red son constantes que no dependen del peso y, por otro lado, que el peso  $w_{ji}^{C-1}$  sólo afecta a la neurona de salida  $i$ ,  $y_i(n)$  (ver Ecuación 3.3), se obtiene que:

$$\frac{\partial e(n)}{\partial w_{ji}^{C-1}} = -(s_i(n) - y_i(n)) \frac{\partial y_i(n)}{\partial w_{ji}^{C-1}} \quad (3.12)$$

A este punto, hay que calcular la derivada de la neurona de salida  $y_i(n)$  respecto al peso  $w_{ji}^{C-1}$ . La salida de la red es la función de activación  $f$  aplicada a la suma de todas las entradas por sus pesos, como se muestra en la Ecuación (3.3). Aplicando la regla de la cadena para derivar la composición de dos funciones y teniendo en cuenta que, de todos los términos del sumatorio (Ecuación 3.3), el único en el que interviene el peso  $w_{ji}^{C-1}$  es  $w_{ji}^{C-1}a_j^{C-1}$ , y, por tanto, el único cuya derivada es distinta de cero, se obtiene:

$$\frac{\partial y_i(n)}{\partial w_{ji}^{C-1}} = f' \left( \sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + u_i^C \right) a_j^{C-1}(n) \quad (3.13)$$

Se define el término  $\delta$  asociado a la neurona  $i$  de la capa de salida -capa  $C$ - y al patrón  $n$ ,  $\delta_i^C(n)$ , del siguiente modo:

$$\delta_i^C(n) = -(s_i(n) - y_i(n)) f' \left( \sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + u_i^C \right) \quad (3.14)$$

Reemplazando entonces en la Ecuación (3.12) el valor de la derivada de la neurona de salida  $y_i(n)$  dado por (3.13) y de acuerdo con el valor  $\delta_i^C(n)$  definido anteriormente (Ecuación 3.14), se obtiene que:

$$\frac{\partial e(n)}{\partial w_{ji}^{C-1}} = \delta_i^C(n) a_j^{C-1}(n) \quad (3.15)$$

Finalmente, reemplazando la derivada del error  $e(n)$  respecto al peso  $w_{ji}^{C-1}$  obtenida en (3.15) en la Ecuación (3.11), se obtiene la ley para modificar dicho peso, la cual toma la siguiente expresión:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha \delta_i^C(n) a_j^{C-1}(n) \quad (3.16)$$

para  $j = 1, 2, \dots, n_{C-1}$   $i = 1, 2, \dots, n_C$

En la Ecuación (3.16) obtenida anteriormente se observa que para modificar el peso de la conexión de la neurona  $j$  de la capa  $C - 1$  a la neurona  $i$  de la capa de salida, basta considerar la activación de la neurona de la que parte la conexión -neurona  $j$  de la capa  $C - 1$ - y el término  $\delta$  de la neurona a la que llega la conexión -neurona de salida  $i$ -, término que contiene el error cometido por la red para dicha neurona de salida (ver Ecuación 3.14).

La ley de aprendizaje obtenida anteriormente para modificar los pesos de última capa puede generalizarse para los umbrales de las neuronas de salida. Como se ha comentado en la sección anterior, en el PERCEPTRON multicapa el umbral de una neurona se trata como una conexión más a la neurona cuya entrada es constante e igual a 1. Siguiendo, entonces, la ley anterior (Ecuación 3.16), se deduce que los umbrales de las neuronas de la capa de salida se modifican de acuerdo con la siguiente expresión:

$$u_i^C(n) = u_i^C(n-1) + \alpha \delta_i^C(n) \text{ para } i = 1, 2, \dots, n_C \quad (3.17)$$

#### Pesos de la capa $c$ a la capa $c + 1$ y umbrales de las neuronas de la capa $c + 1$ para $c = 2, \dots, C - 2$

Con el objetivo de que el desarrollo de la regla de aprendizaje para el resto de los pesos y umbrales de la red sea lo más claro posible, se elige un peso de la capa  $C - 2$  a la capa  $C - 1$ . Sea  $w_{kj}^{C-2}$  el peso de la conexión de la neurona  $k$  de

la capa  $C - 2$  a la neurona  $j$  de la capa  $C - 1$ . Siguiendo el método de descenso del gradiente, la ley para actualizar dicho peso viene dada por:

$$w_{kj}^{C-2}(n) = w_{kj}^{C-2}(n-1) + \alpha \frac{\partial e(n)}{\partial w_{kj}^{C-2}} \quad (3.18)$$

En este caso, y a diferencia del anterior -pesos hacia la capa de salida-, el peso  $w_{kj}^{C-2}$  influye en todas las salidas de la red, por lo que la derivada del error  $e(n)$  (Ecuación 3.9) respecto de dicho peso viene dada por la suma de las derivadas para cada una de las salidas de la red, es decir:

$$\frac{\partial e(n)}{\partial w_{kj}^{C-2}} = - \sum_{i=1}^{n_C} (s_i(n) - y_i(n)) \frac{\partial y_i(n)}{\partial w_{kj}^{C-2}} \quad (3.19)$$

Para calcular la derivada de la salida  $y_i(n)$  respecto al peso  $w_{kj}^{C-2}$  es necesario tener en cuenta que este peso influye en la activación de la neurona  $j$  de la capa oculta  $C - 1$ ,  $a_j^{C-1}$ , y que el resto de las activaciones de las neuronas en esta capa no dependen de dicho peso. Por tanto, y de acuerdo con la Ecuación (3.3), se tiene que:

$$\frac{\partial y_i(n)}{\partial w_{kj}^{C-2}} = f' \left( \sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + u_i^C \right) w_{ji}^{C-1} \frac{\partial a_j^{C-1}}{\partial w_{kj}^{C-2}} \quad (3.20)$$

Sustituyendo este valor en la Ecuación (3.19) y de acuerdo con la definición de  $\delta$  en el punto anterior (Ecuación 3.14), se obtiene que:

$$\frac{\partial e(n)}{\partial w_{kj}^{C-2}} = \sum_{i=1}^{n_C} \delta_i^C(n) w_{ji}^{C-1} \frac{\partial a_j^{C-1}}{\partial w_{kj}^{C-2}} \quad (3.21)$$

Para obtener la ley de aprendizaje para el peso  $w_{kj}^{C-2}$ , sólo falta derivar la activación de la neurona  $j$  de la capa oculta  $C - 1$ ,  $a_j^{C-1}$ , respecto a dicho peso. De nuevo, aplicando la regla de la cadena a la Ecuación (3.2), dicha derivada es:

$$\frac{\partial a_j^{C-1}}{\partial w_{kj}^{C-2}} = f' \left( \sum_{k=1}^{n_{C-2}} w_{kj}^{C-2} a_k^{C-2} + u_j^{C-1} \right) a_k^{C-2}(n) \quad (3.22)$$

Se define el valor  $\delta$  para las neuronas de la capa  $C - 1$ ,  $\delta_j^{C-1}(n)$ , como:

$$\delta_j^{C-1}(n) = f' \left( \sum_{k=1}^{n_{C-2}} w_{kj}^{C-2} a_k^{C-2} + u_j^{C-1} \right) \sum_{i=1}^{n_C} \delta_i^C(n) w_{ji}^{C-1} \quad (3.23)$$

Sustituyendo (3.22) en la Ecuación (3.21) y de acuerdo con el valor  $\delta_j^{C-1}(n)$  definido anteriormente, se obtiene que:

$$\frac{\partial e(n)}{\partial w_{kj}^{C-2}} = \delta_j^{C-1}(n) a_k^{C-2}(n) \quad (3.24)$$

Y como consecuencia, la ley de aprendizaje para modificar el peso  $w_{kj}^{C-2}$  viene dada por:

$$w_{kj}^{C-2}(n) = w_{kj}^{C-2}(n-1) + \alpha \delta_j^{C-1}(n) a_k^{C-2}(n) \quad (3.25)$$

para  $k = 1, 2, \dots, n_{C-2}$  y  $j = 1, 2, \dots, n_{C-1}$

Al igual que la ley obtenida para modificar los pesos de la última capa (Ecuación 3.16), en este caso, también se observa que para modificar el peso de la conexión de la neurona  $k$  de la capa  $C - 2$  a la neurona  $j$  de la capa de  $C - 1$ , basta considerar la activación de la neurona de la que parte la conexión -neurona  $k$  de la capa  $C - 2$ - y el término  $\delta$  de la neurona a la que llega la conexión -neurona  $j$  de la capa  $C - 1$ - (ver Ecuación 3.25). La diferencia radica en la expresión del término  $\delta$ . Para los pesos de la capa  $C - 2$  a  $C - 1$  dicho término viene dado por la derivada de la función de activación y por la suma de los términos  $\delta$  asociados a las neuronas de la siguiente capa -en este caso, las neuronas de salida- por los pesos correspondientes, como se indica en la Ecuación (3.23).

A este punto, es posible generalizar fácilmente la ley dada por la Ecuación (3.25) para los pesos de la capa  $c$  a la capa  $c + 1$  ( $c = 1, 2, \dots, C - 2$ ). Para ello, basta tener en cuenta la activación de la que parte la conexión y el término  $\delta$  de la neurona a la que llega la conexión. Este término se calcula utilizando la derivada de la función de activación y la suma de los términos  $\delta$  de las neuronas de la siguiente capa. De este modo:

$$w_{kj}^c(n) = w_{kj}^c(n-1) + \alpha \delta_j^{c+1}(n) a_k^c(n) \quad (3.26)$$

para  $k = 1, 2, \dots, n_c$ ,  $j = 1, 2, \dots, n_{c+1}$  y  $c = 1, 2, \dots, C - 2$

donde  $a_k^c(n)$  es la activación de la neurona  $k$  de la capa  $c$  para el patrón  $n$  y  $\delta_j^{c+1}(n)$  viene dado por la siguiente expresión:

$$\delta_j^{c+1}(n) = f' \left( \sum_{k=1}^{n_c} w_{kj}^c a_k^c + u_j^c \right) \sum_{i=1}^{n_{c+1}} \delta_i^{c+2}(n) w_{ji}^c \quad (3.27)$$

Es posible, también, generalizar la ley de aprendizaje para el resto de los umbrales de la red; basta tratarlos como conexiones cuya entrada es constante e igual a 1. La ley para modificarlos viene dada por:

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \alpha \delta_j^{c+1}(n) \quad (3.28)$$

para  $j = 1, 2, \dots, n_{c+1}$  y  $c = 1, 2, \dots, C - 2$

#### Derivada de la función de activación

El cálculo de los valores  $\delta$  (Ecuaciones 3.14 y 3.27) para cada neurona del PERCEPTRON multicapa requiere el cálculo de la derivada de la función de activación. Como se ha comentado, el PERCEPTRON multicapa puede utilizar dos tipos de funciones de activación -función sigmoidal (Ecuación 3.4- y función tangente hiperbólica (Ecuación 3.5)), por lo que los valores  $\delta$  dependen, en principio, de la

función de activación empleada. A continuación, se van a obtener las derivadas para ambas funciones de activación, quedando así completado el cálculo de los valores  $\delta$ .

- Derivada de la función sigmoidal

Derivando la expresión dada por la Ecuación (3.4), se obtiene que:

$$f'_1(x) = \frac{1}{(1 + e^{-x})^2}(-e^{-x}) = \frac{1}{1 + e^{-x}} \frac{e^{-x}}{1 + e^{-x}} \quad (3.29)$$

Por tanto,

$$f'_1(x) = f_1(x)(1 - f_1(x)) \quad (3.30)$$

Como consecuencia, cuando se utiliza la función sigmoidal, los valores  $\delta$  asociados a las neuronas de salida (Ecuación 3.14) adoptan la siguiente forma:

$$\delta_i^C(n) = -(s_i(n) - y_i(n))y_i(n)(1 - y_i(n)) \text{ para } i = 1, 2, \dots, n_C \quad (3.31)$$

Y los valores  $\delta$  para el resto de las neuronas de la red (Ecuación 3.27), vienen dados por:

$$\delta_j^{c+1}(n) = a_j^c(n)(1 - a_j^c(n)) \sum_{i=1}^{n_{c+1}} \delta_i^{c+2}(n)w_{ji}^c \quad (3.32)$$

para  $j = 1, 2, \dots, n_{c+1}$  y  $c = 1, 2, \dots, C - 2$

- Derivada de la función tangente hiperbólica

Teniendo en cuenta que  $f_2(x) = 2f_1(x) - 1$ , la derivada de la función  $f_2(x)$  es:

$$f'_2(x) = 2f_1(x)(1 - f_1(x)) \quad (3.33)$$

Cuando se utilice, por tanto, la función de activación tangente hiperbólica, los valores  $\delta$  para las neuronas de la red adoptan las expresiones dadas por las Ecuaciones (3.31) y (3.32) multiplicadas por un factor de 2.

En la sección anterior se ha comentado que, generalmente, todas las neuronas del PERCEPTRON multicapa suelen utilizar la misma función de activación, salvo las neuronas de salida, que pueden tener como función de activación la función identidad, es decir,  $f(x) = x$ . En este caso, la derivada de la función de activación en la salida es 1 y, como consecuencia, los valores  $\delta$  para las neuronas de salida adoptan la siguiente expresión:

$$\delta_i^C(n) = -(s_i(n) - y_i(n)) \text{ para } i = 1, 2, \dots, n_C \quad (3.34)$$

### Obteniendo la regla delta generalizada para un perceptrón multicapa con dos neuronas de entrada, dos ocultas y una salida

Para el PERCEPTRON multicapa que se muestra en la Figura 3.3, el error cometido en la salida viene dado por:

$$e(n) = \frac{1}{2}(s(n) - y(n))^2$$

Para obtener la regla delta generalizada es necesario derivar dicho error respecto a cada uno de los parámetros de la red. A continuación, se muestra cómo se obtienen estas derivadas para esta arquitectura en particular, lo cual puede facilitar la comprensión del procedimiento general.

- **Pesos de la capa oculta a la capa de salida.** Sea el peso  $v_1$  de la neurona oculta 1 a la neurona de salida de la red. La derivada del error respecto a dicho parámetro adopta la siguiente expresión:

$$\frac{\partial e(n)}{\partial v_1} = -(s(n) - y(n)) \frac{\partial y(n)}{\partial v_1}$$

Teniendo en cuenta que  $y = f(v_1a_1^2 + v_2a_2^2 + u^3)$  y aplicando la regla de la cadena,

$$\frac{\partial y(n)}{\partial v_1} = f'(v_1a_1^2 + v_2a_2^2 + u^3)a_1^2(n)$$

Definiendo el término  $\delta^3(n) = -(s(n) - y(n))f'(v_1a_1^2 + v_2a_2^2 + u^3)$ , y de acuerdo con el método de descenso del gradiente (Ecuación 3.10), se obtiene que:

$$v_1(n) = v_1(n-1) + \alpha\delta^3(n)a_1^2(n)$$

Se observa en la expresión anterior que la ley para modificar el parámetro  $v_1$  viene dada por la activación de primera neurona de la capa oculta -de donde parte la conexión- por el término  $\delta$  asociado a la neurona de salida -donde llega la conexión-. Por extensión, se obtiene que:

$$v_2(n) = v_2(n-1) + \alpha\delta^3(n)a_2^2(n)$$

$$u^3(n) = u^3(n-1) + \alpha\delta^3(n)$$

- **Pesos de la capa de entrada a la capa de salida.** Sea, por ejemplo, el peso  $w_{12}$  de la neurona de entrada 1 a la neurona oculta 2. La derivada del error respecto a dicho parámetro es:

$$\frac{\partial e(n)}{\partial w_{12}} = -(s(n) - y(n)) \frac{\partial y(n)}{\partial w_{12}}$$

Teniendo en cuenta la expresión de la salida de la red, y que el parámetro  $w_{12}$  sólo interviene en la activación de la segunda neurona oculta  $a_2^2$ ,

$$\frac{\partial y(n)}{\partial w_{12}} = f' \left( v_1 a_1^2 + v_2 a_2^2 + u^3 \right) v_2 \frac{\partial a_2^2(n)}{\partial w_{12}}$$

Puesto que  $a_2^2 = f(w_{12}a_1^1 + w_{22}a_2^1 + u_2^2)$ ,

$$\frac{\partial a_2^2(n)}{\partial w_{12}} = f' \left( w_{12}a_1^1 + w_{22}a_2^1 + u_2^2 \right) a_1^1$$

Defiendo, entonces, el término  $\delta$  para la segunda neurona oculta como  $\delta_2^2(n) = f'(w_{12}a_1^1 + w_{22}a_2^1 + u_2^2)v_2\delta^3(n)$ , se obtiene que:

$$w_{12}(n) = w_{12}(n-1) + \alpha\delta_2^2(n)a_1^1(n)$$

Por tanto, en la modificación del parámetro  $w_{12}$  interviene la activación de la primera neurona de entrada  $a_1^1 = x_1$  -neurona de la que parte la conexión- y el término  $\delta$  de la segunda neurona de la capa oculta -neurona a la que llega la conexión-. Por extensión, el resto de los parámetros de esta capa se modifican de acuerdo con las siguiente leyes:

$$w_{11}(n) = w_{11}(n-1) + \alpha\delta_1^2(n)a_1^1(n)$$

$$w_{21}(n) = w_{21}(n-1) + \alpha\delta_1^2(n)a_2^1(n)$$

$$w_{22}(n) = w_{22}(n-1) + \alpha\delta_2^2(n)a_2^1(n)$$

$$u_1^2(n) = u_1^2(n-1) + \alpha\delta_1^2(n)$$

$$u_2^2(n) = u_2^2(n-1) + \alpha\delta_2^2(n)$$

donde:

$$\delta_1^2(n) = f' \left( w_{11}a_1^1 + w_{21}a_2^1 + u_2^2 \right) v_1\delta^3(n)$$

$$\delta_2^2(n) = f' \left( w_{12}a_1^1 + w_{22}a_2^1 + u_2^2 \right) v_2\delta^3(n)$$

### 3.3.2. Resumen de la regla delta generalizada

La regla delta generalizada viene descrita por las Ecuaciones anteriormente obtenidas (3.16, 3.17, 3.26 y 3.28) y, como su nombre indica, es una generalización de la regla delta vista en el capítulo anterior. Esta generalización es necesaria debido a la presencia de neuronas ocultas en la red, para las cuales no se conoce el error cometido. Como se ha visto anteriormente, esto se resuelve mediante la aplicación de la regla de la cadena, la cual actúa del siguiente modo:

Cada neurona de salida distribuye hacia atrás su error o valor  $\delta$  a todas las neuronas ocultas que se conectan a ella, ponderado por el valor de la conexión. De este modo, cada neurona oculta recibe un cierto error o valor  $\delta$  de cada neurona de salida, y la suma de estas cantidades es el término  $\delta$  asociado a la neurona oculta. Dichos valores permiten, a su vez, obtener los valores  $\delta$  de las neuronas ocultas de la capa anterior y así sucesivamente hasta llegar a la primera capa oculta. De ahí viene el nombre de algoritmo de RETROPROPAGACIÓN, pues los errores para las neuronas de la red se retropropagan hacia todas las neuronas de la capa anterior.

A continuación, se resumen las expresiones que definen la regla delta generalizada cuando se utiliza función de activación sigmoidal:

- **Pesos de la capa oculta  $C - 1$  a la capa de salida y umbrales de la capa de salida**

Pesos:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha\delta_i^C(n)a_j^{C-1}(n)$$

para  $j = 1, 2, \dots, n_{C-1}$  y  $i = 1, 2, \dots, n_C$

Umbrales:

$$u_i^C(n) = u_i^C(n-1) + \alpha\delta_i^C(n) \text{ para } i = 1, 2, \dots, n_C$$

donde:

$$\delta_i^C(n) = (s_i(n) - y_i(n))y_i(n)(1 - y_i(n))$$

- **Pesos de la capa  $c$  a la capa  $c + 1$  y umbrales de las neuronas de la capa  $c + 1$  para  $c = 1, 2, \dots, C - 2$**

Pesos:

$$w_{kj}^c(n) = w_{kj}^c(n-1) + \alpha\delta_j^{c+1}(n)a_k^c(n) \quad (3.26)$$

para  $k = 1, 2, \dots, n_c$ ,  $j = 1, 2, \dots, n_{c+1}$  y  $c = 1, 2, \dots, C - 2$

Umbrales:

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \alpha\delta_j^{c+1}(n)$$

para  $j = 1, 2, \dots, n_{c+1}$  y  $c = 1, 2, \dots, C - 2$

donde:

$$\delta_j^{c+1}(n) = a_j^c(n)(1 - a_j^c(n)) \sum_{i=1}^{n_{c+1}} \delta_i^{c+2}(n)w_{ji}^c$$

### 3.3.3. Razón de aprendizaje. Inclusión del momento en la ley de aprendizaje

El cambio en un peso de la red es proporcional al gradiente del error como se refleja en la Ecuación (3.10). Dicha proporcionalidad viene dada por el parámetro  $\alpha$ , también llamado razón o tasa de aprendizaje. Este parámetro es el encargado de controlar cuánto se desplazan los pesos de la red en la superficie del error siguiendo la dirección negativa del gradiente. Determina, por tanto, la magnitud de dicho desplazamiento, influyendo así en la velocidad de convergencia del algoritmo.

Valores altos de la razón de aprendizaje, en principio, podrían favorecer una convergencia más rápida, pues permiten avanzar rápidamente en la superficie del error. Sin embargo, razones de aprendizaje altas pueden tener consecuencias negativas en el aprendizaje, como que el método se salte un mínimo o incluso que el método oscile alrededor del mínimo. Valores pequeños de razones de aprendizaje podrían evitar estos problemas, aunque a costa de una convergencia más lenta del algoritmo de aprendizaje, pues la magnitud de los desplazamientos en la superficie del error es más pequeña.

Un método simple para evitar inestabilidad en el algoritmo de aprendizaje debido a la razón de aprendizaje es modificar la ley de aprendizaje dada por la Ecuación (3.10) mediante la inclusión de un segundo término, llamado *momento*, obteniendo así la siguiente ley:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w} + \eta \Delta w(n-1) \quad (3.35)$$

donde  $\Delta w(n-1) = w(n-1) - w(n-2)$  es el incremento que sufrió el parámetro  $w$  en la iteración anterior y  $\eta$  es un número positivo que controla la importancia asignada al incremento anterior.

Dicha regla fue propuesta por [Rumelhart et al., 1986a] y conserva las propiedades de la regla anterior (Ecuación 3.10) en el sentido de que modifica los parámetros de la red con el objetivo de minimizar la función error  $E$  (Ecuación 3.8). Sin embargo, el nuevo término,  $\eta \Delta w(n-1)$ , incorpora en el método cierta inercia, haciendo que la modificación actual del parámetro dependa de la dirección de la modificación anterior, lo cual puede evitar oscilaciones.

Aplicando sucesivamente la Ecuación (3.35) al término  $\Delta w(n-1)$ , se obtiene que:

$$\begin{aligned} \Delta w(n-1) &= w(n-1) - w(n-2) = -\alpha \frac{\partial e(n-1)}{\partial w} + \eta \Delta w(n-2) = \\ &= \dots = -\alpha \sum_{t=0}^{n-1} \eta^{n-1-t} \frac{\partial e(t)}{\partial w} \end{aligned}$$

Por tanto, la ley dada por la Ecuación (3.35) puede escribirse de la forma:

$$w(n) = w(n-1) - \alpha \sum_{t=0}^n \eta^{n-t} \frac{\partial e(t)}{\partial w} \quad (3.36)$$

En la expresión obtenida anteriormente se observa que el cambio actual de un parámetro viene dado por la suma de los gradientes del error para todas las iteraciones anteriores. Por tanto, cuando la derivada parcial del error respecto al peso tiene signos opuestos en iteraciones consecutivas, la suma puede contrarrestar estos cambios de signos, y, de este modo, procurar un cambio más suave en el peso, lo cual conduce a un método más estable, sin oscilaciones bruscas. Por otra parte, si la derivada parcial del error respecto al peso tiene el mismo signo en iteraciones consecutivas, la utilización del momento procura un cambio mayor en el peso, acelerando así la convergencia del algoritmo.

### 3.4. Proceso de aprendizaje del Perceptrón multicapa

Como se ha comentado en el apartado anterior, el objetivo del aprendizaje o entrenamiento del PERCEPTRON multicapa es ajustar los parámetros de la red - pesos y umbrales- con el fin de que las entradas presentadas produzcan las salidas deseadas, es decir, con el fin de minimizar la función error  $E$  (Ecuación 3.8). En esta sección, se van a detallar los pasos que involucra el proceso completo de aprendizaje del PERCEPTRON multicapa.

Sea  $\{(X(n), S(n)), n = 1, \dots, N\}$  el conjunto de muestras o patrones que representan el problema a resolver, donde  $X(n) = (x_1(n), \dots, x_{n_1}(n))$  son los patrones de entrada a la red,  $S(n) = (s_1(n), \dots, s_{n_C}(n))$  son las salidas deseadas para dichas entradas y  $N$  es el número de patrones disponibles. Generalmente, es frecuente encontrar los patrones de entrada y salida normalizados o escalados mediante una transformación lineal en los intervalos  $[0, 1]$  o  $[-1, 1]$  dependiendo de la función de activación empleada, sigmoidal o tangente hiperbólica, respectivamente. Es necesario señalar, sin embargo, que esta transformación de los patrones no es una condición necesaria para realizar el aprendizaje de la red, sino que los datos pueden presentarse a la red sin sufrir dicha normalización. Sólo será necesario tener en cuenta que, en el caso de que los patrones de salida se utilicen sin escalar, la función de activación de las neuronas de salida de la red debe ser la identidad, pues si se utilizan funciones de activación sigmoidales, las salidas de la red siempre estarán en los rangos de valores  $[0, 1]$  o  $[-1, 1]$  y, por tanto, nunca podrán aproximarse a la salida deseada.

Los pasos que componen el proceso de aprendizaje del PERCEPTRON multicapa son los siguientes:

Paso 1 Se inicializan los pesos y umbrales de la red. Generalmente, esta inicialización es aleatoria y con valores alrededor del cero.

Paso 2 Se toma un patrón  $n$  del conjunto de entrenamiento,  $(X(n), S(n))$ , y se propaga hacia la salida de la red el vector de entrada  $X(n)$  utilizando las Ecuaciones (3.1), (3.2) y (3.3), obteniéndose así la respuesta de la red para dicho vector de entrada,  $Y(n)$ .

Paso 3 Se evalúa el error cuadrático cometido por la red para el patrón  $n$  utilizando la Ecuación (3.9).

Paso 4 Se aplica la regla delta generalizada para modificar los pesos y umbrales de la red. Para ello se siguen los siguiente pasos:

- 4.1 Se calculan los valores  $\delta$  para todas las neuronas de la capa de salida utilizando la Ecuación (3.14).
- 4.2 Se calculan los valores  $\delta$  para el resto de las neuronas de la red utilizando la Ecuación (3.27) empezando desde la última capa oculta y retropropagando dichos valores hacia la capa de entrada.
- 4.3 Se modifican pesos y umbrales de la red siguiendo las Ecuaciones (3.16) y (3.17) para los pesos y umbrales de la capa de salida y (3.26) y (3.28) para el resto de los parámetros de la red.

Paso 5 Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento, completando así una iteración o *ciclo de aprendizaje*.

Paso 6 Se evalúa el error total  $E$  (Ecuación 3.8) cometido por la red. Dicho error también recibe el nombre de error de entrenamiento, pues se calcula utilizando los patrones de entrenamiento.

Paso 7 Se repiten los pasos 2, 3, 4, 5 y 6 hasta alcanzar un mínimo del error de entrenamiento, para lo cual se realizan  $m$  ciclos de aprendizaje.

Una idea más intuitiva del proceso de aprendizaje del PERCEPTRON multicapa puede resumirse de la siguiente forma:

Partiendo de un punto aleatorio  $W(0)$  del espacio  $\mathbf{R}^{n_w}$ , donde  $n_w$  es el número de parámetros de la red -pesos más umbrales-, el proceso de aprendizaje desplaza el vector de parámetros  $W(n-1)$  en el espacio  $\mathbf{R}^{n_w}$  siguiendo la dirección negativa del gradiente del error en dicho punto, alcanzando así un nuevo punto en dicho espacio,  $W(n)$ , que estará más próximo al mínimo de la función error que el anterior. El proceso continúa hasta que se encuentre un mínimo de la función error  $E$ , lo cual sucede cuando  $\frac{\partial E}{\partial w} \approx 0$ . En este momento, y de acuerdo con la Ecuación (3.10), los parámetros dejan de sufrir cambios significativos de una iteración a otra y el proceso de aprendizaje finaliza.

Por tanto, y desde un punto de vista teórico, el proceso de aprendizaje del PERCEPTRON multicapa debe finalizar cuando  $\frac{\partial E}{\partial w} \approx 0$ , momento en el que los parámetros de la red no cambian de una iteración a otra. Sin embargo, desde un punto de vista práctico y a la hora de implementar el proceso de aprendizaje del PERCEPTRON multicapa, se suele fijar un número de ciclos de aprendizaje, de modo que cuando se alcanza dicho número, se detiene el aprendizaje. En este punto se analiza si es necesario realizar más ciclos de aprendizaje o si

son suficientes los establecidos a priori. Este análisis se basa únicamente en observar si el error cometido por la red se mantiene prácticamente constante de una iteración a otra o si, por el contrario, el error sigue decreciendo. Así, por ejemplo, si se representa el error de entrenamiento cometido por la red en función del número de ciclos, en el caso de la situación presentada en la Figura 3.4(a), sería necesario realizar más ciclos de aprendizaje, mientras que en el caso de la Figura 3.4(b), es posible detener el proceso de aprendizaje de la red.

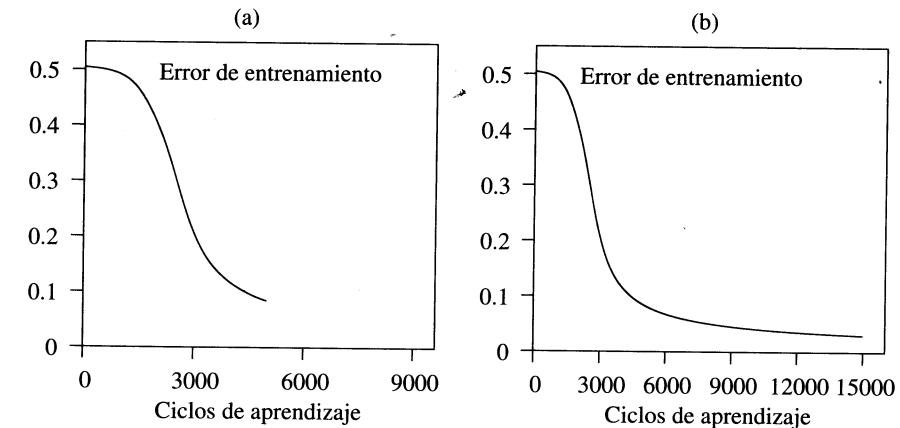


Figura 3.4: Evolución del error a lo largo del proceso de aprendizaje

Finalmente, se debe señalar que el error de entrenamiento se estabiliza, precisamente, cuando las salidas de la red no cambian de una iteración a otra, lo cual viene provocado porque los parámetros de la red sufren cambios insignificantes de una iteración a otra. Como se comentó anteriormente, esto es debido a que se ha localizado un mínimo de la función error, es decir que  $\frac{\partial E}{\partial w} \approx 0$ .

El proceso de aprendizaje descrito anteriormente es el más comúnmente usado, aunque existe una variante, conocida con el nombre de proceso batch. Básicamente, este proceso se diferencia del anterior en que los parámetros de la red se modifican una vez que todos los patrones de entrenamiento han sido presentados a la red, y no para cada patrón de entrenamiento. Ambos mecanismos son equivalentes, aunque el más usado e implementado en los simuladores de redes de neuronas es el proceso de aprendizaje descrito anteriormente, también conocido con el nombre de proceso continuo.

### 3.4.1. Capacidad de generalización

A la hora de evaluar el comportamiento de una red de neuronas, y en particular del PERCEPTRON multicapa, no sólo es importante saber si la red ha aprendido con éxito los patrones utilizados durante el aprendizaje, sino que es imprescindible, también, conocer el comportamiento de la red ante patrones que no se han utilizado durante el entrenamiento. Es decir, de nada sirve disponer de

una red que haya aprendido correctamente los patrones de entrenamiento y que no responda adecuadamente ante patrones nuevos. Es necesario que durante el proceso de aprendizaje la red extraiga las características de las muestras, para poder así responder correctamente a patrones diferentes. Esto se conoce como la capacidad de la red para generalizar las características presentes en el conjunto de muestras o capacidad de generalización de la red.

La Figura 3.5 muestra cómo la generalización podría ser llevada a cabo por una red. Los puntos etiquetados con cruces representan los patrones de entrenamiento, la línea punteada la función a aproximar y la línea sólida representa la salida que proporciona el PERCEPTRON multicapa una vez entrenada. En la Figura 3.5(a), la red ha alcanzado un buen nivel de generalización, mientras que en la Figura 3.5(b) se observa que la red ha memorizado los patrones de entrenamiento y no ha conseguido una buena generalización.

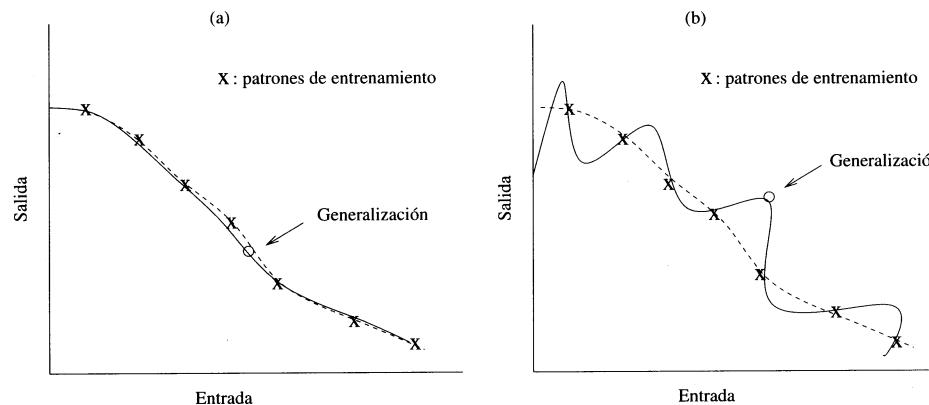


Figura 3.5: Generalización: (a) Buenas propiedades; (b) Escasas propiedades

Por tanto, cuando se realiza el proceso de aprendizaje de la red es muy importante, e incluso imprescindible, evaluar la capacidad de generalización. Para ello, es necesario disponer de dos conjuntos de muestras o patrones; uno para entrenar la red y modificar sus pesos y umbrales -*conjunto de entrenamiento*, y otro para medir la capacidad de la red para responder correctamente ante patrones que no han sido utilizados durante el entrenamiento -*conjunto de validación o test*. Estos conjuntos se obtienen de las muestras disponibles sobre el problema y es conveniente que la separación sea aleatoria, con el fin de tener conjuntos lo más representativos posible, tanto de entrenamiento como de validación.

A veces, y dependiendo de las características de los conjuntos de entrenamiento y validación, un entrenamiento riguroso podría anular la capacidad de generalización de la red. Por tanto, en ocasiones puede ser conveniente exigir un menor aprendizaje de la red sobre los patrones de entrenamiento, con el objetivo de obtener mejores propiedades de generalización. Con este propósito, es

necesario evaluar el error que comete la red sobre el conjunto de patrones de validación, no sólo cuando el proceso de aprendizaje ha concluido, sino también durante el proceso de aprendizaje. Cada cierto número de ciclos, se debe presentar a la red el conjunto de patrones de validación y calcular el error cometido por la red sobre dicho conjunto.

Al igual que se analiza la evolución del error de entrenamiento a lo largo de los ciclos de aprendizaje, se debe analizar también la evolución del error de validación. Al observar la evolución de ambos errores, se pueden encontrar las siguientes situaciones:

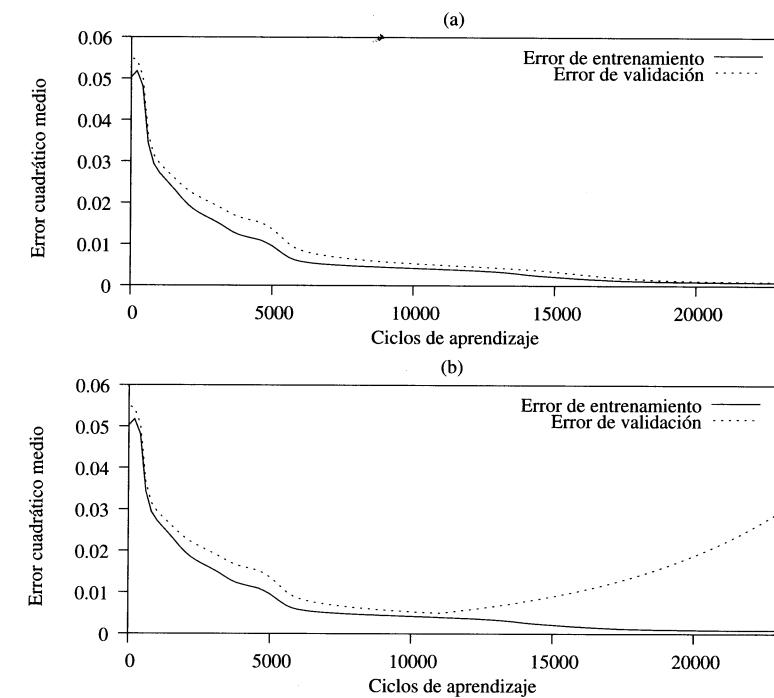


Figura 3.6: Evolución de los errores de entrenamiento y validación a lo largo del proceso de aprendizaje

- Ambos errores, de entrenamiento y validación, permanecen estables después de un cierto número de ciclos (ver Figura 3.6(a)). En este caso, se puede decir que el aprendizaje ha acabado con éxito, pues la red ha sido capaz de extraer las características del problema, alcanzando un buen nivel de generalización.
- A partir de un cierto número de ciclos, el error de validación comienza a aumentar (ver Figura 3.6(b)). En este caso, se puede decir que el número de ciclos realizado (23000) es adecuado para encontrar un mínimo del error.

de entrenamiento, pero a costa de perder propiedades de generalización de la red. Por tanto, no se puede afirmar que el aprendizaje haya acabado con éxito, sino que hubiera sido conveniente detener el proceso de aprendizaje en el momento en el que el error de validación comienza a crecer, para poder disponer así de una red con mejor capacidad de generalización. En estas situaciones se suele decir que se ha producido sobreaprendizaje en la red.

El sobreaprendizaje en un PERCEPTRON multicapa ocurre, por tanto, cuando la red ha aproximado correctamente los patrones de aprendizaje, pero no es capaz de responder adecuadamente ante los patrones de validación -por ejemplo, el caso que se observa en la Figura 3.6(b)). Este hecho puede producirse, como se ha comentado anteriormente, debido a un número elevado de ciclos de aprendizaje. Sin embargo, no es la única causa de sobreaprendizaje. En ocasiones, es producido por la utilización de demasiadas neuronas ocultas en la red. Un número excesivo de neuronas ocultas puede conducir a una escasa capacidad de generalización de la red. En estos casos, la red tiende a ajustar con mucha exactitud los patrones de entrenamiento, lo cual evita que la red extraiga la tendencia o las características del conjunto de entrenamiento. Particularmente, en problemas en los que las muestras poseen ruido, la utilización de muchas neuronas ocultas hace que la red se ajuste al ruido de los patrones, impidiendo así la generalización.

### 3.4.2. Deficiencias del algoritmo de aprendizaje

A pesar del éxito del algoritmo de RETROPROPAGACIÓN para entrenar el PERCEPTRON multicapa, este algoritmo posee también una serie de deficiencias, las cuales son analizadas a continuación.

- **Mínimos locales**

La superficie que define el error  $E$  (Ecuación 3.8) en función de los parámetros de la red es compleja y llena de valles y colinas. Debido a la utilización del método del gradiente para encontrar un mínimo de esta función error, se corre el riesgo de que el proceso de minimización finalice en un mínimo local [Minsky and Papert, 1988]. Se ha mencionado anteriormente que el proceso de adaptar los pesos finaliza cuando  $\frac{\partial E}{\partial w} \approx 0$ , lo cual no garantiza que el mínimo alcanzado sea un mínimo global, sino simplemente que se ha alcanzado un mínimo, que podría ser local. Claramente, es indeseable que el proceso de aprendizaje de la red finalice en un mínimo local, sobre todo si dicho mínimo está localizado lejos del mínimo global.

Cuando el proceso de aprendizaje cae en un mínimo local, una posible vía para evitar dicho problema es aumentar el número de neuronas ocultas en la red. En ocasiones, se considera que el proceso cae en un mínimo local debido a que la red posee un escaso poder de representación interna, de manera que no es capaz de distinguir patrones diferentes, proporcionando

las mismas salidas para estos patrones. Esto podría resolverse aumentando el número de neuronas ocultas, de manera que la red posea un mayor número de parámetros libres y pueda conseguir una mayor representabilidad del problema.

Existen otras aproximaciones para evitar el problema de los mínimos locales, como utilizar razón de aprendizaje decreciente a lo largo del proceso de aprendizaje, partir de otras inicializaciones de los parámetros de la red o añadir ruido al método de descenso del gradiente, entre otras.

- **Parálisis**

El fenómeno de la parálisis, también conocido como saturación, en el PERCEPTRON mutlicapa se produce cuando la entrada total a una neurona de la red toma valores muy altos, tanto positivos como negativos. Debido a que las funciones de activación poseen dos asíntotas horizontales (ver Figura 3.2), si la entrada a una neurona alcanza un valor alto, ésta se satura, y alcanza un valor de activación máximo o mínimo.

En las Ecuaciones (3.31) y (3.32) se observa que el ajuste de los parámetros de la red es proporcional a  $y_j(n)(1 - y_j(n))$ , siendo  $y_j(n)$  la activación de la neurona  $j$  de salida. Si una neurona de salida está saturada, dicho valor es próximo a cero, pues  $y_j(n) = 0$  o  $y_j(n) = 1$ . Por tanto, cuando ocurre el fenómeno de parálisis en un PERCEPTRON multicapa, los parámetros de la red permanecen invariables y, como consecuencia, la suma de los errores locales permanece constante por un periodo largo de tiempo [Lee et al., 1991]. Aunque esta situación pueda confundirse con la presencia de un mínimo local, pues el error es invariable, en este caso es posible que después de un cierto tiempo, el error comience de nuevo a decrecer.

El fenómeno de la parálisis en el PERCEPTRON multicapa ocurre, fundamentalmente, cuando los parámetros de la red toman valores muy altos. Por tanto, para evitar este problema es conveniente partir de valores iniciales aleatorios próximos a cero.

## 3.5. Ejemplo de funcionamiento del perceptron multicapa

En esta sección se presenta un ejemplo que muestra, por un lado, los pasos a seguir para la resolución de un problema utilizando el PERCEPTRON multicapa, y por otro lado, la influencia que ciertos parámetros podrían tener en el comportamiento de la red.

Como caso práctico se plantea un problema de aproximación de funciones. Supóngase que se desea construir un PERCEPTRON multicapa para aproximar la siguiente función **definida por partes** (ver Figura 3.7):

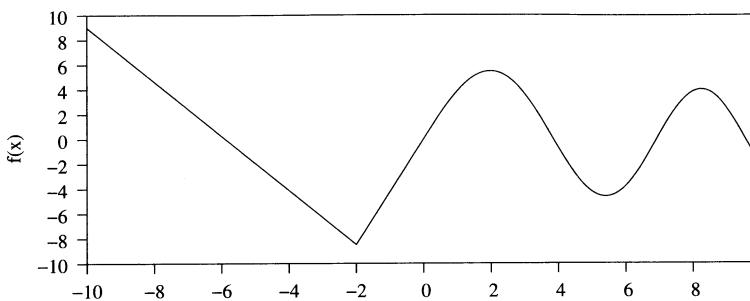


Figura 3.7: Función definida a trozos

$$f(x) = \begin{cases} -2,186x - 12,864 & \text{si } -10 \leq x < -2 \\ 4,246x & \text{si } -2 \leq x < 0 \\ 10 \exp(-0,05x-0,5) \operatorname{sen} [(0,03x+0,7)x] & \text{si } 0 \leq x < 10 \end{cases} \quad (3.37)$$

Los pasos a seguir para la aproximación de dicha función utilizando el PERCEPTRON multicapa se detallan a continuación.

### 1. Conjunto de muestras o ejemplos sobre el problema

En primer lugar, y como en cualquier aplicación de redes de neuronas, es necesario disponer de un conjunto de muestras o variables numéricas que representen el problema. En la mayor parte de las aplicaciones reales, el conjunto de muestras se corresponden con medidas tomadas por un experto y que representan el comportamiento del problema a resolver. En nuestro caso práctico, y puesto que se conoce la expresión analítica de la función, dicho conjunto se obtiene variando la variable  $x$  en el intervalo  $[-10, 10]$  y obteniendo el valor de la función  $f(x)$  en dicho intervalo. Concretamente, se han obtenido 200 muestras uniformemente distribuidas en el intervalo  $[-10, 10]$ .

Aunque no es estrictamente necesario, puede ser recomendable realizar una transformación de los datos de entrada y salida para que estén normalizados o escalados en el intervalo  $[0, 1]$ , como se ha comentado en la Sección 3.4. De manera genérica, si  $x$  es una variable que representa el problema, y  $\max_x$  y  $\min_x$  son, respectivamente, los valores máximos y mínimos que puede tomar dicha variable, la normalización o escalado se puede realizar utilizando la siguiente transformación de la variable  $x$ :

$$\operatorname{tran}(x) = \frac{1}{\max_x - \min_x} (x - \min_x) \quad (3.38)$$

### 2. Extracción del conjunto de entrenamiento y validación

Del conjunto de muestras disponibles sobre el problema, se extraen los conjuntos de entrenamiento y validación o test. Es conveniente realizar dicha separación de manera aleatoria, para que los conjuntos no tengan ningún sesgo de información sobre el problema. En el caso práctico que se está tratando, la separación se realiza de manera aleatoria, tomando un 60 % de los datos para entrenar y un 40 % para medir la capacidad de generalización de la red.

### 3. Diseño de la arquitectura del perceptron multicapa

En este caso, el PERCEPTRON multicapa tiene una neurona de entrada y otra de salida. En una primera aproximación, se fija una única capa oculta con 10 neuronas ocultas. Posteriormente, dicho parámetro será modificado con el objetivo de conseguir la red más adecuada.

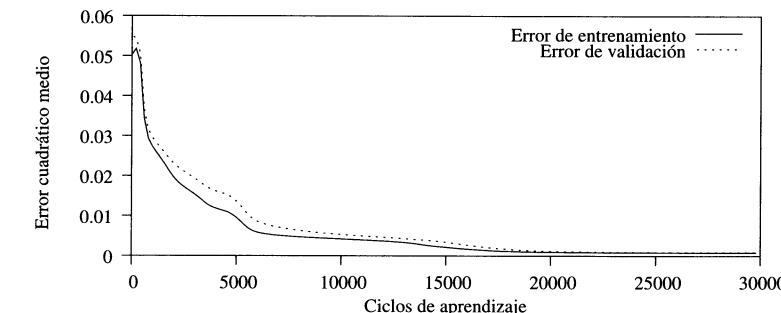


Figura 3.8: Evolución de los errores de entrenamiento y validación

### 4. Proceso de aprendizaje del perceptron multicapa

Una vez definida la arquitectura de la red, se procede a realizar el proceso de aprendizaje o entrenamiento de la red. Para ello, es necesario, en primer lugar, fijar los parámetros que intervienen en dicho proceso: la razón de aprendizaje y el número de ciclos de aprendizaje. Ambos parámetros siempre van a depender del problema a resolver y será necesario realizar varias simulaciones previas para fijarlos de acuerdo con el problema. En nuestro caso se fijan a:

- Razón de aprendizaje:  $\alpha = 0,2$
- Ciclos de aprendizaje: 30000

En la Figura 3.8 se muestra la evolución de los errores cuadráticos para el conjunto de entrenamiento y validación a lo largo de los ciclos de aprendizaje. Se observa que dichos errores van decreciendo y llega un momento en el que se estabilizan, alrededor de los 20000 ciclos de aprendizaje.

En la Figura 3.9 se muestran las aproximaciones de los patrones de validación proporcionadas por la red en diferentes momentos del aprendizaje:

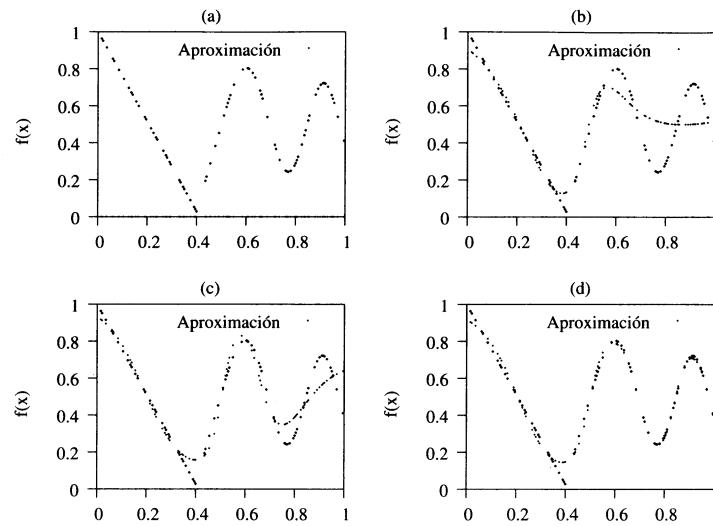


Figura 3.9: Aproximaciones de la función definida a trozos. (a) Antes de comenzar el aprendizaje; (b) Después de 5000 ciclos; (c) Después de 10000 ciclos; (d) Después de 30000 ciclos

antes de comenzar a realizar el aprendizaje -con pesos y umbrales iniciales aleatorios en el intervalo  $[-1, 1]$ - y después de 5000, 10000 y 30000 ciclos de aprendizaje, respectivamente. Se observa cómo, a través del proceso de aprendizaje, la red va aproximando las relaciones entre la salida y la entrada, extrayendo el comportamiento de la función. En este caso, el PERCEPTRON multicapa encuentra dificultades para aproximar correctamente el pico de la función.

A este punto, surge una serie de cuestiones acerca de la optimización de la red y del proceso de aprendizaje. Éstas son:

- **¿Cómo afecta el número de neuronas ocultas de la red a la resolución de este problema?** es decir, ¿puede conseguirse el mismo nivel de aproximación disminuyendo el número de neuronas ocultas?, o ¿puede mejorarse el nivel de aproximación incorporando un mayor número de neuronas ocultas en la red?. Para responder a estas preguntas, se lanzan nuevas simulaciones, utilizando 2, 20 y 50 neuronas ocultas. En la Figura 3.10 se muestra la evolución, a lo largo de los 30000 ciclos de aprendizaje, de los errores de validación para las redes de 2, 10, 20 y 50 neuronas ocultas. Se observa que, en este caso, dos neuronas no son suficientes para aproximar la función, mientras que utilizar 10 o 20 neuronas ocultas no afecta a los resultados. Con 50 neuronas ocultas, la red necesita más ciclos de aprendizaje para

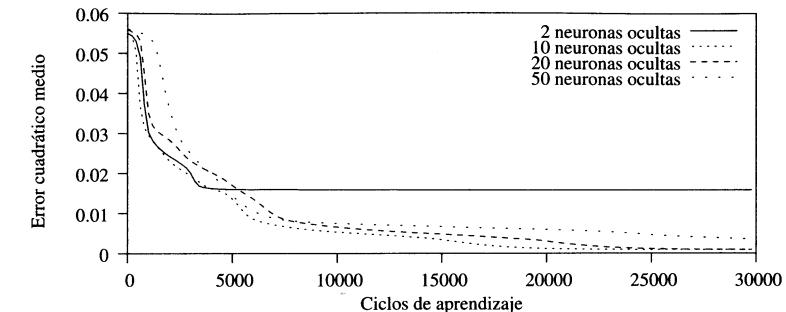


Figura 3.10: Evolución del error variando el número de neuronas ocultas

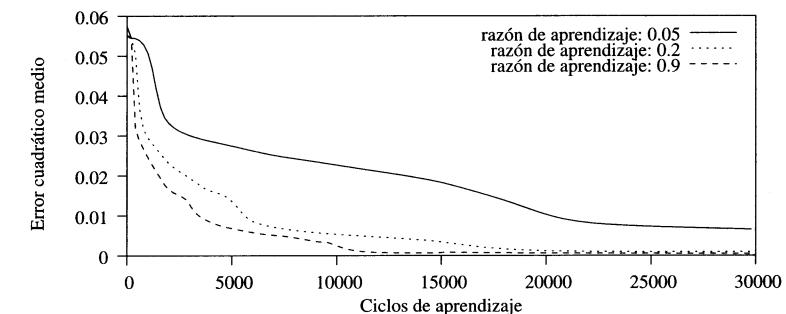


Figura 3.11: Evolución del error variando la razón de aprendizaje

conseguir la estabilización del error, lo cual es debido a que el número de parámetros de la red ha aumentado considerablemente. Ante esta situación, se puede decir que la red con 10 neuronas ocultas es adecuada para aproximar la función definida por partes (Ecuación 3.37). Es necesario hacer hincapié que estas conclusiones son válidas únicamente en este dominio, pues el número de neuronas ocultas más adecuado depende del problema a resolver.

- **¿Cómo afecta la razón de aprendizaje al proceso de entrenamiento?** es decir, ¿se puede conseguir el mismo error en un menor número de ciclos?. Para ello se lanzan dos nuevas simulaciones para la red de 10 neuronas ocultas; la primera utilizando razón de aprendizaje  $\alpha = 0,05$  y la segunda con  $\alpha = 0,9$ . En la Figura 3.11 se muestra la evolución de los errores de validación para las diferentes razones de aprendizaje. Se observa que razones de aprendizaje pequeñas retrasan la convergencia, mientras que razones de aprendizaje mayores aceleran la convergencia del algoritmo. Por tanto, para la resolución de nuestro problema sería aconsejable utilizar razón de aprendizaje mayor que 0,2. Al igual que en el punto anterior, estos resultados

dependen del problema, y una razón de aprendizaje adecuada para un problema podría ser pequeña o demasiado grande para otro problema.

En el cuadro 3.3 se muestran los errores sobre el conjunto de entrenamiento y validación para las simulaciones realizadas, variando el número de neuronas ocultas y la razón de aprendizaje. Se observa que aumentar el número de neuronas ocultas no mejora la capacidad de la red para aproximar la función definida por partes, aunque aumentar la razón de aprendizaje acelera la convergencia del algoritmo consiguiendo una mejor aproximación.

Cuadro 3.3: Errores cuadráticos obtenidos para las diferentes simulaciones

	Error de entrenamiento	Error de validación
10 ocultas $\alpha = 0,2$	0,000830	0,000904
10 ocultas $\alpha = 0,05$	0,005092	0,0065160
<b>10 ocultas <math>\alpha = 0,9</math></b>	<b>0,000245</b>	<b>0,000471</b>
2 ocultas $\alpha = 0,2$	0,011700	0,015746
20 ocultas $\alpha = 0,2$	0,000772	0,000899
50 ocultas $\alpha = 0,2$	0,002690	0,003503

Debido a que no existe un método automático general para determinar el número óptimo de neuronas ocultas o la razón de aprendizaje más adecuada, una posible metodología a seguir es la que se ha utilizado en esta sección: partir de ciertos valores y variarlos para analizar el comportamiento de los errores de entrenamiento y validación con vistas a obtener la red más adecuada.

##### 5. Utilización de la red entrenada

Una vez realizado el proceso de aprendizaje de diferentes arquitecturas de redes y elegida la más adecuada, generalmente, en términos de la red capaz de producir el menor error de validación -mayor capacidad de generalización- en el menor número de ciclos de aprendizaje, ésta puede utilizarse para aproximar la función dada por la Ecuación (3.37). Durante la utilización de la red, los nuevos patrones son presentados a la red y propagados hacia la salida, obteniendo así la respuesta de la red para los nuevos patrones.

## Capítulo 4

# REDES DE NEURONAS DE BASE RADIAL

### 4.1. Introducción

Las redes de neuronas de base radial son redes multicapa con conexiones hacia adelante, al igual que el PERCEPTRON multicapa tratado en el capítulo anterior. Las redes de base radial se caracterizan porque están formadas por una única capa oculta y cada neurona de esta capa posee un carácter local, en el sentido de que cada neurona oculta de la red se activa en una región diferente del espacio de patrones de entrada. Este carácter local viene dado por el uso de las llamadas funciones de base radial, generalmente la función gausiana, como funciones de activación. Las neuronas de la capa de salida de las redes de base radial simplemente realizan una combinación lineal de las activaciones de las neuronas ocultas.

La mayor contribución a la teoría, diseño y aplicaciones de las redes de neuronas de base radial se debe a Moody y Darken [Moody and Darken, 1989], Renals [Renals, 1989] y a Poggio y Girosi [Poggio and Girosi, 1990]. Uno de los objetivos iniciales de los autores era construir una red de neuronas que requiriese un menor tiempo de aprendizaje que el que necesita el PERCEPTRON multicapa y, de este modo, disponer de una red de neuronas que pudiera ser apropiada para aplicaciones en tiempo real. Esto se consiguió incorporando funciones de activaciones locales en las neuronas ocultas de la red, lo cual permitía que sólo unas pocas neuronas ocultas tuvieran que ser procesadas para nuevos patrones de entrada.

Al igual que el PERCEPTRON multicapa, las redes de neuronas de base radial son **aproximadores universales**, en el sentido de que pueden aproximar cualquier

función continua sobre un compacto de  $\mathbf{R}^n$ . Una demostración formal de este resultado fue realizada por Park y Sandberg [Park and Sandberg, 1991].

Las funciones de base radial definen hiperesferas o hiperelipses que dividen el espacio de entrada. Por tanto, cada neurona oculta de la red de base radial construye una aproximación local y no lineal en una determinada región de dicho espacio. Puesto que la salida de la red es combinación lineal de las funciones de base radial, las aproximaciones que construyen las redes de base radial son combinaciones lineales de múltiples funciones locales y no lineales. De este modo, se suele decir que las redes de base radial aproximan relaciones complejas mediante una colección de aproximaciones locales menos complejas, dividiendo el problema en varios subproblemas menos complejos. Esto hace que las aproximaciones construidas por las redes de base radial sean de naturaleza diferente a las aproximaciones globales y basadas en hiperplanos que construye el PERCEPTRON multicapa. Sin embargo, cada una de estas clases de aproximadores -redes de base radial y PERCEPTRON multicapa- tiene sus propias características, como se verá en este capítulo.

Las redes de neuronas de base radial han sido aplicadas a una gran variedad de problemas, aunque es necesario señalar que su aplicación no ha sido tan extendida como en el caso del PERCEPTRON multicapa. Sin embargo, se han utilizado en diferentes campos, como análisis de series temporales [Moody and Darken, 1989], [Kadirkamanathan et al., 1991], procesamiento de imágenes [Saha et al., 1991], diagnósticos médicos [Lowe and Webb, 1990], reconocimiento automático del habla [Niranjan and Fallside, 1990], etc.

## 4.2. Arquitectura de las redes de base radial

Las redes de neuronas de base radial están formadas por tres capas de neuronas: la capa de entrada, una única capa oculta y la capa de salida, como se muestra en la Figura 4.1. La capa de entrada la componen un conjunto de neuronas que reciben las señales del exterior, transmíténdolas a la siguiente capa sin realizar ningún procesado sobre dichas señales. Las neuronas de la capa oculta reciben las señales de la capa de entrada y realizan una transformación local y no lineal sobre dichas señales. Este carácter local es lo que las diferencia del PERCEPTRON multicapa, no sólo en cuanto a arquitectura, sino también en cuanto a comportamiento. Esta capa es la única que incluye componentes no lineales en las redes de base radial. Y, finalmente, la capa de salida que realiza una combinación lineal de las activaciones de las neuronas ocultas, que actúa además como salida de la red.

Las redes de neuronas de base radial son redes con conexiones hacia adelante, como se observa en la Figura 4.1, y estas conexiones se dirigen siempre de una capa a la siguiente capa. La red se caracteriza porque las conexiones de la capa de entrada a la capa oculta no llevan asociado ningún peso, mientras que, y como es habitual en el contexto de redes de neuronas, las conexiones de la capa oculta a la capa de salida sí llevan asociado un número real o peso de la conexión. En lo referente a los umbrales de las neuronas, en las redes de base radial únicamente

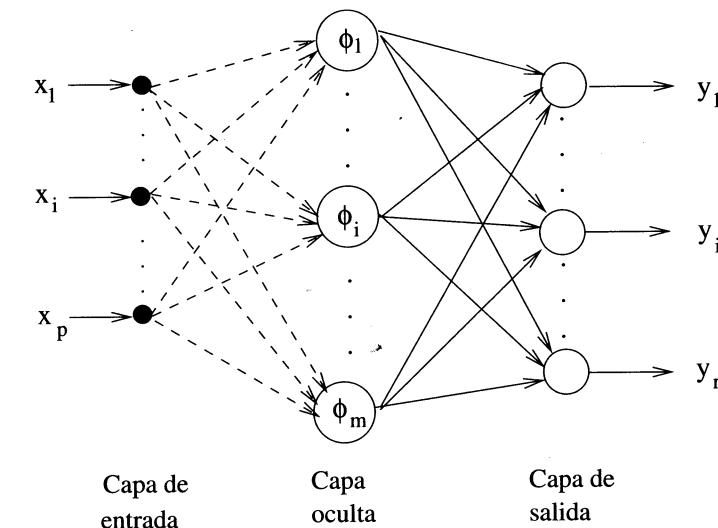


Figura 4.1: Arquitectura de la red de neuronas de base radial

las neuronas de salida poseen un umbral, que también se suele tratar -al igual que en el PERCEPTRON multicapa- como una conexión más de la neurona cuya entrada es constante e igual a 1.

Las redes de neuronas de base radial definen una relación no lineal entre las variables de entrada y las variables de salida de la red, propagando hacia la salida las señales o muestras recibidas en la entrada. A continuación, se presentan las expresiones para calcular las activaciones de las neuronas de las redes de base radial.

### 4.2.1. Activaciones de las neuronas de la red de base radial

Dada una red de neuronas de base radial con  $p$  neuronas en la capa de entrada,  $m$  neuronas en la capa oculta y  $r$  neuronas en la capa de salida, las activaciones de las neuronas de salida para el patrón de entrada  $n$ ,  $X(n) = (x_1(n), x_2(n), \dots, x_p(n))$ , denotadas como  $y_k(n)$ , vienen dadas por la siguiente expresión:

$$y_k(n) = \sum_{i=1}^m w_{ik} \phi_i(n) + u_k \text{ para } k = 1, 2, \dots, r \quad (4.1)$$

donde  $w_{ik}$  representa el peso de la conexión de la neurona oculta  $i$  a la neurona de salida  $k$ ,  $u_k$  es el umbral de la neurona de salida  $k$  y  $\phi_i(n)$  son las activaciones de las neuronas ocultas para el patrón de entrada  $X(n)$ . Se observa en la ecuación anterior que las neuronas de salida de la red utilizan la función de activación

identidad, realizando una transformación lineal de las activaciones de todas las neuronas ocultas.

Las funciones  $\phi_i$ , también conocidas como funciones de base radial, determinan las activaciones de las neuronas ocultas de la red en función del vector de entrada a la red  $X(n)$  y vienen dadas por la siguiente expresión:

$$\phi_i(n) = \phi\left(\frac{\|X(n) - C_i\|}{d_i}\right) \text{ para } i = 1, 2, \dots, m \quad (4.2)$$

donde  $\phi$  es una función de base radial;  $C_i = (c_{i1}, \dots, c_{ip})$  son vectores que representan los centros de la función de base radial;  $d_i$  son números reales que representan la desviación, anchura o dilatación de la función de base radial; y  $\|\cdot\|$  es la distancia euclídea del vector de entrada  $X(n)$  al centro  $C_i$ , definida como:

$$\|X(n) - C_i\| = \left( \sum_{j=1}^p (x_j(n) - c_{ij})^2 \right)^{\frac{1}{2}} \quad (4.3)$$

Por tanto, la activación de una neurona oculta en las redes de base radial depende de la distancia del patrón de entrada  $X(n)$  al centro  $C_i$  de la función de base radial. Estas funciones bases  $\phi$  poseen un carácter local, pues son funciones que alcanzan un nivel cercano al máximo de su recorrido cuando el patrón de entrada  $X(n)$  está próximo al centro de la neurona; a medida que el patrón se aleja del centro, el valor de función va tendiendo al valor mínimo de su recorrido (ver Figura 4.2).

La función de base radial  $\phi$  puede adoptar diferentes formas y expresiones (Figura 4.2), entre ellas:

- Función gausiana:

$$\phi(r) = e^{-\frac{r^2}{2}} \quad (4.4)$$

- Función inversa cuadrática:

$$\phi(r) = \frac{1}{1 + r^2} \quad (4.5)$$

- Función inversa multicuadrática:

$$\phi(r) = \frac{1}{\sqrt{1 + r^2}} \quad (4.6)$$

En el contexto de redes de neuronas de base radial, la más utilizada es la función gausiana [Moody and Darken, 1989]. Por tanto, la activación de las neuronas ocultas de las redes de base radial viene dada, generalmente, por la siguiente expresión:

$$\phi_i(n) = \exp^{-\frac{\|X(n) - C_i\|^2}{2d_i^2}} = \exp^{-\frac{\sum_{j=1}^p (x_j(n) - c_{ij})^2}{2d_i^2}} \text{ para } i = 1, 2, \dots, m \quad (4.7)$$

Las salidas de las redes de neuronas de base radial (Ecuación 4.1) son, por tanto, una combinación lineal de gaußianas, cada una de las cuales se activa para una determinada porción del espacio definido por los patrones de entrada.

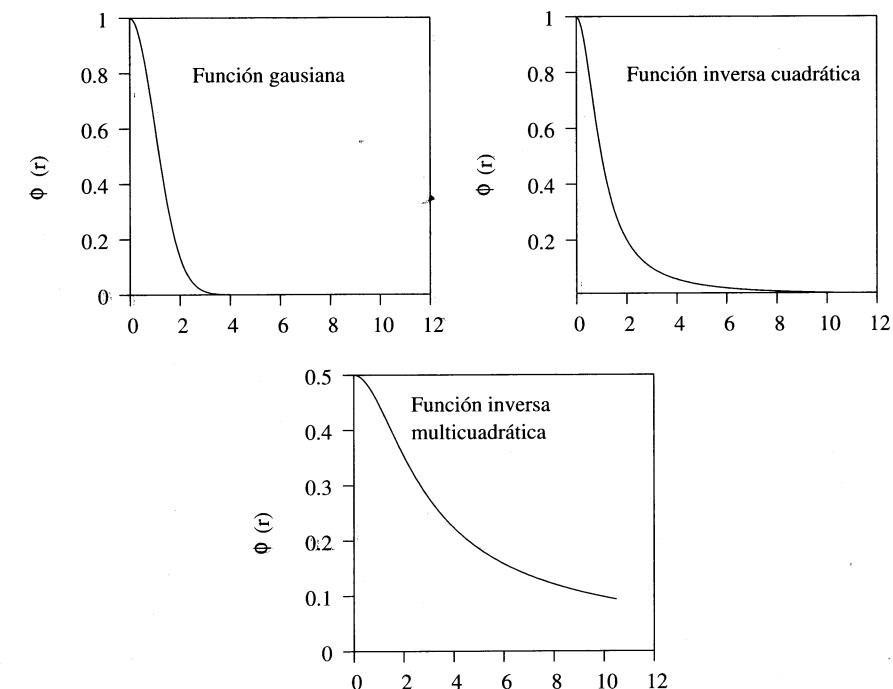


Figura 4.2: Funciones de base radial

### Cálculo de la salida de una red de neuronas de base radial con dos neuronas de entrada, dos ocultas y una salida

Sea la red de base radial que se muestra en la Figura 4.3. La ecuación para obtener la salida de la red para un patrón de entrada  $n$ ,  $X(n) = (x_1(n), x_2(n))$ , viene dada por:

$$y(n) = w_1\phi_1(n) + w_2\phi_2(n) + u$$

Si  $C_1 = (c_{11}, c_{12})$  y  $C_2 = (c_{21}, c_{22})$  son los centros de las funciones de base radial  $\phi_1$  y  $\phi_2$ , y  $d_1$  y  $d_2$  sus respectivas amplitudes, las activaciones de las neuronas ocultas toman la siguiente expresión:

$$\phi_1(n) = \exp^{-\frac{(x_1(n) - c_{11})^2 + (x_2(n) - c_{12})^2}{2d_1^2}}$$

$$\phi_2(n) = \exp^{-\frac{(x_1(n) - c_{21})^2 + (x_2(n) - c_{22})^2}{2d_2^2}}$$

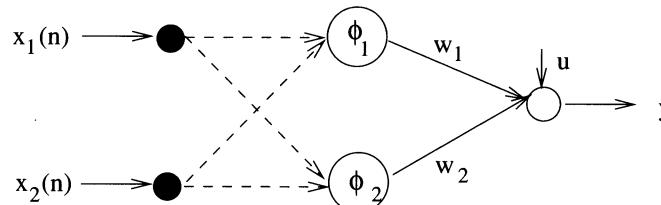


Figura 4.3: Ejemplo de arquitectura de una red de base radial

Así, si  $X(n) = (0,87, 0,49)$  y la red tiene como parámetros  $C_1 = (0,26, 0,43)$ ,  $C_2 = (0,78, 0,53)$ ,  $d_1 = 0,26$ ,  $d_2 = 0,26$ ,  $w_1 = 0,6$ ,  $w_2 = 0,57$  y  $u = 0$ , la salida de la red toma el valor  $y(n) = 0,58$ .  $\phi_1 \approx 0,06$   
 $\phi_2 \approx 0,93$

#### 4.2.2. Carácter local de las redes de base radial

Las funciones de base radial  $\phi$  (ver Figura 4.2) se caracterizan porque poseen un nivel máximo de activación para valores de entrada cercanos a cero y dicho nivel decrece a medida que la variable de entrada se aleja de dicho punto.

En la Ecuación (4.7) se observa que la activación de la neurona oculta  $i$  en una red de base radial viene dada por la función gaussiana de centro  $C_i$  y desviación  $d_i$ . Por tanto, si el patrón de entrada a la red  $X(n)$  está en la vecindad del centro  $C_i$ , la neurona oculta  $i$  alcanzará un valor alto de activación. A medida que el patrón  $X(n)$  se aleja del centro -dependiendo de la desviación- la activación de la neurona disminuye, y puede activarse otra neurona oculta de la red.

Por esta razón, se dice que las redes de neuronas de base radial son redes con carácter local, ya que, dado un patrón de entrada a la red, sólo aquellas neuronas ocultas cuyos centros estén en la vecindad de dicho patrón se van a activar; el resto de las neuronas ocultas permanecerán inactivas o con un menor nivel de activación. Esto no sucede cuando se utilizan funciones de activaciones sigmoidales, como en el caso del PERCEPTRON multicapa, pues éstas se activan en todo un rango de valores. Esto hace que, aunque ambas arquitecturas sean redes multicapa con conexiones hacia adelante, cada una de ellas posea sus propias características, así como sus ventajas e inconvenientes, las cuales serán analizadas en la Sección 4.5.

El carácter local de las redes de base radial se ilustra en la Figura 4.4. En dicha figura se muestran las activaciones de las neuronas ocultas de una red de base radial con una neurona de entrada y tres neuronas ocultas. Los puntos simbolizados mediante un rombo representan los patrones de entrada a la red y las curvas las activaciones de cada neurona oculta. Se observa que cada zona del espacio de entrada está representado por una neurona oculta, de manera que para ciertos patrones de entrada, la neurona oculta con mayor activación es diferente que para otro grupo de patrones de entrada. En el cuadro 4.1 se recogen los parámetros de dichas gausianas -centro y amplitud-, así como el

Cuadro 4.1: Parámetros y activaciones de las neuronas ocultas representadas en la Figura 4.4

	Centro	Amplitud	Activación para el patrón $x = 0,23$	Activación para el patrón $x = 0,59$
Neurona oculta 1	0,19	0,13	<b>0,953</b>	0,008
Neurona oculta 2	0,5	0,1	0,026 0,047	<b>0,666</b>
Neurona oculta 3	0,82	0,14	0,0	0,203 0,326

valor de dichas funciones para los patrones marcados en la gráfica. Estos valores muestran que para el patrón  $x = 0,23$  la neurona con mayor activación es la primera, mientras que para el patrón  $x = 0,59$  la mayor activación la alcanza la segunda neurona. De este modo, cada patrón está representado por una neurona oculta de la red.

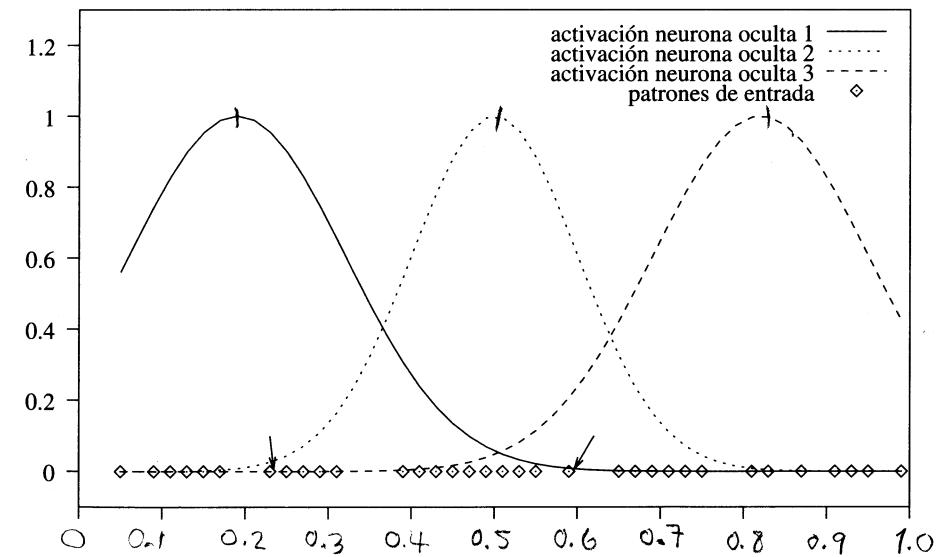


Figura 4.4: Activación local de las neuronas ocultas en una red de base radial

#### 4.2.3. Diseño de la arquitectura de las redes de base radial

El número de entradas y salidas en una red de base radial viene dado por el número de variables que definen el problema. Como ocurría cuando se utilizaba un PERCEPTRON multicapa, en algunas aplicaciones no hay lugar a duda sobre

dichas variables. Sin embargo, existen aplicaciones en las que pudiera ser necesario llevar a cabo un análisis de las variables más relevantes y significativas que definen el problema.

En lo que respecta al número de neuronas ocultas en la red, generalmente, se determina por prueba y error, variando el número de neuronas hasta conseguir una red capaz de resolver el problema, como en el caso del PERCEPTRON multicapa para determinar el número de capas ocultas y el número de neuronas en cada capa. No obstante, y a diferencia del PERCEPTRON multicapa, para las redes de base radial añadir o eliminar unas pocas neuronas ocultas podría influir significativamente en los resultados obtenidos por la red. Esto es debido a que cada neurona oculta representa una determinada región del espacio de entrada, pudiendo no estar adecuadamente representado dicho espacio, bien por la presencia de pocas clases, bien por la presencia de demasiadas neuronas ocultas en una misma zona, con su posterior consecuencia negativa en la aproximación de la red.

En los últimos años ha habido un interés creciente por el desarrollo de métodos que permitan determinar de modo automático el número de neuronas ocultas en las redes de base radial. Existen varias líneas de investigación abiertas en este campo. Una de ellas se basa en el uso de técnicas evolutivas, como algoritmos genéticos o estrategias evolutivas, que realizan una búsqueda en el espacio de todas las posibles arquitecturas de redes ([Gruau, 1995], [Whitehead and Choate, 1996]). Otros trabajos, sin embargo, se centran en el desarrollo de algoritmos incrementales, es decir, algoritmos que partiendo de una única neurona oculta van incorporando nuevas neuronas a medida que se reciben patrones de entrada que no están representados con las neuronas ocultas existentes ([Platt, 1991]). La mayor parte de estos métodos tratan no sólo de encontrar el número óptimo de neuronas, sino también los parámetros asociados a dichas neuronas, es decir, centros y amplitudes.

### 4.3. Aprendizaje de las redes de base radial

El proceso de aprendizaje implica la determinación de todos los parámetros que intervienen en la red de base radial. Éstos son: los centros y las desviaciones de las neuronas ocultas y los pesos de la capa oculta a la capa de salida, así como los umbrales de las neuronas de salida.

Debido a que las capas de neuronas en una red de base radial realizan tareas diferentes, es razonable separar el proceso de optimización de los parámetros de la capa oculta y los de la capa de salida mediante la utilización de diferentes técnicas. Así, para los parámetros de la capa oculta -centros y desviaciones- el proceso de aprendizaje debe estar guiado por una optimización en el espacio de patrones de entrada, pues cada una de las neuronas ocultas en la red de base radial va a representar una zona diferente del espacio de entrada. Sin embargo, para los parámetros de la capa de salida la optimización se debe realizar en base a las salidas que se desea obtener o salidas deseadas, ya que las redes de base radial se utilizan para aproximar relaciones entre el conjunto de variables

de entrada y salida que definen el problema. Por tanto, uno de los mecanismos más usados para el aprendizaje de las redes de base radial es el llamado método híbrido, que combina dos fases: una fase no supervisada para la determinación de los centros y otra supervisada para la determinación de los pesos y umbrales.

A pesar de que el método de aprendizaje híbrido es el más utilizado para realizar el aprendizaje de una red de base radial, pues conserva las propiedades locales de la red, existe otro mecanismo de aprendizaje, llamado método de aprendizaje totalmente supervisado. Como su nombre indica, realiza una adaptación supervisada de todos los parámetros de la red. A continuación, se va a describir en detalle cada uno de estos métodos de aprendizaje, así como sus propiedades y características.

#### 4.3.1. Método de aprendizaje híbrido

Como se ha comentado anteriormente, el método híbrido realiza el aprendizaje de las redes de base radial en dos fases:

- Fase no supervisada: determinación de los centros y amplitudes de las neuronas de la capa oculta.
- Fase supervisada: determinación de pesos y umbrales de la capa de salida.

El proceso de optimización en cada una de las fases se formula con un objetivo diferente y las técnicas para su resolución serán también, por tanto, diferentes.

##### Fase no supervisada

Puesto que las neuronas ocultas de las redes de base radial se caracterizan porque representan zonas diferentes del espacio de patrones de entrada, los centros y las desviaciones de las funciones de base radial deben ser determinados con este objetivo, es decir, con el objetivo de clasificar el espacio de entrada en diferentes clases. El representante de cada clase será el centro de la función de base radial y la desviación vendrá dada por la amplitud de cada clase.

##### ■ Determinación de los centros: Algoritmo de K-medias

Los centros de las funciones de base radial se determinan, por tanto, mediante un algoritmo de clasificación no supervisado que permita dividir el espacio de patrones de entrada en clases. El número de clases es el número de neuronas ocultas en la red de base radial. El método más utilizado es el algoritmo de K-medias, aunque es necesario destacar que cualquier algoritmo de clasificación no supervisado podría ser utilizado, como, por ejemplo, los mapas autoorganizados de Kohonen.

El algoritmo de K-medias ([Lloyd, 1982], [MacQueen, 1967]) es un algoritmo de clasificación no supervisado mediante el cual el espacio de patrones de entrada se divide en  $K$  clases o regiones. El representante de cada una de estas clases,  $C_i$ , será el centro de la neurona oculta  $i$ . Dichos centros se

determinan con el objetivo de minimizar las distancias euclídeas entre los patrones de entrada y el centro más cercano, es decir:

$$J = \sum_{i=1}^K \sum_{n=1}^N M_{in} \|X(n) - C_i\| \quad (4.8)$$

donde  $N$  es el número de patrones,  $\|\cdot\|$  es la distancia euclídea,  $X(n)$  es el patrón de entrada  $n$  y  $M_{in}$  es la función de pertenencia, que vale 1 si el centro  $C_i$  es el más cercano al patrón  $X(n)$ , y 0 en otro caso, es decir:

$$M_{in} = \begin{cases} 1 & \text{si } \|X(n) - C_i\| < \|X(n) - C_s\| \forall s \neq i, s = 1, 2, \dots, K \\ 0 & \text{en otro caso} \end{cases} \quad (4.9)$$

Dado  $K$  el número de clases,  $\{X(n) = (x_1(n), x_2(n), \dots, x_p(n))\}_{n=1\dots N}$  el conjunto de patrones de entrada y  $\{C_i = (c_{i1}, c_{i2}, \dots, c_{ip})\}_{i=1, \dots, K}$  los centros de las clases, los pasos para la aplicación del algoritmo son los siguientes:

Paso 1 Se inicializan los centros de las  $K$  clases. Pueden inicializarse a  $K$  patrones aleatorios del conjunto de patrones disponibles, o bien puede realizarse aleatoriamente, en cuyo caso es conveniente que se inicien dentro del rango de valores de los patrones de entrada.

Paso 2 Se asignan  $N_i$  patrones de entrada a cada clase  $i$  del siguiente modo:

El patrón  $X(n)$  pertenece a la clase  $i$  si  $\|X(n) - C_i\| < \|X(n) - C_s\| \forall s \neq i$  con  $s = 1, 2, \dots, K$ .

Por tanto, cada clase tendrá asociado un determinado número de patrones de entrada, aquellos más cercanos al centro de la clase.

Paso 3 Se calcula la nueva posición de los centros de las clases como la media de todos los patrones que pertenecen a su clase, es decir:

$$c_{ij} = \frac{1}{N_i} \sum_{n=1}^N M_{in} x_j(n) \text{ para } j = 1, 2, \dots, p, i = 1, 2, \dots, K \quad (4.10)$$

Paso 4 Se repiten los pasos 2 y 3 hasta que las nuevas posiciones de los centros no se modifiquen respecto a su posición anterior, es decir, hasta que:

$$\|C_i^{nuevo} - C_i^{anterior}\| < \varepsilon \quad \forall i = 1, 2, \dots, K \quad (4.11)$$

siendo  $\varepsilon$  un número real positivo próximo a cero que marca la finalización del algoritmo.

El algoritmo de K-medias es un método fácil de implementar y usar; suele ser un algoritmo bastante eficiente en problemas de clasificación, pues converge en pocas iteraciones hacia un mínimo de la función  $J$  dada por la Ecuación (4.8), aunque podría tratarse de un mínimo local.

Uno de los inconvenientes o desventajas que se le puede atribuir al algoritmo de K-medias es su dependencia de los valores iniciales asignados a cada centro, lo cual hace que en muchas ocasiones, y siempre dependiendo del problema, se obtengan soluciones locales. Esta dependencia de las inicializaciones de los centros se ilustra en la Figura 4.5. La figura muestra los centros obtenidos después de aplicar el algoritmo de K-medias (con  $K=10$ ) a una distribución aleatoria de puntos en el intervalo  $[0, 1]$  para dos inicializaciones diferentes. Se observa que los centros -etiquetados con cruces- convergen a puntos diferentes del plano. Para la primera inicialización, la función  $J$  dada por la Ecuación (4.8) toma el valor  $J = 1,5532$ , y partiendo de la segunda inicialización, el error del algoritmo toma el valor  $J = 1,6705$ . Algunos autores han propuesto mejoras del algoritmo de K-medias, siempre con el objetivo de obtener soluciones óptimas o globales que no dependan de la inicialización de los centros ([Chen, 1995], [Chinrungruang and Sequin, 1995]).

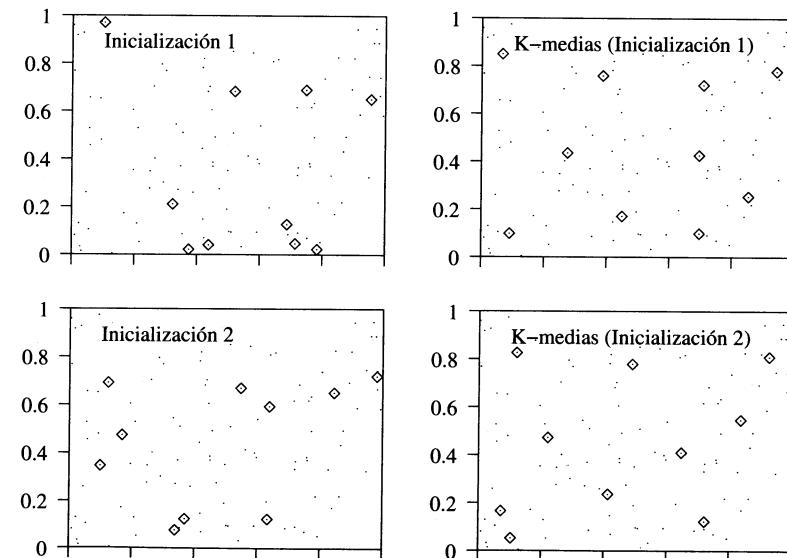


Figura 4.5: Aplicación del algoritmo K-medias a dos inicializaciones diferentes

#### ■ Determinación de las amplitudes

Una vez determinados los centros de las funciones de base radial, las amplitudes o ~~desviaciones~~ de dichas funciones deben calcularse de manera

que cada neurona oculta se active en una región del espacio de entrada y de manera que el solapamiento de las zonas de activación de una neurona a otra sea lo más ligero posible, para suavizar así la interpolación.

Las amplitudes de cada función de base radial se pueden determinar usando heurísticas basadas en los /itp-vecinos más cercanos, tal y como propuso Moody [Moody and Darken, 1989], las cuales permiten que el solapamiento entre las neuronas ocultas sea lo más suave posible. Se pueden utilizar diferentes heurísticas, como, por ejemplo:

- Media uniforme de las distancias euclídeas del centro  $C_i$  a los  $p$  centros más cercanos:

$$d_i = \frac{1}{p} \sum_p \|C_i - C_p\| \quad (4.12)$$

- Otra opción bastante efectiva es determinar la amplitud de la función de base radial como la media geométrica de la distancia del centro a sus dos vecinos más cercanos:

$$d_i = \sqrt{\|C_i - C_t\| \|C_i - C_s\|} \quad (4.13)$$

siendo  $C_t$  y  $C_s$  los dos centros más cercanos al centro  $C_i$ .

### Fase supervisada

En esta fase se calculan los pesos y umbrales de las neuronas de salida de la red. En este caso, el objetivo es minimizar las diferencias entre las salidas de la red y las salidas deseadas. Por tanto, el proceso de aprendizaje está guiado por la minimización de una función error computada en la salida de la red:

$$E = \frac{1}{N} \sum_{n=1}^N e(n) \quad (4.14)$$

donde  $N$  es el número de patrones o muestras y  $e(n)$  es el error cometido por la red para el patrón  $X(n)$ , que viene dado generalmente por:

$$e(n) = \frac{1}{2} \sum_{k=1}^r (s_k(n) - y_k(n))^2 \quad (4.15)$$

siendo  $Y(n) = (y_1(n), \dots, y_r(n))$  y  $S(n) = (s_1(n), \dots, s_r(n))$  los vectores de salida de la red y salida deseada para el patrón de entrada  $X(n)$ , respectivamente.

#### Mínimos cuadrados

Para resolver este problema de optimización se suele utilizar una técnica basada en la corrección del error. En la Ecuación (4.1) se observa que las salidas de la red de base radial dependen linealmente de los pesos y umbrales, por lo que un método bastante simple y eficiente es el algoritmo de los mínimos cuadrados. De este modo, los pesos y umbrales de la red se determinan mediante un proceso iterativo gobernado por la siguiente ley:

$$w_{ik}(n) = w_{ik}(n-1) - \alpha_1 \frac{\partial e(n)}{\partial w_{ik}} \quad (4.16)$$

$$u_k(n) = u_k(n-1) - \alpha_1 \frac{\partial e(n)}{\partial u_k} \quad (4.17)$$

para  $k = 1, 2, \dots, r$  y para  $i = 1, \dots, m$

donde  $e(n)$  es el error dado por la Ecuación (4.15) y  $\alpha_1$  es la razón o tasa de aprendizaje.

Teniendo en cuenta la expresión del error (Ecuación 4.15) y que el peso  $w_{ik}$  y el umbral  $u_k$  únicamente afectan a la neurona de salida  $k$ , se obtiene que:

$$\frac{\partial e(n)}{\partial w_{ik}} = -(s_k(n) - y_k(n)) \frac{\partial y_k(n)}{\partial w_{ik}} \quad (4.18)$$

$$\frac{\partial e(n)}{\partial u_k} = -(s_k(n) - y_k(n)) \frac{\partial y_k(n)}{\partial u_k} \quad (4.19)$$

Derivando la salida  $y_k(n)$  de la red de base radial dada en la Ecuación (4.1) respecto a los pesos y umbrales, se obtiene que:

$$\frac{\partial y_k(n)}{\partial w_{ik}} = \phi_i(n) \quad (4.20)$$

donde  $\phi_i(n)$  es la activación de la neurona oculta  $i$  para el patrón de entrada  $X(n)$ , y

$$\frac{\partial y_k(n)}{\partial u_k} = 1 \quad (4.21)$$

Por tanto, las leyes dadas por las ecuaciones (4.16) y (4.17) para adaptar los pesos y umbrales de la capa de salida de la red de base radial se pueden escribir de la siguiente forma:

$$w_{ik}(n) = w_{ik}(n-1) + \alpha_1 (s_k(n) - y_k(n)) \phi_i(n) \quad (4.22)$$

$$u_k(n) = u_k(n-1) + \alpha_1 (s_k(n) - y_k(n)) \quad (4.23)$$

para  $k = 1, 2, \dots, r$  y para  $i = 1, \dots, m$

Cuando se calculan los pesos mediante la ley de aprendizaje dada por las ecuaciones (4.22) y (4.23), la convergencia es bastante rápida, consiguiendo una solución en un número pequeño de iteraciones o ciclos de aprendizaje.

### ■ Matriz seudoinversa

Debido a que la salida de la red (Ecuación 4.1) depende linealmente de los pesos y umbrales, otro método para el cálculo de dichos parámetros es el llamado *método de la seudoinversa* [Broomhead and Lowe, 1988]. Se trata de un método que proporciona una solución directa al problema de optimización. Dicha solución viene dada por la siguiente expresión matricial:

$$W = G^+ \cdot S = (G^t \cdot G)^{-1} \cdot G^t \cdot S \quad (4.24)$$

donde  $W$  es la matriz de pesos y umbrales de la red de base radial, de orden  $(m + 1) \times r$ :

$$W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1r} \\ w_{21} & w_{22} & \dots & w_{2r} \\ \dots & \dots & \dots & \dots \\ w_{m1} & w_{m2} & \dots & w_{mr} \\ u_1 & u_2 & \dots & u_r \end{pmatrix}$$

$G^+ = (G^t \cdot G)^{-1} \cdot G^t$  es la matriz seudoinversa de  $G$ , siendo  $G^t$  la matriz traspuesta de  $G$ .  $G$  es una matriz de orden  $N \times (m + 1)$  que contiene las activaciones de las neuronas ocultas de la red para los patrones de entrada:

$$G = \begin{pmatrix} \phi_1(1) & \phi_2(1) & \dots & \phi_m(1) & 1 \\ \phi_1(2) & \phi_2(2) & \dots & \phi_m(2) & 1 \\ \dots & \dots & \dots & \dots & \dots \\ \phi_1(N) & \phi_2(N) & \dots & \phi_m(N) & 1 \end{pmatrix}$$

donde  $\phi_i(n)$  es la activación de la neurona oculta  $i$  para el patrón de entrada  $X(n)$ .

$Y$   $S$  es la matriz de salidas deseadas para la red, de orden  $N \times r$ :

$$S = \begin{pmatrix} s_1(1) & s_2(1) & \dots & s_r(1) \\ s_1(2) & s_2(2) & \dots & s_r(2) \\ \dots & \dots & \dots & \dots \\ s_1(N) & s_2(N) & \dots & s_r(N) \end{pmatrix}$$

donde  $s_k(n)$  es la coordenada  $k$  de la salida deseada para el patrón  $X(n)$ .

De este modo, y como se observa en la Ecuación (4.24), los pesos y umbrales de la red se pueden obtener de manera directa; basta calcular la seudoinversa de la matriz  $G$ , con lo que se obtiene así una solución óptima al problema de minimización. Sin embargo, es necesario señalar que, aunque dicho método proporciona una solución directa al problema y no iterativo,

desde un punto de vista práctico no es precisamente el método más eficiente, pues el cálculo de la matriz seudoinversa debe realizarse mediante métodos numéricos ([Golub and Loan, 1989], [Haykin, 1991]), los cuales podrían requerir un alto coste computacional y ocasionar errores debido a problemas de precisión. Por tanto, en el contexto de redes de neuronas de base radial, el método más utilizado para la determinación de pesos y umbrales es el algoritmo de los mínimos cuadrados.

### Resumen del aprendizaje híbrido

Dado  $\{(X(n), S(n))\}_{n=1,\dots,N}$  el conjunto de patrones de entrada y sus salidas deseadas, el método de aprendizaje híbrido para las redes de neuronas de base radial se puede resumir en los siguientes pasos:

- Se aplica el algoritmo de K-medias sobre el conjunto de patrones de entrada  $\{(X(n))\}_{n=1,\dots,N}$  para el cálculo de los centros de las funciones de base radial, siendo  $K$  el número de neuronas ocultas de la red.
- Se calculan las amplitudes o desviaciones de las funciones de base radial utilizando alguna de las expresiones dadas por las ecuaciones (4.12) o (4.13).
- Se determinan los pesos y umbrales de la capa de salida siguiendo el siguiente proceso iterativo:
  1. Se inicializan aleatoriamente los pesos y umbrales de la capa de salida.
  2. Se toma un patrón del conjunto de patrones disponibles  $(X(n), S(n))$  y se calcula la salida de la red,  $Y(n)$ , para el patrón de entrada  $X(n)$ .
  3. Se evalúa el error  $e(n)$  cometido por la red para dicho patrón (Ecuación 4.15).
  4. Se modifican los parámetros de la red utilizando las leyes de aprendizaje dadas por las ecuaciones (4.22) y (4.23).
  5. Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento.
  6. Se repiten los pasos 2, 3, 4 y 5 hasta conseguir la convergencia, es decir, hasta que la suma de los errores para todos los patrones (Ecuación 4.14) se estabilice.

#### 4.3.2. Método de aprendizaje totalmente supervisado

A diferencia del método de aprendizaje híbrido descrito anteriormente, el método de aprendizaje totalmente supervisado no conserva, en principio, las propiedades o características locales de las redes de base radial. En este caso, todos los parámetros de la red de base radial -centros, amplitudes, pesos y umbrales- se determinan de manera completamente supervisada y con el objetivo de minimizar el error cuadrático medio, es decir, las diferencias entre las salidas de la red y las salidas esperadas (ecuaciones 4.14 y 4.15).

Al utilizar este método, en ningún momento el proceso de aprendizaje se guía para que las amplitudes alcancen valores tales que el solapamiento de las activations de las neuronas ocultas sea lo más suave posible, sino que se determinan para minimizar el error cometido por la red en la capa de salida. Por tanto, no es posible esperar que la red siga conservando sus características locales.

En las redes de neuronas de base radial, como se comentó anteriormente, las salidas de la red dependen linealmente respecto de los pesos y umbrales. Sin embargo, la dependencia pasa a ser no lineal en lo que respecta a los parámetros de las neuronas ocultas -centros y desviaciones- debido al uso de las funciones de base radial, las cuales son funciones no lineales. Por tanto, al utilizar el método de aprendizaje totalmente supervisado, los parámetros de la red se determinan empleando una técnica de optimización no lineal, más concretamente el método de descenso del gradiente. Mediante un proceso iterativo y siguiendo la dirección negativa del gradiente del error, el método proporciona un mínimo, que pudiera ser local, de la función error  $E$  dada por la Ecuación (4.14). De este modo, los centros, amplitudes, pesos y umbrales de la red se modifican para cada patrón  $X(n)$  de acuerdo con las siguientes leyes de aprendizaje:

$$w_{ik}(n) = w_{ik}(n-1) - \alpha_1 \frac{\partial e(n)}{\partial w_{ik}} \quad (4.25)$$

$$u_k(n) = u_k(n-1) - \alpha_1 \frac{\partial e(n)}{\partial u_k} \quad (4.26)$$

$$c_{ij}(n) = c_{ij}(n-1) - \alpha_2 \frac{\partial e(n)}{\partial c_{ij}} \quad (4.27)$$

$$d_i(n) = d_i(n-1) - \alpha_3 \frac{\partial e(n)}{\partial d_i} \quad (4.28)$$

para  $j = 1, 2, \dots, p$ , para  $i = 1, \dots, m$  y para  $k = 1, 2, \dots, r$

donde  $\alpha_1, \alpha_2$  y  $\alpha_3$  son las razones o tasas de aprendizaje para los pesos, centros y amplitudes, respectivamente. Dichas tasas no tienen por qué tomar los mismos valores.

En el caso de las redes de neuronas de base radial, la utilización del método de descenso del gradiente no implica una retropropagación del error, como ocurría en la regla delta generalizada para el aprendizaje del PERCEPTRON multicapa (Sección 3.3). En el contexto de las redes de base radial, la aplicación del método de descenso del gradiente implica el cálculo de la derivada del error  $e(n)$  respecto a cada uno de los parámetros -centros, amplitudes o desviaciones, pesos y umbrales-. Dichas derivadas poseen expresiones diferentes, ya que cada uno de estos parámetros intervienen de manera distinta en la salida de la red. A continuación, se desarrollan las leyes de aprendizaje para los parámetros de las redes de base radial.

#### ■ Pesos y umbrales

Las derivadas del error  $e(n)$  (Ecuación 4.15) respecto a los pesos y umbrales de la red han sido obtenidas en el apartado anterior (**fase supervisada**)

del método híbrido), obteniendo las leyes de aprendizaje dadas por las ecuaciones (4.22) y (4.23):

$$w_{ik}(n) = w_{ik}(n-1) + \alpha_1(s_k(n) - y_k(n))\phi_i(n)$$

$$u_k(n) = u_k(n-1) + \alpha_1(s_k(n) - y_k(n))$$

para  $k = 1, 2, \dots, r$  y para  $i = 1, \dots, m$

#### ■ Centros

Teniendo en cuenta la expresión del error  $e(n)$  (Ecuación 4.15) y aplicando la regla de la cadena para derivar, la derivada de dicho error respecto al parámetro  $c_{ij}$  viene dada por:

$$\frac{\partial e(n)}{\partial c_{ij}} = - \sum_{k=1}^r (s_k(n) - y_k(n)) \frac{\partial y_k(n)}{\partial c_{ij}} \quad (4.29)$$

El parámetro  $c_{ij}$  -coordenada  $j$  del centro  $i$ - sólo interviene en la activación de la neurona oculta  $i$ , por lo que al derivar la salida  $k$  de la red (Ecuación 4.1) sólo es necesario derivar el término  $i$  del sumatorio, resultando entonces:

$$\frac{\partial e(n)}{\partial c_{ij}} = - \sum_{k=1}^r (s_k(n) - y_k(n)) w_{ik} \frac{\partial \phi_i(n)}{\partial c_{ij}} \quad (4.30)$$

Aplicando de nuevo la regla de la cadena para derivar la función  $\phi_i$  dada por la Ecuación (4.7) respecto a  $c_{ij}$  se obtiene que:

$$\frac{\partial \phi_i(n)}{\partial c_{ij}} = \phi_i(n) \frac{(x_j - c_{ij})}{d_i^2} \quad (4.31)$$

Sustituyendo la Ecuación (4.31) en (4.30), la ley para modificar los centros de las funciones de base radial dada por la Ecuación (4.27) adopta la siguiente expresión:

$$c_{ij}(n) = c_{ij}(n-1) + \alpha_2 \left( \sum_{k=1}^r (s_k(n) - y_k(n)) w_{ik} \right) \phi_i(n) \frac{(x_j - c_{ij})}{d_i^2} \quad (4.32)$$

para  $j = 1, 2, \dots, p$  y para  $i = 1, \dots, m$

#### ■ Amplitudes

Al igual que en el caso anterior, para obtener la derivada del error  $e(n)$  respecto al parámetro  $d_i$  es necesario derivar las salidas de la red respecto a dicho parámetro:

$$\frac{\partial e(n)}{\partial d_i} = - \sum_{k=1}^r (s_k(n) - y_k(n)) w_{ik} \frac{\partial \phi_i(n)}{\partial d_i} \quad (4.33)$$

La derivada de la función  $\phi_i$  (Ecuación 4.7) respecto al parámetro  $d_i$  es:

$$\frac{\partial \phi_i(n)}{\partial d_i} = \phi_i(n) \frac{\|X(n) - C_i\|^2}{d_i^3} \quad (4.34)$$

Por tanto, la ley para modificar las amplitudes de las funciones de base radial (Ecuación 4.28) es:

$$d_i(n) = d_i(n-1) + \alpha_3 \left( \sum_{k=1}^r (s_k(n) - y_k(n)) w_{ik} \right) \phi_i(n) \frac{\|X(n) - C_i\|^2}{d_i^3} \quad (4.35)$$

para  $i = 1, \dots, m$

Al tratarse de un proceso iterativo, es necesario inicializar todos los parámetros de la red. Generalmente, en el contexto de redes de neuronas, dicha inicialización suele hacerse de manera aleatoria y con valores cercanos a cero. Sin embargo, en el caso de las redes de base radial, ésta no tiene que ser la manera más adecuada de inicializar, fundamentalmente, los centros de las funciones de base radial. Sería aconsejable inicializar los centros de manera que representen zonas del espacio de patrones de entrada, limitando así la búsqueda del método a ciertas regiones del espacio. Por ejemplo, los centros podrían inicializarse a patrones de entrada seleccionados aleatoriamente o incluso a valores de los centros alcanzados después de aplicar un algoritmo de clasificación no supervisada a los patrones de entrada, lo cual facilitará la labor de optimización al método de descenso del gradiente. Esta manera de inicializar los centros conduce a la aplicación de la combinación de ambos métodos, híbrido y totalmente supervisado, para realizar el aprendizaje de redes de neuronas de base radial, como se verá más adelante.

#### Obtención de las leyes de aprendizaje del método totalmente supervisado para una red de base radial con dos neuronas de entrada, dos ocultas y una salida

En este ejemplo se va a ilustrar la obtención de las leyes de aprendizaje para los pesos y el umbral  $-w_1$ ,  $w_2$  y  $u$ , los centros  $-C_1 = (c_{11}, c_{12})$  y  $C_2 = (c_{21}, c_{22})$  y las desviaciones  $-d_1$  y  $d_2$  de la red que se muestra en la Figura 4.3 cuando se utiliza el método totalmente supervisado.

Puesto que la red tiene una neurona de salida, el error cometido para el patrón  $n$  viene dado por:

$$e(n) = \frac{1}{2} (s(n) - y(n))^2$$

- **Pesos y umbral.** Teniendo en cuenta que  $y(n)$  viene dado por:

$$y(n) = w_1 \phi_1(n) + w_2 \phi_2(n) + u$$

el cálculo de la derivada del error respecto de  $w_1$ ,  $w_2$  y  $u$  es inmediato, pues basta derivar  $y(n)$  respecto de dichos parámetros. Se obtienen, entonces, las siguientes leyes de aprendizaje:

$$w_i(n) = w_i(n-1) + \alpha_1 (s(n) - y(n)) \phi_i(n) \text{ para } i = 1, 2$$

$$u(n) = u(n-1) + \alpha_1 (s(n) - y(n))$$

- **Centros.** Sea  $c_{12}$  la coordenada 2 del centro de la neurona 1,  $C_1$ . La derivada del error  $e(n)$  respecto a dicho parámetro es:

$$\frac{\partial e(n)}{\partial c_{12}} = -(s(n) - y(n)) \frac{\partial y(n)}{\partial c_{12}}$$

Debido a que  $y(n) = w_1 \phi_1(n) + w_2 \phi_2(n) + u$  y que  $c_{12}$  sólo interviene en la función de base radial  $\phi_1$ , se obtiene que:

$$\frac{\partial e(n)}{\partial c_{12}} = -(s(n) - y(n)) w_1 \frac{\partial \phi_1(n)}{\partial c_{12}}$$

$$\text{Derivando } \phi_1(n) = \exp^{-\frac{(x_1(n) - c_{11})^2 + (x_2(n) - c_{12})^2}{2d_1^2}},$$

$$\frac{\partial \phi_1(n)}{\partial c_{12}} = \phi_1(n) \frac{(x_2 - c_{12})}{d_1^2}$$

Por tanto:

$$c_{12}(n) = c_{12}(n-1) + \alpha_2 (s(n) - y(n)) w_1 \phi_1(n) \frac{(x_2 - c_{12})}{d_1^2}$$

Se observa en la expresión anterior que en la modificación del parámetro  $c_{12}$  interviene, por una parte, el peso  $w_1$  y la función de base radial  $\phi_1$ , y por otra parte, la coordenada  $x_2$  del patrón de entrada. Por tanto, dicha ley se puede generalizar fácilmente para el resto de las coordenadas de los centros de la red, obteniendo:

$$c_{ij}(n) = c_{ij}(n-1) + \alpha_2 (s(n) - y(n)) w_i \phi_i(n) \frac{(x_j - c_{ij})}{d_i^2} \text{ para } i, j = 1, 2$$

- Amplitudes.** Sea  $d_1$  la desviación de la neurona oculta 1. Siguiendo el procedimiento anterior, la derivada del error respecto de dicho parámetro viene dada por:

$$\frac{\partial e(n)}{\partial d_1} = -(s(n) - y(n))w_1 \frac{\partial \phi_1(n)}{\partial d_1}$$

La derivada de la función  $\phi_1(n) = \exp^{-\frac{(x_1(n)-c_{11})^2+(x_2(n)-c_{12})^2}{2d_1^2}}$  respecto al parámetro  $d_1$  es:

$$\frac{\partial \phi_1(n)}{\partial d_1} = \phi_1(n) \frac{\|X(n) - C_1\|^2}{d_1^3}$$

Por tanto, la ley para modificar el parámetro  $d_1$  es:

$$d_1(n) = d_1(n-1) + \alpha_3(s(n) - y(n))w_1 \phi_1(n) \frac{\|X(n) - C_1\|^2}{d_1^3}$$

Al igual que en el caso anterior, esta ley es fácilmente generalizable para el resto de las amplitudes de las funciones de base radial, obteniendo:

$$d_i(n) = d_i(n-1) + \alpha_3(s(n) - y(n))w_i \phi_i(n) \frac{\|X(n) - C_i\|^2}{d_i^3} \text{ para } i = 1, 2$$

### Resumen del aprendizaje totalmente supervisado

Dado  $\{(X(n), S(n))\}_{n=1,\dots,N}$  el conjunto de patrones de entrada y sus salidas deseadas, el método de aprendizaje totalmente supervisado para las redes de neuronas de base radial se resume en los siguientes pasos:

1. Se inicializan todos los parámetros de la red. En el caso de las amplitudes, pesos y umbrales, esta inicialización suele hacerse de manera aleatoria con valores cercanos a cero. Para los centros es, generalmente, preferible inicializarlos aleatoriamente a patrones de entrada o a la salida de un algoritmo de clasificación aplicado en el espacio de entrada, como se comentó anteriormente.
2. Se toma un patrón del conjunto de patrones disponibles  $(X(n), S(n))$  y se calcula la salida de la red,  $Y(n)$ , para el patrón de entrada  $X(n)$ .
3. Se evalúa el error  $e(n)$  cometido por la red para dicho patrón (Ecuación 4.15).
4. Se modifican los pesos, umbrales, centros y amplitudes de la red utilizando las ecuaciones (4.22), (4.23), (4.32) y (4.35), respectivamente.
5. Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento.
6. Se repiten los pasos 2, 3, 4 y 5 hasta conseguir la convergencia, es decir, hasta que la suma de los errores para todos los patrones (Ecuación 4.14) se estabilice, momento en el que se alcanza un mínimo de dicha función.

### 4.3.3. Combinando ambos métodos de aprendizaje: híbrido y totalmente supervisado

Desde el punto de vista de la arquitectura de la red de neuronas de base radial, parece más lógico utilizar el método de aprendizaje híbrido para determinar sus parámetros, pues mantiene el carácter local de estas redes y, por tanto, sus características diferenciadoras.

En principio, esto no ocurre cuando se utiliza el método de aprendizaje totalmente supervisado, pues los parámetros de la red se modifican para minimizar el error en la salida. Esto hace, por ejemplo, que las desviaciones no estén limitadas a permanecer con valores tales que el solapamiento en las activaciones de las neuronas ocultas de la red sea el menor posible.

Una cuestión obvia que surge ante esta situación general es qué ventajas puede ofrecer el método totalmente supervisado frente al método híbrido. Es decir, qué ventaja puede ofrecer el adaptar de manera supervisada los centros y las desviaciones de las funciones de base radial. La respuesta a esta pregunta va a depender del problema o aplicación. No obstante es posible afirmar, a partir de estudios y resultados que aparecen en la literatura ([Lowe, 1989], [Wettschereck and Dietterich, 1992]), que, generalmente, una adaptación supervisada de los centros y las desviaciones puede conducir a una mayor precisión en los resultados, aunque siempre dependiendo del problema a resolver.

Para realizar el aprendizaje de las redes de base radial cabe la posibilidad de combinar ambos métodos de aprendizaje, híbrido y totalmente supervisado, con el objetivo de mejorar la precisión de la red en la resolución de problemas. Esta combinación consiste, básicamente, en inicializar los centros y desviaciones de las funciones de base radial utilizando el método híbrido, para, posteriormente, adaptarlos de manera supervisada utilizando el método totalmente supervisado.

La combinación de los métodos de aprendizaje híbrido y totalmente supervisado se resume en los siguientes pasos:

1. Se calculan los centros de las funciones de base radial aplicando el algoritmo de K-medias sobre el conjunto de patrones de entrada  $\{(X(n))\}_{n=1,\dots,N}$ .
2. Se calculan las amplitudes o desviaciones de las funciones de base radial utilizando alguna de las expresiones dadas por las ecuaciones (4.12) o (4.13).
3. Se aplica el algoritmo de los mínimos cuadrados para el cálculo de los pesos y umbrales de la red (ecuaciones 4.22 y 4.23).
4. Se toma un patrón del conjunto de patrones disponibles  $(X(n), S(n))$  y se calcula la salida de la red,  $Y(n)$ , para el patrón de entrada  $X(n)$ .
5. Se evalúa el error  $e(n)$  cometido por la red para dicho patrón (Ecuación 4.15).

6. Se modifican los pesos, umbrales, centros y amplitudes de la red utilizando las ecuaciones (4.22), (4.23), (4.32) y (4.35), respectivamente.

En este punto es necesario prestar especial atención a las razones o tasas de aprendizaje de los centros y amplitudes,  $\alpha_2$  y  $\alpha_3$ . Al aplicar las leyes dadas por las ecuaciones (4.32) y (4.35), los centros y desviaciones no deben sufrir cambios bruscos respecto a los valores obtenidos en los pasos 1 y 2, pues, en ese caso, la información obtenida en la fase no supervisada se perdería.

7. Se repiten los pasos 4, 5 y 6 para todos los patrones de entrenamiento.
8. Se repiten los pasos 4, 5, 6 y 7 hasta conseguir la convergencia, es decir, hasta que la suma de los errores para todos los patrones (Ecuación 4.14) se estabilice, momento en el que se alcanza un mínimo de dicha función.

#### 4.4. Ejemplo de funcionamiento de las redes de base radial

En esta sección se presenta un caso práctico para mostrar el funcionamiento de las redes de neuronas de base radial, tanto en lo que concierne a su diseño, como en lo que respecta al comportamiento de los diferentes métodos de aprendizaje, híbrido y totalmente supervisado.

Como caso práctico se plantea un problema de aproximación que consiste en aproximar el polinomio de Hermite, el cual es ampliamente utilizado en la literatura para estudiar el comportamiento de las redes de neuronas de base radial. El polinomio de Hermite (Figura 4.6) viene dado por la siguiente expresión:

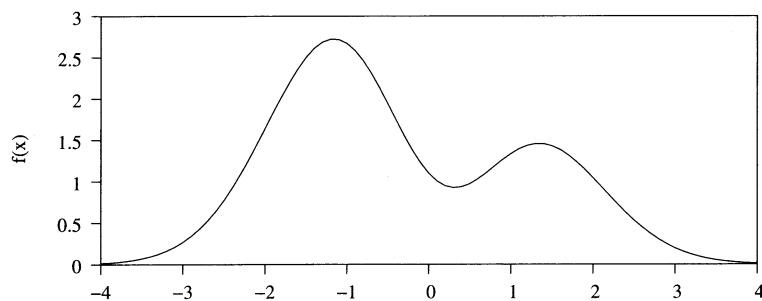


Figura 4.6: Polinomio de Hermite

$$f(x) = 1.1(1 - x + 2x^2)e^{\frac{1}{2}x^2} \quad (4.36)$$

A continuación se describen los pasos a seguir para la resolución de dicho problema utilizando las redes de neuronas de base radial. Algunos de estos pasos

son comunes a los realizados cuando se aborda el problema con el PERCEPTRON multicapa (ver Sección 3.5), por lo que serán tratados más brevemente.

##### 1. Conjunto de muestras o ejemplos sobre el problema

A partir de la expresión analítica del polinomio de Hermite se extrae un conjunto de 240 muestras, las cuales se generan siguiendo una distribución uniforme en el intervalo  $[-4, 4]$ . Dichos datos se normalizan o escalan en el intervalo  $[0, 1]$  realizando una transformación lineal, como se indicaba en la Sección 3.5.

##### 2. Extracción del conjunto de entrenamiento y validación

Del conjunto de muestras generadas, se utilizan 40 muestras extraídas aleatoriamente como patrones de entrenamiento y las 200 restantes muestras se utilizan como patrones de validación.

##### 3. Diseño de la arquitectura de la red de neuronas de base radial

En nuestro caso, la red de base radial tendrá una neurona de entrada que recibe el valor de la variable  $x$  y una neurona de salida. En principio se fijan 10 neuronas ocultas en la red. Posteriormente, se cambiará dicho parámetro para ver cómo influye en la resolución del problema.

##### 4. Aprendizaje de la red de base radial

La determinación de los parámetros de la red se llevará a cabo utilizando los diferentes métodos de aprendizaje, así como la combinación de ellos.

###### ■ Método de aprendizaje híbrido

Para la determinación de centros se va a utilizar el algoritmo de K-medias, con  $K = 10$ , pues la red tiene 10 neuronas ocultas y la inicialización de los centros se realiza de manera aleatoria. Téngase en cuenta que como dicha inicialización influye en el algoritmo de K-medias, influirá también en los resultados que se obtengan con la red. Para simplificar el ejemplo, sólo se realiza una inicialización de los centros, aunque para obtener conclusiones definitivas habría que realizar una batería de experimentos con diferentes inicializaciones y su posterior análisis de ciertas medidas estadísticas sobre los resultados (media, desviación típica, etc.).

La amplitud de cada neurona será calculada utilizando la media geométrica del centro de la neurona a sus dos vecinos más próximos (Ecuación 4.13). Y, finalmente, los pesos y umbrales de la red se determinan utilizando el método de los mínimos cuadrados (ecuaciones 4.22 y 4.23), con una razón de aprendizaje de  $\alpha_1 = 0.1$ .

En la Figura 4.7 se muestra la aproximación proporcionada por la red para los patrones de validación después de conseguir la convergencia de la red, para lo cual se han realizado 500 ciclos de aprendizaje o iteraciones.

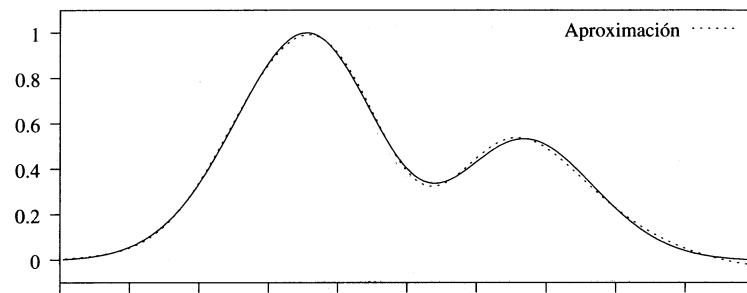


Figura 4.7: Aproximación del polinomio de Hermite con una red de base radial

Cuadro 4.2: Errores cuadráticos: método de aprendizaje híbrido

	Error de entrenamiento	Error de validación
5 neuronas ocultas	0,007031	0,006904
10 neuronas ocultas	0,000051	0,000053
15 neuronas ocultas	0,000008	0,000008

A continuación, se cambia el número de neuronas ocultas en la red para ver si es posible mejorar los resultados obtenidos anteriormente. Se entrena redes con 5 y 15 neuronas ocultas. En el cuadro 4.2 se muestran los errores cuadráticos cometidos por las diferentes arquitecturas sobre los conjuntos de entrenamiento y validación. Se observa, por un lado, que el mejor resultado se obtiene con 15 neuronas y, por otro lado, que el número de neuronas ocultas en la red podría ser un parámetro significativo en la resolución del problema utilizando redes de base radial, pues los errores obtenidos después de conseguir la convergencia son bastante diferentes. En la Figura 4.8 se muestra la evolución de los errores de validación durante 500 ciclos de aprendizaje para cada una de las arquitecturas.

#### ■ Método totalmente supervisado

Las redes de base radial con 5, 10 y 15 neuronas ocultas se entrena con el método totalmente supervisado, partiendo de valores iniciales aleatorios cercanos a cero para todos los parámetros de la red -centros, amplitudes, pesos y umbrales- y utilizando razones de aprendizaje  $\alpha_1 = \alpha_2 = \alpha_3 = 0,01$ . Nótese que si la razón de aprendizaje se elige más alta, los errores de entrenamiento y validación sufren oscilaciones bruscas a lo largo del aprendizaje.

En el cuadro 4.3 se muestran los errores cuadráticos obtenidos después de 500 ciclos de aprendizaje con las diferentes redes y en la Figura 4.9 se muestra la evolución de los errores de validación. Se observa que la red con 5 neuronas obtiene un mejor error que cuan-

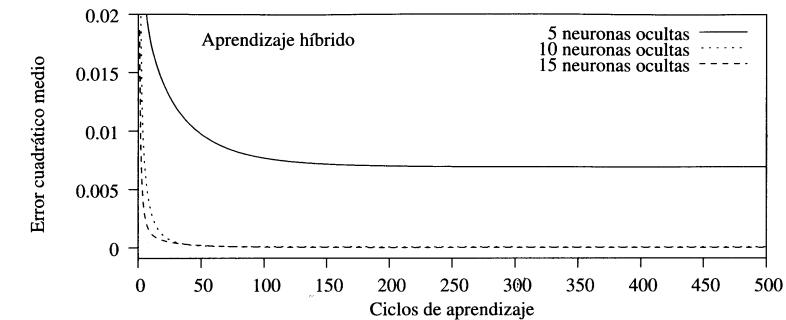


Figura 4.8: Evolución de los errores de validación variando el número de neuronas ocultas: método de aprendizaje híbrido

Cuadro 4.3: Errores cuadráticos: método de aprendizaje totalmente supervisado

	Error de entrenamiento	Error de validación
5 neuronas ocultas	0,0001530	0,0005734
10 neuronas ocultas	0,0000475	0,0000482
15 neuronas ocultas	0,0000166	0,0000449

do se utiliza el método de aprendizaje híbrido, aunque si se elige un número adecuado de neuronas (10 y 15 neuronas), dicho aprendizaje supera al método totalmente supervisado.

#### ■ Combinación de ambos métodos: híbrido y totalmente supervisado

En este caso, el método totalmente supervisado utilizado en el punto anterior se aplica a las redes con 5, 10 y 15 neuronas ocultas, pero en lugar de inicializar los parámetros -centros, amplitudes, pesos y umbrales- de manera aleatoria, se parte de los parámetros obtenidos después de aplicar el método de aprendizaje híbrido. Las razones de aprendizajes se fijan a  $\alpha_1 = \alpha_2 = \alpha_3 = 0,001$  para evitar oscilaciones en el error a lo largo de los ciclos de aprendizaje. Los errores obtenidos se muestran en el cuadro 4.4. Se observa que los errores cometidos por las redes al utilizar el método de aprendizaje híbrido (ver cuadro 4.2) han disminuido, con lo que se consigue una mejor precisión en los resultados.

## 4.5. Redes de base radial frente a perceptron multicapa

Las redes de base radial y el PERCEPTRON multicapa son dos tipos de redes de neuronas no lineales que poseen sus neuronas agrupadas en capas y las conexio-

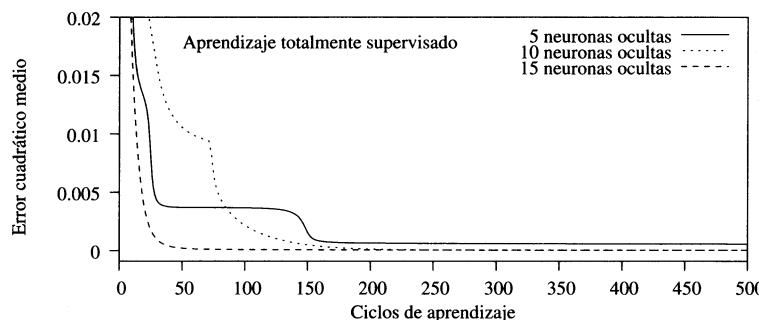


Figura 4.9: Evolución de los errores de validación variando el número de neuronas ocultas: método de aprendizaje totalmente supervisado

Cuadro 4.4: Errores cuadráticos: combinación de los métodos de aprendizaje híbrido y totalmente supervisado

	Error de entrenamiento	Error de validación
5 neuronas ocultas	0,0000074	0,0000077
10 neuronas ocultas	0,0000053	0,0000063
15 neuronas ocultas	0,0000010	0,0000010

nes están dirigidas hacia adelante -redes feedforward-. Ambas redes son además aproximadores universales, pues se ha demostrado que pueden aproximar cualquier función continua.

Estos dos tipos de redes presentan también algunas diferencias, como el número de capas ocultas -una única capa en el caso de las redes de base radial y tantas capas como se desee en el caso del PERCEPTRON multicapa-, la distribución de los pesos -en las redes de base radial las conexiones de la capa de entrada a la oculta no llevan pesos asociados- y, por ejemplo, también la linealidad en la capa de salida, la cual no es imprescindible para el caso del PERCEPTRON multicapa.

Sin embargo, la principal diferencia entre las redes de base radial y el PERCEPTRON multicapa radica en la función de activación de las neuronas ocultas de la red. Las primeras utilizan funciones de base radial, las cuales hacen que las neuronas ocultas de la red posean un carácter local, como se comentó en el apartado 4.2.2, activándose cada neurona en una determinada región del espacio de entrada. Por otro lado, el PERCEPTRON multicapa emplea funciones de activación sigmoidales, que permiten que las neuronas ocultas de la red posean activación global, es decir, que se activen en todo el espacio de entrada.

El uso de diferentes funciones de activación con las propiedades anteriormente mencionadas, hace que cada una de estas arquitecturas, PERCEPTRON multicapa y redes de base radial, tenga sus propias características, las cuales se analizan a continuación.

#### ■ El perceptron multicapa construye aproximaciones globales

Debido al uso de funciones de activación sigmoidal, el PERCEPTRON multicapa construye relaciones globales entre los datos de entrada y salida disponibles. Esto hace que el aprendizaje de la red sea lento, pues el cambio en un solo peso de la red provoca cambios en la salida para todos los patrones de entrada presentados anteriormente, reduciéndose así el efecto de previos ciclos de aprendizaje y retrasando la convergencia del algoritmo de aprendizaje.

#### ■ Las redes de base radial construyen aproximaciones locales

Cada neurona oculta de la red de base radial se especializa en una determinada región del espacio de entrada y construyen una aproximación local en dicha región. Por tanto, la relación que definen las redes de base radial entre los datos de entrada y salida es una suma de funciones no lineales y locales para diferentes regiones del espacio de entrada. A diferencia de cuando se construyen aproximaciones globales, la construcción de aproximaciones locales permite que el aprendizaje sea más rápido, ya que el cambio en un solo peso de red afecta únicamente a la neurona oculta asociada a dicho peso y, por tanto, a un determinado grupo de patrones de entrada, los pertenecientes a la clase que representa la neurona oculta en cuestión.

Debido al carácter local, el aprendizaje de estas redes es, generalmente, menos sensible al orden de presentación de los patrones que en el caso del PERCEPTRON multicapa.

En muchos casos, sin embargo, ocurre que para poder construir una aproximación mediante la suma de aproximaciones locales se requiere un alto número de neuronas ocultas, lo cual podría influir negativamente en la capacidad de generalización de las redes de base radial.

Finalmente, debe señalarse que el número de neuronas ocultas de la red puede aumentar exponencialmente con la dimensión del espacio de entrada. Por tanto, para aplicaciones que requieren un alto número de variables de entrada, las redes de base radial podrían no ser las más adecuadas.

# Capítulo 5

## REDES DE NEURONAS RECURRENTES

### 5.1. Introducción

En las redes de neuronas estudiadas en capítulos anteriores, como el PERCEPTRON multicapa, las redes de base radial o los mapas autoorganizados de Kohonen, las conexiones entre las neuronas suelen estar sometidas a una fuerte restricción, que consiste en no permitir conexiones entre neuronas creando ciclos o bucles. En este capítulo se van a estudiar un grupo de redes de neuronas que no están sujetas a dicha restricción en la conectividad, y que se engloban bajo el nombre de redes de neuronas recurrentes. Si bien, en este capítulo, no se pretende hacer un estudio detallado y riguroso de todas las arquitecturas de redes recurrentes que aparecen en la literatura y de sus algoritmos de aprendizaje, sí se pretende proporcionar una visión global de este grupo de redes, presentando las arquitecturas más conocidas y utilizadas.

Las redes de neuronas recurrentes se caracterizan porque se crean bucles en las neuronas de la red mediante el uso de las llamadas conexiones recurrentes, pudiendo aparecer en la red conexiones de una neurona a ella misma, conexiones entre neuronas de una misma capa o conexiones de las neuronas de una capa a la capa anterior, como se muestra en la Figura 5.1.

La consideración de conexiones recurrentes en una red de neuronas implica, generalmente, un aumento del número de pesos o parámetros ajustables en la red, lo cual permite que aumente la capacidad de representación, pues en las redes de neuronas artificiales la información se representa de manera distribuida en los pesos de las conexiones y no en las propias neuronas. Sin embargo, el aumento de parámetros ajustables y, sobre todo, la inclusión de éstos de manera recurrente complica, generalmente, el aprendizaje de las redes recurrentes.

Al introducir conexiones recurrentes creando bucles, la activación de una neurona con conexiones recurrentes (ver Figura 5.1) ya no depende sólo de las

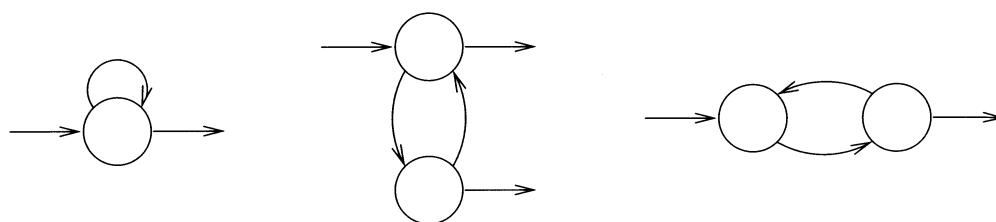


Figura 5.1: Ejemplos de neuronas con conexiones recurrentes

activaciones de las neuronas en la capa anterior, sino que depende también del estado o activación de cualquier otra neurona de la red conectada a ella, o incluso de su propia activación. Por tanto, en el contexto de redes de neuronas recurrentes es necesario incluir la variable tiempo en la activación o estado de una neurona, la cual viene dada por:

$$a_i(t+1) = f_i \left( \sum_j w_{ji} a_j(t) \right) \quad (5.1)$$

donde el índice  $j$  varía en el conjunto de todas las neuronas conectadas a la neurona  $i$ .

La presencia de la variable tiempo en las activaciones de las neuronas recurrentes, hace que estas redes posean un comportamiento dinámico o temporal. Dicho comportamiento temporal puede entenderse de dos formas diferentes, lo cual implica dos maneras distintas de entender el modo de actuación y aprendizaje dentro del grupo de redes recurrentes. Éstos son:

- Evolución de las activaciones de la red hasta alcanzar un punto estable.

Para estas redes el modo de actuación consiste en evolucionar la red, es decir las activaciones de sus neuronas, desde un estado inicial hasta conseguir que las activaciones de todas las neuronas de la red no se modifiquen, momento en el que se considera que la red ha alcanzado un punto estable.

Generalmente, el estado inicial viene dado por el patrón de entrada y el estado estable representa el patrón de salida de la red.

Una de las redes recurrentes más conocidas dentro de este modo de actuación es la red de Hopfield, la cual será estudiada en este capítulo.

- Evolución de las activaciones de la red en modo continuo.

En este caso, para cada instante de tiempo se dispone de la salida de la red, la cual depende de la entrada en el instante inmediatamente anterior. En términos generales, el aprendizaje de este tipo de redes se puede llevar a cabo de dos modos diferentes:

- Aprendizaje por épocas: dado un intervalo de tiempo o época, la red se evoluciona durante dicho intervalo, y cuando alcanza el instante

final se adaptan o modifican los pesos de la red. Una vez transcurrida la época, la red se reinicializa y se entra en una nueva época.

- Aprendizaje en tiempo real o continuo: la ley de aprendizaje para modificar los pesos de la red se aplica en cada instante de tiempo, siempre y cuando exista la salida deseada para la red en dicho instante.

Dentro del grupo de redes recurrentes que poseen este modo de actuación se pueden englobar las redes conocidas como redes parcialmente recurrentes y redes totalmente recurrentes. Las primeras se caracterizan porque sólo unas pocas conexiones recurrentes son introducidas en la red, mientras que las segundas no tienen restricciones en la consideración de conexiones recurrentes. Ambos tipos de redes utilizan algoritmos de aprendizaje supervisados para la modificación de sus parámetros, los cuales serán estudiados en este capítulo.

El comportamiento dinámico de las redes recurrentes debido a la inclusión de conexiones recurrentes facilita el tratamiento de información temporal o patrones dinámicos, es decir, patrones que dependen del tiempo en el sentido de que el valor del patrón en un determinado instante depende de sus valores en instantes anteriores de tiempo. En principio, las redes recurrentes son las más indicadas para abordar este tipo de problemas. Sin embargo, éstas no son las únicas redes de neuronas que pueden utilizarse para tratar información temporal. De hecho, también pueden emplearse estructuras estáticas de redes de neuronas, como el PERCEPTRON multicapa o las redes de base radial; basta considerar como entrada a la red una secuencia finita temporal del patrón en el pasado.

Aunque la aplicación fundamental de las redes recurrentes es el procesamiento de patrones dinámicos, se pueden aplicar también a patrones estáticos, es decir, patrones en los que no interviene la variable tiempo y cuyo procesamiento no depende del orden de presentación a la red. Por ejemplo, el modelo recurrente de red de Hopfield se aplica fundamentalmente a patrones estáticos.

A pesar de la complejidad que poseen las redes de neuronas recurrentes y la dificultad, en ocasiones, de realizar el aprendizaje, éstas han sido utilizadas para abordar diferentes tipos de problemas. Por ejemplo, problemas de modelización neurobiológica [Anastasio, 1991], tareas lingüísticas [Giles et al., 1991], reconocimiento de palabras y fonemas [Robinson and Fallside, 1991], control de procesos dinámicos ([Parlos et al., 1994], [Puskoris and Feldkamp, 1994]), entre otros.

## 5.2. Red de Hopfield

Una de las mayores contribuciones en el área de las redes de neuronas fue realizada por John Hopfield a principios de la década de los 80, proponiendo un modelo de red de neuronas no lineal, conocida como red de Hopfield [Hopfield, 1982].

Generalmente, la red de Hopfield es presentada como un modelo de memoria asociativa de patrones o muestras, en el sentido de que es capaz de recuperar patrones almacenados a partir de información incompleta sobre los patrones o incluso a partir de patrones con ruido.

Debido a la arquitectura y al funcionamiento, la red de Hopfield se puede incluir dentro de las redes de neuronas recurrentes, pues todas las neuronas están conectadas con todas las demás, además de existir un procesamiento temporal de los patrones. Sin embargo, lo que la diferencia del resto de las redes de neuronas recurrentes es que actúa como memoria asociativa, procesando patrones generalmente estáticos, es decir, patrones en las que no interviene la variable tiempo.

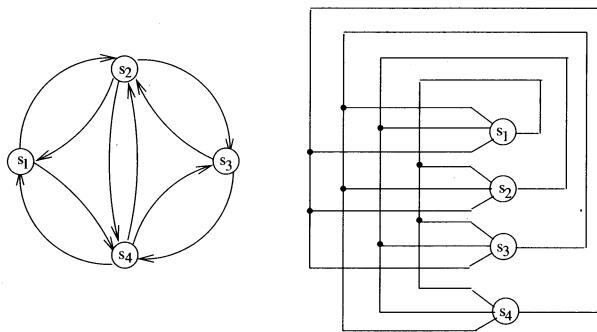


Figura 5.2: Dos esquemas diferentes para representar la red de Hopfield para  $n=4$

La red de Hopfield está formada por  $n$  neuronas, cada una conectada a todas las demás salvo a ella misma, como se muestra en la Figura 5.2. La matriz de conexiones de la red de Hopfield es una matriz  $W = (w_{ij})$  de orden  $n \times n$ , donde  $w_{ij}$  representa el peso de la conexión de la neurona  $i$  a la neurona  $j$ . Dicha matriz posee las siguientes particularidades:

- Es una matriz simétrica, es decir,  $w_{ij} = w_{ji} \forall i, j = 1, \dots, n$ . Esto implica que el peso de la conexión de la neurona  $i$  a la neurona  $j$  es igual al peso de la conexión de la neurona  $j$  a la neurona  $i$ .
- Los elementos de la diagonal de la matriz son iguales a cero, es decir,  $w_{ii} = 0 \forall i = 1, \dots, n$ , debido a que en la red de Hopfield no existen conexiones de una neurona a ella misma.

Inspirado en el concepto formal de neurona de McCulloch-Pitts, las neuronas de la red de Hopfield poseen dos estados, generalmente  $-1$  y  $1$ , que vienen determinados por el nivel o potencial de activación que recibe la neurona. De este modo, el estado de la neurona  $i$  en un instante de tiempo  $t + 1$ , denotado como  $s_i(t + 1)$ , viene dado por:

$$s_i(t + 1) = \text{sgn}(v_i(t + 1)) \text{ para } i = 1, 2, \dots, n \quad (5.2)$$

donde  $\text{sgn}$  es la función signo dada por:

$$\text{sgn}(v_i(t + 1)) = \begin{cases} +1 & \text{si } v_i(t + 1) > 0 \\ -1 & \text{si } v_i(t + 1) < 0 \end{cases} \quad (5.3)$$

y  $v_i(t + 1)$  es el nivel de activación que actúa sobre la neurona  $i$ , calculado como:

$$v_i(t + 1) = \sum_{j=1}^n w_{ji} s_j(t) - u_i \text{ para } i = 1, 2, \dots, n \quad (5.4)$$

donde  $s_j(t)$  es el estado de la neurona  $j$  en el instante anterior  $t$  y  $u_i$  es un umbral fijo aplicado a la neurona  $i$ .

En el caso de que el nivel de activación que recibe la neurona,  $v_i(t + 1)$ , sea igual a cero, se considera que el estado de la neurona no cambia con respecto al instante de tiempo anterior, es decir que  $s_i(t + 1) = s_i(t)$ .

De las definiciones anteriores, se observa que para la red de Hopfield no tiene sentido hablar de neuronas de entrada o salida de la red, sino del estado de la red en cada instante de tiempo. Para una red de Hopfield con  $n$  neuronas, el estado viene dado por:

$$s(t + 1) = [s_1(t + 1), s_2(t + 1), \dots, s_n(t + 1)]^t \quad (5.5)$$

donde el símbolo  $t$  denota la matriz traspuesta. Dicho estado  $s$  representa una palabra binaria de  $n$  bits de información.

### 5.2.1. Aprendizaje y mecanismo de actuación de la red de Hopfield

En la red de Hopfield se distinguen dos fases de operación, llamadas fase de almacenamiento y fase de recuperación. Durante la fase de almacenamiento se van a determinar los valores que deben tomar los pesos de la red para almacenar un conjunto de patrones, y la fase de recuperación describe el mecanismo para recuperar la información almacenada a partir de información incompleta.

#### Fase de almacenamiento

Sea  $\{x(k) = (x_1(k), x_2(k), \dots, x_n(k))\}_{k=1, \dots, p}$  el conjunto de patrones que se desea almacenar, donde cada patrón  $x(k)$  es un vector  $n$ -dimensional cuyas componentes toman valores binarios, es decir, valores  $-1$  o  $1$ . De acuerdo con la regla Hebb [Hebb, 1949] para almacenar patrones, el peso de la conexión de la neurona  $j$  a la neurona  $i$  en la red de Hopfield viene dado por:

$$w_{ji} = \sum_{k=1}^p x_j(k)x_i(k) \forall i \neq j \quad (5.6)$$

En la ecuación anterior se observa que si  $x_j(k)$  y  $x_i(k)$  son iguales [ $x_j(k) = x_i(k) = 1$  o  $-1$ ], el valor del peso  $w_{ji}$  se incrementa en una unidad, y en cualquier otro caso, el valor del peso se decrementa en una unidad.

### Fase de recuperación

Sea  $x = (x_1, x_2, \dots, x_n)$  un patrón de prueba, diferente a los patrones almacenados en la fase anterior. Dicho patrón representa, generalmente, una versión de algún patrón almacenado  $x(k)$  con información incompleta o ruido. Mediante esta fase, la red de Hopfield va a recuperar el patrón almacenado más parecido al patrón de prueba  $x$ . Para ello, se sigue el siguiente procedimiento:

1. Se inicializan los estados de las  $n$  neuronas de la red utilizando dicho patrón  $x$ , es decir:

$$s_i(0) = x_i \text{ para } i = 1, 2, \dots, n \quad (5.7)$$

2. Se calculan los estados de la red en los siguientes instantes de tiempo utilizando las ecuaciones (5.2), (5.3) y (5.4) hasta conseguir un punto estable o punto fijo de la red, entendiendo como punto estable aquel en el que los estados de todas las neuronas de la red permanecen invariantes con el tiempo, es decir:

$$s_i(t+1) = s_i(t) \forall i = 1, 2, \dots, n \quad (5.8)$$

El estado estable de la red representa el patrón recuperado a partir del patrón de prueba  $x$ .

Es posible que durante la fase de recuperación la red de Hopfield converja a estados estables que no se corresponden con los patrones almacenados. Dichos estados reciben el nombre de estados esprios y, generalmente, vienen producidos por el almacenamiento de un número elevado de patrones. Dicho problema puede ser ligeramente corregido aplicando la regla de Hebb al revés [Hopfield et al., 1983].

### 5.2.2. Función energía

Se ha visto en capítulos anteriores que para otros tipos de redes, como el PERCEPTRON multicapa, los mapas auto-organizados de Kohonen, etc, existe una función error o función energía que describe el comportamiento de dichas redes y permite entender su funcionamiento. En el caso de la red de Hopfield también existe dicha función, la cual será analizada en este apartado.

Dada una red de Hopfield con  $n$  neuronas y con conexiones  $W = (w_{ij})$ , siendo  $W$  una matriz simétrica y con ceros en la diagonal, la función energía asociada a dicha red viene dada por la siguiente ecuación:

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j \neq i}^n w_{ij} s_i s_j + \sum_{i=1}^n u_i s_i \quad (5.9)$$

Separamos la contribución de una neurona  $k$  a la función energía dada por la expresión anterior, se puede escribir que:

$$E = -\frac{1}{2} \sum_{i \neq k} \sum_{j \neq k} w_{ij} s_i s_j + \sum_{i \neq k} u_i s_i - \frac{1}{2} s_k \sum_{i \neq k} w_{kj} s_j - \frac{1}{2} s_k \sum_i w_{ik} s_i + u_k s_k \quad (5.10)$$

El cambio de estado de la neurona  $k$  de la red, denotado como  $\Delta s_k = s_k(t+1) - s_k(t)$ , produce un cambio en la función energía, denotado como  $\Delta E = E(t+1) - E(t)$ , el cual, de acuerdo con la Ecuación (5.10), adopta la siguiente expresión:

$$\Delta E = -\frac{1}{2} \Delta s_k \sum_j w_{kj} s_j - \frac{1}{2} \Delta s_k \sum_i w_{ik} s_i + u_k \Delta s_k \quad (5.11)$$

Debido a que las conexiones de la red de Hopfield son simétricas, se puede escribir:

$$\Delta E = -\Delta s_k \left( \sum_j w_{jk} s_j - u_k \right) \quad (5.12)$$

Por tanto, cuando los estados de la red cambian siguiendo las ecuaciones (5.2), (5.3) y (5.4),  $\Delta E$  es siempre negativo, por lo que la función  $E$  es monótona decreciente respecto a los estados de la red. De este modo, el punto estable de la red de Hopfield se corresponde con un mínimo local de la función energía. De hecho, la manera de modificar los estados de la red en la fase de recuperación (ecuaciones 5.2, 5.3 y 5.4) no es más que el resultado de aplicar el método de descenso del gradiente para encontrar un mínimo de la función energía dada por la Ecuación (5.9).

Debido a que un mínimo local de la función energía se corresponde con un punto estable de la red de Hopfield, todo problema de optimización que pueda escribirse en términos de la función energía (Ecuación 5.9) puede ser, en principio, resuelto con la red de Hopfield asociada a dicha función. Así, por ejemplo, una aplicación interesante de la red de Hopfield, desde el punto de vista teórico, es el problema del viajante, en el que se busca la distancia mínima entre  $n$  ciudades. En [Hopfield and Tank, 1985] se formula dicho problema en términos de la función energía dada por la Ecuación (5.9).

### 5.3. Redes parcialmente recurrentes

En esta sección se van a tratar las llamadas redes parcialmente recurrentes, las cuales se caracterizan porque son redes multicapa, en las que se introducen sólo unas pocas conexiones recurrentes. Dichas conexiones permiten recordar el estado o nivel de activación de ciertas neuronas de la red en un pasado reciente.

Generalmente, cuando se habla de redes parcialmente recurrentes existe un grupo de neuronas especiales en la capa de entrada, llamadas neuronas de contexto o neuronas de estado. De este modo, en la capa de entrada de las redes parcialmente recurrentes se distinguen dos tipos de neuronas, aquellas que actúan como entrada propiamente dicha, recibiendo las señales del exterior, y las neuronas de contexto, como se muestra en la Figura 5.3. Estas neuronas son las receptoras de las conexiones recurrentes y funcionan como una memoria de la red donde se almacenan las activaciones de las neuronas de una cierta capa de la red en el instante o iteración anterior.

Concatenando las activaciones de las neuronas de entrada y de las neuronas de contexto, una red parcialmente recurrente se puede ver como una red multicapa, al igual que el PERCEPTRON multicapa. Por tanto, en las redes parcialmente recurrentes el cálculo de las activaciones de todas las neuronas de la red se realiza como en una red multicapa sin recurrencias, es decir, desde la capa de entrada a la capa de salida, pasando por la capa oculta.

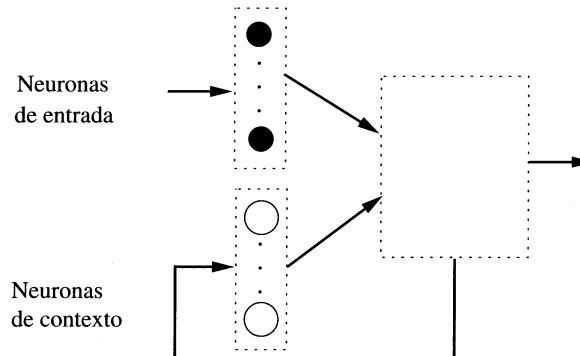


Figura 5.3: Esquema de una red parcialmente recurrente

Las conexiones recurrentes en las redes parcialmente recurrentes suelen ser conexiones uno a uno, es decir, de una cierta neurona de la red a una única neurona de contexto. En el caso de que dicha conexión lleve un parámetro o peso asociado, generalmente, se mantiene constante y no está sometido a aprendizaje. Esto hace que el aprendizaje en las redes parcialmente recurrentes se pueda llevar a cabo utilizando el algoritmo de retropropagación para redes multicapa con conexiones hacia adelante, ya que los únicos parámetros sometidos a aprendizaje son los pesos asociados a las conexiones que van de la capa de entrada a la capa oculta y de la capa oculta a la capa de salida.

Las redes parcialmente recurrentes más conocidas son la red de Elman y la red de Jordan. Su diferencia fundamental radica en que las neuronas de contexto reciben copias de diferentes capas de la red. Así, en la red de Elman, las neuronas de contexto reciben una copia de las neuronas ocultas, y en la red Jordan, reciben una copia de las neuronas de la capa de salida y de sí mismas.

A continuación se describe la arquitectura y características de la red de Jordan y de la red de Elman.

#### ■ Red de Jordan

El modelo de red de Jordan propuesto por Jordan en 1986 ([Jordan, 1986a], [Jordan, 1986b]) se caracteriza porque las neuronas de contexto reciben una copia de las neuronas de salida de la red y de ellas mismas, como se muestra en la Figura 5.4. En este caso, las conexiones recurrentes de la capa de salida a las neuronas de contexto llevan un parámetro asociado,  $\mu$ , que, generalmente, toma un valor constante positivo y menor que 1.

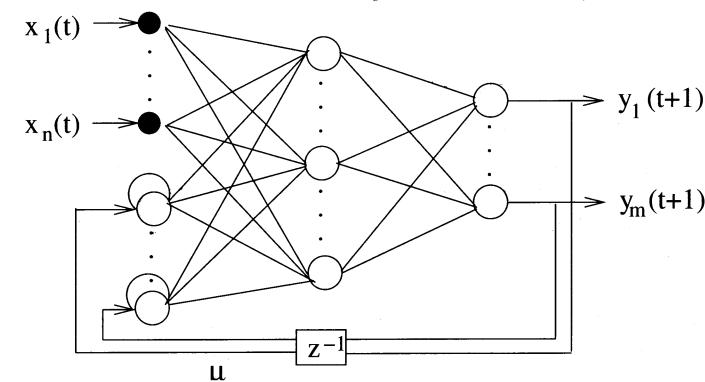


Figura 5.4: Red de Jordan

Cada neurona de contexto recibe una conexión de una neurona de salida y de sí misma, por lo que la activación de las neuronas de contexto en el instante de tiempo o iteración  $t$ , denotada como  $c_i(t)$ , viene dada por:

$$c_i(t) = \mu c_i(t-1) + y_i(t-1) \text{ para } i = 1, 2, \dots, m \quad (5.13)$$

donde  $m$  es el número de salidas de la red y  $y(t-1) = (y_1(t-1), y_2(t-1), \dots, y_m(t-1))$  es el vector salida de la red en el instante de tiempo  $t-1$ .

Como se comentó anteriormente, el resto de las activaciones de la red se calculan como en una red multicapa con conexiones hacia adelante; basta considerar como entrada total a la red en el instante de tiempo  $t$  el vector  $u(t)$  que resulta de la concatenación de las activaciones de las neuronas de entrada y de las neuronas de contexto:

$$u(t) = (x_1(t), \dots, x_n(t), c_1(t), \dots, c_m(t)) \quad (5.14)$$

donde  $x_i(t)$  representan las señales que la red recibe del exterior.

Tanto las neuronas ocultas de la red como las neuronas de salida tienen función de activación sigmoidal al igual que el PERCEPTRON multicapa.

Las neuronas de contexto poseen, sin embargo, funciones de activación lineal, como se observa en la Ecuación (5.13). Esto hace que la activación de las neuronas de contexto se pueda desarrollar en el tiempo del siguiente modo:

$$\begin{aligned} c_i(t) &= \mu^2 c_i(t-2) + \mu y_i(t-2) + y_i(t-1) = \dots = \\ &= \mu^{t-2} y_i(1) + \mu^{t-3} y_i(2) + \dots + \mu y_i(t-2) + y_i(t-1) \end{aligned}$$

y se obtiene entonces la siguiente expresión:

$$c_i(t) = \sum_{j=1}^{t-1} \mu^{j-1} y_i(t-j) \text{ para } i = 1, 2, \dots, m \quad (5.15)$$

Por tanto, el parámetro  $\mu$  que aparece en la arquitectura de la red de Jordan dota de cierta inercia a las neuronas de contexto de dicha red. En la expresión anterior (Ecuación 5.15), se observa que las neuronas de contexto acumulan las salidas de la red en todos los instantes anteriores de tiempo y el valor del parámetro  $\mu$  determina la sensibilidad de las neuronas de contexto para retener dicha información. Así, valores cercanos a 1 permiten memorizar estados muy alejados del tiempo actual, y a medida que el valor de  $\mu$  se acerca a cero, dichos estados tienen una menor presencia en la activación actual de las neuronas de contexto.

#### ■ Red de Elman

La red de Elman [Elman, 1990] se caracteriza porque las neuronas de contexto reciben una copia de las neuronas ocultas de la red, como se observa en la Figura 5.5, y dichas conexiones no llevan asociado ningún parámetro. Existen, por tanto, tantas neuronas de contexto como neuronas ocultas tenga la red. De este modo, la activación de las neuronas de contexto viene dada por:

$$c_i(t) = a_i(t-1) \text{ para } i = 1, 2, \dots, r \quad (5.16)$$

donde  $r$  es el número de neuronas ocultas de la red y  $a_i(t-1)$  son las activaciones de dichas neuronas en el instante  $t-1$ .

Al igual que en la red de Jordan, el resto de las activaciones de la red se calculan como en una red multicapa con conexiones hacia adelante, considerando como entrada total a la red el vector  $u(t)$  dado por la Ecuación (5.14).

En el caso de la red de Jordan, las conexiones recurrentes permiten que las neuronas ocultas de la red vean las salidas de la red en instantes anteriores de tiempo. Para la red de Elman, sin embargo, las conexiones recurrentes hacen que las neuronas ocultas contengan información sobre las señales de entrada que proceden del exterior en el instante inmediatamente anterior.

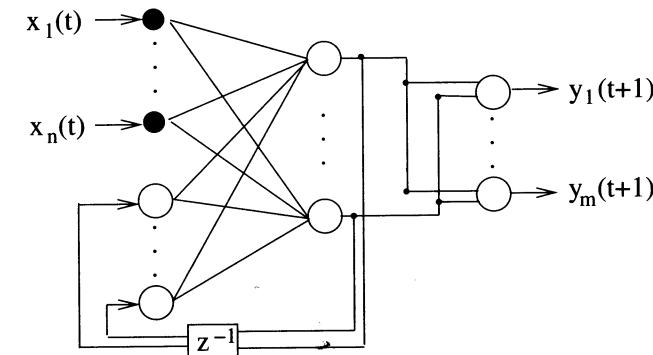


Figura 5.5: Red de Elman

Aunque las redes de Jordan y Elman son los modelos de redes parcialmente recurrentes más conocidos, es posible diseñar y utilizar otros modelos diferentes, como, por ejemplo, incorporar conexiones de las neuronas de contexto a ellas mismas en el modelo de Elman; considerar neuronas de contexto que no sólo memoricen el estado de ciertas neuronas de la red en el instante inmediatamente anterior, sino en un intervalo de tiempo, etc..

El mecanismo para realizar el aprendizaje, tanto de la red de Jordan y de Elman, como de cualquier red parcialmente recurrente, se puede resumir en los siguientes pasos:

1. Se inicializan las neuronas de contexto de la red parcialmente recurrente en el instante de tiempo  $t = 0$ .
2. Se presenta a la red en el instante de tiempo  $t$  el patrón de entrada procedente del exterior,  $x(t) = (x_1(t), \dots, x_n(t))$ , que junto con la activación de las neuronas de contexto en ese instante,  $c_i(t)$ , forman el vector de entrada total a la red,  $u(t)$  (Ecuación 5.14).
3. El vector  $u(t)$  se propaga hacia la salida de la red, obteniendo así la salida de la red en dicho instante de tiempo.
4. Se aplica la regla delta generalizada para modificar los pesos de la red.
5. Se incrementa la variable tiempo en una unidad y se vuelve al paso 2.

#### 5.4. Redes totalmente recurrentes

En esta sección se van a tratar redes de neuronas en las que no existe restricción en la conectividad, las cuales, para diferenciarlas de las redes parcialmente recurrentes, se llaman **redes totalmente recurrentes**.

Las redes totalmente recurrentes se caracterizan porque sus neuronas reciben como entradas la activación del resto de las neuronas de la red, así como su propia activación. De este modo, si  $a_i(t)$  representa la activación de una neurona  $i$  de la red en el instante de tiempo  $t$ , su valor viene dado por la siguiente expresión:

$$a_i(t) = f_i \left( \sum_{j \in A \cup B} w_{ij} a_j(t-1) \right) \quad (5.17)$$

donde  $f_i$  es la función de activación,  $A$  es el conjunto de neuronas de entrada a la red,  $B$  representa el resto de las neuronas de la red y  $w_{ij}$  representa el peso de la conexión de la neurona  $j$  a la neurona  $i$ . En la expresión anterior se ha supuesto que todas las neuronas de la red están conectadas entre sí, aunque podrían existir ciertas restricciones.

En las redes totalmente recurrentes, los parámetros o pesos de las conexiones recurrentes no se consideran fijos, sino que se suelen someter al proceso de adaptación o aprendizaje, produciéndose así un aumento considerable del número de parámetros ajustables de la red. Esto significa, por una parte, aumentar la capacidad de representación de la red, aunque, por otra parte, la existencia de ciclos o conexiones recurrentes en la red complica, generalmente, su aprendizaje.

El algoritmo de retropropagación estudiado en el Capítulo 3 para entrenar el PERCEPTRON multicapa no puede aplicarse directamente a redes recurrentes, pues los pesos en estas redes poseen una distribución diferente. El entrenamiento de las redes recurrentes se puede llevar a cabo utilizando dos métodos o algoritmos de aprendizaje diferentes: el *algoritmo de retropropagación a través del tiempo* (Back-Propagation Through Time) y el *algoritmo de aprendizaje recurrente en tiempo real* (Real-Time Recurrent Learning), los cuales serán descritos a continuación. Ambos métodos no son más que modificaciones y extensiones del algoritmo de retropropagación para redes con conexiones recurrentes.

#### 5.4.1. Algoritmo de retropropagación a través del tiempo

El algoritmo de retropropagación a través del tiempo se basa en la idea de que para cada red recurrente es posible construir una red multicapa con conexiones hacia adelante y con idéntico comportamiento; basta desarrollar en el tiempo la red recurrente ([Werbos, 1990], [Pearlmutter, 1989]). Para ilustrar esta idea, se considera la red recurrente que se muestra en la Figura 5.6. Se trata de una red simple, con dos neuronas totalmente conectadas, y la activación de cada una de las neuronas viene dada por:

$$a_1(t) = f(w_{11}a_1(t-1) + w_{12}a_2(t-1)) \quad (5.18)$$

$$a_2(t) = f(w_{21}a_1(t-1) + w_{22}a_2(t-1)) \quad (5.19)$$

Desarrollando dichas activaciones en el tiempo se obtiene que:

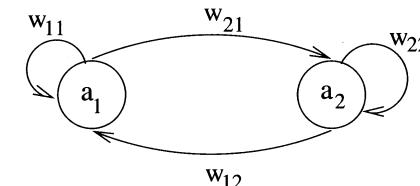


Figura 5.6: Arquitectura de red recurrente con dos neuronas

$$a_i(1) = f(w_{i1}a_1(0) + w_{i2}a_2(0))$$

$$a_i(2) = f(w_{i1}a_1(1) + w_{i2}a_2(1))$$

$$\dots$$

$$a_i(n) = f(w_{i1}a_1(n-1) + w_{i2}a_2(n-1))$$

para  $i = 1, 2$

Por tanto, las activaciones hasta el instante  $n$  de la red recurrente que se muestra en la Figura 5.6 son equivalentes a las activaciones de las neuronas de la red multicapa con conexiones hacia adelante que se muestra en la Figura 5.7. De este modo, la red recurrente se puede representar mediante una red con conexiones hacia adelante, la cual se obtiene añadiendo una nueva capa por cada unidad de tiempo.

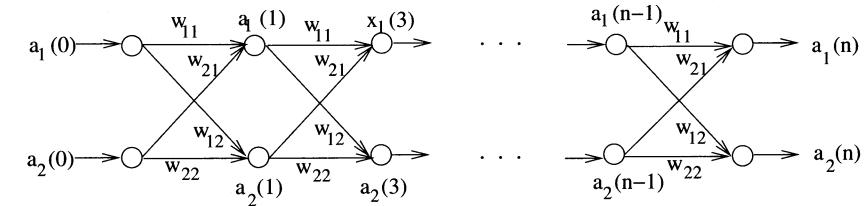


Figura 5.7: Red recurrente desarrollada en el tiempo

Para la aplicación del algoritmo de retropropagación a través del tiempo, es necesario definir una época o intervalo de tiempo en el cual la red recurrente será desarrollada. Dado  $[n_0, n_1]$  el intervalo de tiempo o época en el que la red recurrente se desarrolla, se define el error cometido por la red en dicho intervalo como:

$$E(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{i \in S} e_i^2(n) \quad (5.20)$$

donde  $S$  es el conjunto de índices que representan las neuronas de salida de la red, es decir, neuronas para las cuales se dispone de una salida deseada; y  $e_i(n)$

es el error que comete la red en el instante  $n$ , medido como la diferencia entre la salida de la red y la salida deseada. Los pesos de la red totalmente recurrente se van a modificar siguiendo la dirección negativa del gradiente del error dado por la Ecuación (5.20), por lo que el cambio del peso  $w_{ij}$  de la neurona  $j$  a la neurona  $i$  viene dado por:

$$\Delta w_{ij} = -\alpha \frac{\partial E(n_0, n_1)}{\partial w_{ij}} \quad (5.21)$$

En términos generales, el aprendizaje de la red recurrente mediante el algoritmo de retropropagación a través del tiempo implica el cálculo de la derivada de  $E(n_0, n_1)$  respecto de las conexiones de la red. Para ello, se aplica el algoritmo de retropropagación estudiado en el Capítulo 3 a la red multicapa desarrollada en el tiempo (Figura 5.7). Al aplicar dicho algoritmo es necesario tener en cuenta los siguientes aspectos:

- Cada capa de la red desarrollada puede tener neuronas para las cuales existen salidas deseadas, ya que las neuronas de salida de la red recurrente también se replican tantas veces como indique la época o intervalo de tiempo definido.
- Los pesos coinciden para todas las capas de la red multicapa desarrollada, por lo que al calcular el gradiente con respecto a un peso hay que hacerlo individualmente para cada capa y posteriormente sumarlos.

A continuación, se detallan los pasos para la aplicación del algoritmo de retropropagación a través del tiempo para el aprendizaje de una red totalmente recurrente.

1. Dado un tiempo inicial  $n_0$ , la red totalmente recurrente se desarrolla en el intervalo  $[n_0, n_1]$ , obteniendo una red multicapa con conexiones hacia adelante.
2. Se calculan y almacenan las activaciones de todas las neuronas de la red multicapa.

Se denota como  $f(v_i(n))$  la activación de la neurona  $i$  de la capa  $n$  ( $n = n_0, \dots, n_1$ ) en la red multicapa, donde  $f$  es la función de activación y  $v_i(n)$  es el nivel total de activación que recibe la neurona  $i$ .

3. La aplicación del algoritmo de retropropagación a la red multicapa equivalente implica el cálculo de los valores  $\delta$  para cada una de las capas de la red multicapa, empezando por la última capa, capa  $n_1$ , hasta llegar a la primera capa oculta, capa  $n_0 + 1$ .

Denotando por  $\delta_i(n)$  el valor  $\delta$  asociado a la neurona  $i$  de la capa  $n$  para todo  $i \in S$  y teniendo en cuenta el mecanismo para calcular dichos valores cuando se utiliza el algoritmo de retropropagación, se obtiene que:

$$\delta_i(n) = \begin{cases} f'(v_i(n))e_i(n) & \text{si } n = n_1 \\ f'(v_i(n))(e_i(n) + \sum_{j \in S} w_{ji}\delta_j(n+1)) & \text{si } n_0 < n < n_1 \end{cases} \quad (5.22)$$

donde  $f'(\cdot)$  es la derivada de la función de activación respecto a su argumento.

Aplicando dicha ecuación, se obtienen  $\delta(n_1), \delta(n_1 - 1), \dots, \delta(n_0 + 1)$ . Por tanto, es necesario calcular tantos valores  $\delta$  como el número de instantes de tiempo que contiene el intervalo o época  $[n_0, n_1]$ .

4. Una vez que se obtiene el valor  $\delta$  de la capa  $n_0 + 1$ , el cambio o ajuste para el peso  $w_{ij}$  viene dado por la siguiente expresión:

$$\Delta w_{ij} = \alpha \frac{\partial E(n_0, n_1)}{\partial w_{ij}} = \alpha \sum_{n=n_0+1}^{n_1} \delta_i(n)a_j(n-1) \quad (5.23)$$

donde  $\alpha$  es la razón de aprendizaje y  $a_j(n-1)$  es la entrada  $j$  a la neurona  $i$ .

5. Con los nuevos pesos, se repite el proceso para el instante de tiempo  $n_0 + 1$ , preparando la red para una nueva época.

El algoritmo de retropropagación a través del tiempo no es precisamente un método adecuado para aplicaciones en tiempo real, es decir, para aplicaciones que requieren una adaptación continua de la red recurrente. Este algoritmo involucra un coste computacional para el cálculo de los valores  $\delta$  que podría ser elevado, así como la necesidad de almacenar en memoria el estado de la red desarrollada en el tiempo. En [Willians and Peng, 1990] se describe una variante de este algoritmo que, desde el punto de vista computacional, es más eficiente. Dicho algoritmo se basa en la idea de que sucesos ocurridos muy atrás en el tiempo no tienen por qué influir en sucesos actuales, de modo que los autores proponen utilizar los gradientes sólo hasta un pasado más reciente.

#### 5.4.2. Algoritmo de aprendizaje recurrente en tiempo real

Como se ha comentado en el apartado anterior, el algoritmo de retropropagación a través del tiempo presenta limitaciones para su aplicación en tiempo real. En este apartado se va a presentar otro algoritmo para entrenar redes recurrentes, el cual permite que el aprendizaje pueda realizarse en tiempo real. Este algoritmo se conoce como algoritmo recurrente en tiempo real [Willians and Zipser, 1989] y, generalmente, se asocia a una arquitectura de red recurrente, llamada *red recurrente en tiempo real*.

La red recurrente en tiempo real (ver Figura 5.8) está compuesta por  $N$  neuronas y  $M$  neuronas de entrada que reciben las señales del exterior. De las  $N$  neuronas, algunas se considerarán neuronas de salida, es decir, neuronas

para las que se dispone de una salida deseada. Se trata de una red con  $N \cdot N$  conexiones recurrentes, pues las  $N$  neuronas se conectan con todas las demás e incluso con ellas mismas; y  $M \cdot N$  conexiones hacia adelante, pues cada neurona de entrada se conecta con el resto de las neuronas de la red, como se observa en la Figura 5.8. Por tanto, la matriz de pesos de la red,  $W = (w_{ij})$ , es una matriz de orden  $N \times (M + N)$ , donde  $w_{ij}$  representa la conexión de la neurona  $j$  a la neurona  $i$ , para  $j = 1, \dots, M + N$  y para  $i = 1, \dots, N$ .

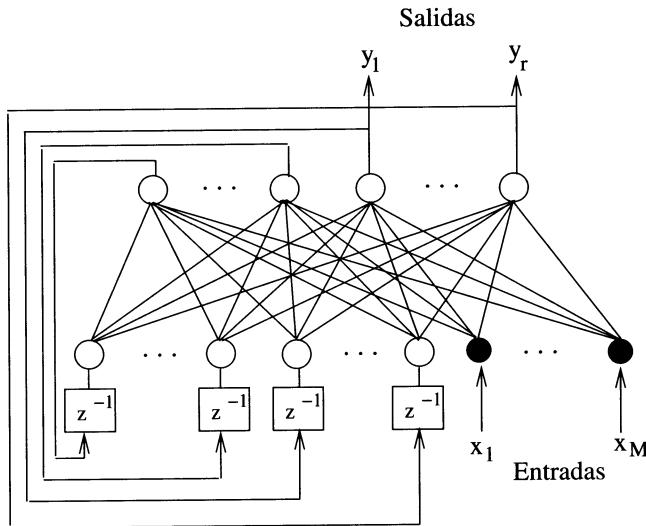


Figura 5.8: Red recurrente en tiempo real

Sea  $A$  el conjunto de índices que representan las neuronas de entrada y sea  $B$  el conjunto de índices para el resto de las neuronas de la red. Según la red definida anteriormente, el cardinal de  $A$  es  $M$  y el cardinal de  $B$  es  $N$ . Sea  $x(n) = (x_i(n))_{i \in A}$  el vector de entrada a la red en el instante de tiempo  $n$  y sea  $y(n) = (y_i(n))_{i \in B}$  el vector formado por las activaciones de las  $N$  neuronas de la red en el instante de tiempo  $n$ . Se define entonces el vector  $u(n)$  como la concatenación de dichos vectores, de manera que la coordenada  $i$  de dicho vector viene dada por:

$$u_i(n) = \begin{cases} x_i(n) & \text{si } i \in A \\ y_i(n) & \text{si } i \in B \end{cases} \quad (5.24)$$

Las activaciones de las neuronas de la red,  $y_i(n)$ , vienen dadas por:

$$y_i(n) = f(v_i(n-1)) \text{ para } i \in B \quad (5.25)$$

siendo  $f$  la función de activación y  $v_i(n-1)$  el nivel total de activación que recibe la neurona  $i$ , es decir, la suma de los productos de las entradas a la neurona por sus correspondiente conexiones:

$$v_i(n-1) = \sum_{j \in A \cup B} w_{ij} u_j(n-1) \quad (5.26)$$

La aplicación del algoritmo de aprendizaje recurrente en tiempo real a la red definida anteriormente consiste en ajustar los pesos de la red siguiendo la dirección negativa del gradiente del error computado en las neuronas que actúan como salida de la red. Debido a que la activación de una neurona de salida en un instante  $n$  depende de las activaciones de todas las neuronas de la red en el instante anterior  $n-1$  (ver Ecuaciones 5.25 y 5.26), incluida ella misma, en el cálculo de la derivada de la neurona de salida con respecto a un peso de la red interviene también la derivada del resto de las activaciones en el instante anterior respecto a dicho peso, por lo que el algoritmo de retropropagación estudiado en el Capítulo 3 no puede ser aplicado directamente. A continuación, se va a desarrollar la extensión de dicho algoritmo para redes recurrentes en tiempo real, dando lugar al llamado algoritmo de aprendizaje recurrente en tiempo real.

Sea  $S$  el conjunto de índices que representan las neuronas de salida de la red, es decir, las neuronas para las cuales existe una salida deseada  $d_i(n)$ . Se define el error cometido por la red en el instante  $n$  como:

$$E(n) = \frac{1}{2} \sum_{i \in S} e_i^2(n) \quad (5.27)$$

donde  $e_i(n)$  es el error para la neurona de salida  $i$ :

$$e_i(n) = d_i(n) - y_i(n) \text{ para } i \in S \quad (5.28)$$

La ley para modificar los pesos de la red siguiendo la dirección negativa del gradiente del error adopta la siguiente expresión:

$$w_{kl}(n) = w_{kl}(n-1) - \alpha \frac{\partial E(n)}{\partial w_{kl}} \text{ para } k \in B, l \in A \cup B \quad (5.29)$$

donde  $w_{kl}$  representa el peso de la conexión de la neurona  $l$  a la neurona  $k$ .

Derivando la Ecuación (5.27) respecto al peso  $w_{kl}$ , se obtiene que:

$$\frac{\partial E(n)}{\partial w_{kl}} = - \sum_{i \in S} e_i(n) \frac{\partial y_i(n)}{\partial w_{kl}} \quad (5.30)$$

Aplicando la regla de la cadena a la Ecuación (5.25), la derivada de la activación de la neurona  $i$ ,  $y_i(n)$ , respecto al peso es:

$$\frac{\partial y_i(n)}{\partial w_{kl}} = f'(v_i(n-1)) \frac{\partial v_i(n-1)}{\partial w_{kl}} \quad (5.31)$$

Derivando la expresión de  $v_i(n-1)$  dada por la Ecuación (5.26), se obtiene:

$$\frac{\partial v_i(n-1)}{\partial w_{kl}} = \sum_{j \in A \cup B} \frac{\partial}{\partial w_{kl}} (w_{ij} u_j(n-1)) = \quad (5.32)$$

$$= \sum_{j \in A \cup B} \left[ w_{ij} \frac{\partial u_j(n-1)}{\partial w_{kl}} + \frac{\partial w_{ij}}{\partial w_{kl}} u_j(n-1) \right]$$

Por tanto,

$$\frac{\partial v_i(n-1)}{\partial w_{kl}} = \sum_{j \in A \cup B} w_{ij} \frac{\partial u_j(n-1)}{\partial w_{kl}} + \delta_{ki} u_l(n-1) \quad (5.33)$$

donde  $\delta_{ki}$  es la función delta de Kronecker, que vale 1 cuando  $i = k$  y 0 en cualquier otro caso.

Sustituyendo la Ecuación (5.33) en la Ecuación (5.31) se obtiene que:

$$\frac{\partial y_i(n)}{\partial w_{kl}} = f'(v_i(n-1)) \sum_{j \in A \cup B} w_{ij} \frac{\partial u_j(n-1)}{\partial w_{kl}} + \delta_{ki} u_l(n-1) \quad (5.34)$$

Para obtener la derivada de  $y_i(n)$  respecto de  $w_{kl}$ , sólo falta calcular  $\frac{\partial u_j(n-1)}{\partial w_{kl}}$ . Según se define el vector  $u$  en la Ecuación (5.24), la parte del vector correspondiente a las variables de entrada  $x_j(n-1)$  no depende del peso  $w_{kl}$ , por lo que su derivada es cero. Sin embargo, las activaciones  $y_j(n-1)$  sí dependen de dicho peso, debido a las conexiones recurrentes. Por tanto:

$$\frac{\partial u_j(n-1)}{\partial w_{kl}} = \begin{cases} 0 & \text{si } j \in A \\ \frac{\partial y_j(n-1)}{\partial w_{kl}} & \text{si } j \in B \end{cases} \quad (5.35)$$

Por tanto,

$$\frac{\partial y_i(n)}{\partial w_{kl}} = f'(v_i(n-1)) \sum_{j \in B} w_{ij} \frac{\partial y_j(n-1)}{\partial w_{kl}} + \delta_{ki} u_l(n-1) \quad (5.36)$$

Denotando  $p_{kl}^i(n) = \frac{\partial y_i(n)}{\partial w_{kl}}$ , la Ecuación (5.36) puede escribirse de la forma:

$$p_{kl}^i(n) = f'(v_i(n-1)) \sum_{j \in B} w_{ij} p_{kl}^j(n-1) + \delta_{ki} u_l(n-1) \quad (5.37)$$

para  $k \in B$ ,  $l \in A \cup B$  e  $i \in S$

Entonces, la derivada del error  $E(n)$  (Ecuación 5.27) viene dada por:

$$\frac{\partial E(n)}{\partial w_{kl}} = - \sum_{i \in S} e_i(n) p_{kl}^i(n) \quad (5.38)$$

donde  $p_{kl}^i(n)$  es la salida en el instante de tiempo  $n$  del sistema dinámico definido por la Ecuación (5.37), con valores o condiciones iniciales  $p_{kl}^i(0) = 0$ , y así queda completado el desarrollo de la ley de aprendizaje dada por la Ecuación (5.29).

Mediante este algoritmo de aprendizaje, los pesos se adaptan en cada instante de tiempo  $n$  utilizando las salidas del sistema dinámico dado anteriormente,  $p_{kl}^i(n)$ , variables que serán utilizadas en el siguiente instante de tiempo  $n+1$ . A diferencia del algoritmo de retropropagación a través del tiempo, el algoritmo de aprendizaje recurrente puede aplicarse en tiempo real, pues no necesita almacenar en memoria el estado de la red durante un intervalo de tiempo, lo cual lo hace más eficiente.

Aunque en la literatura el algoritmo de aprendizaje recurrente en tiempo real se asocia a la red recurrente en tiempo real que se muestra en la Figura 5.8, la idea básica de este algoritmo puede aplicarse a cualquier red recurrente; basta tener en cuenta en el cálculo del gradiente del error las conexiones recurrentes que contenga la red. Así, por ejemplo, en [Puskoris and Feldkamp, 1994] y [Parlos et al., 1994] se desarrollan algoritmos de este tipo para entrenar redes multicapa con conexiones recurrentes entre las neuronas de las capas ocultas.

### Desarrollo del algoritmo de aprendizaje recurrente en tiempo real para una red recurrente con una neurona de entrada y una neurona de salida con una conexión recurrente

Como se ha comentado, la idea base del algoritmo de aprendizaje recurrente consiste en tener en cuenta que en el cálculo de la derivada de la activación de una neurona en un instante de tiempo  $n$  respecto a un peso, interviene también la derivada de la activación de dicha neurona en el instante anterior, pues dicha activación también depende del peso.

Para ilustrar esta idea, se considera la red recurrente que se muestra en la Figura 5.9, la cual posee una neurona de salida con una conexión recurrente hacia ella misma. A continuación, se va a desarrollar el algoritmo de aprendizaje recurrente en tiempo real para esta red.

La activación de la neurona de la red viene dada por:

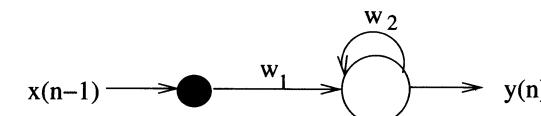


Figura 5.9: Arquitectura de red recurrente

$$y(n) = f(w_1 x(n-1) + w_2 y(n-1)) \quad (5.39)$$

Siguiendo la dirección negativa del gradiente del error, la ley para adaptar el peso de la conexión recurrente viene dada por:

$$w_2(n) = w_2(n-1) - \alpha \frac{\partial E(n)}{\partial w_2} \quad (5.40)$$

Derivando el error  $E(n) = \frac{1}{2} e(n)^2 = \frac{1}{2} (d(n) - y(n))^2$  cometido por la red, se obtiene que:

$$\frac{\partial E(n)}{\partial w_2} = -e(n) \frac{\partial y(n)}{\partial w_2} \quad (5.41)$$

Teniendo en cuenta la expresión de la salida de la red y considerando que la activación en el instante anterior  $y(n-1)$  también depende del peso  $w_2$ , se tiene que:

$$\frac{\partial y(n)}{\partial w_2} = f'(w_1x(n-1) + w_2y(n-1)) \left( w_2 \frac{\partial y(n-1)}{\partial w_2} + y(n-1) \right) \quad (5.42)$$

Denotando  $p_2(n) = \frac{\partial y(n)}{\partial w_2}$ , la expresión anterior puede escribirse de la forma:

$$p_2(n) = f'(w_1x(n-1) + w_2y(n-1)) \left( w_2p_2(n-1) + y(n-1) \right) \quad (5.43)$$

Por tanto, la ley para modificar el peso  $w_2$  viene dada por:

$$w_2(n) = w_2(n-1) + \alpha e(n)p_2(n) \quad (5.44)$$

siendo  $p_2(n)$  la salida del sistema dinámico dado por la Ecuación (5.43).

En el caso de la conexión no recurrente en la red,  $w_1$ , la ley para su adaptación viene dada por la siguiente ecuación:

$$w_1(n) = w_1(n-1) + \alpha e(n)p_1(n) \quad (5.45)$$

donde:

$$p_1(n) = f'(w_1x(n-1) + w_2y(n-1))(x(n-1) + w_2p_1(n)) \quad (5.46)$$

Basta tener en cuenta que

$$\frac{\partial y(n)}{\partial w_1} = f'(w_1x(n-1) + w_2y(n-1))(x(n-1) + w_2 \frac{\partial y(n-1)}{\partial w_1}) \quad (5.47)$$

## Capítulo 6

### APRENDIZAJE NO SUPERVISADO

#### 6.1. Características básicas

Las Redes de Neuronas Artificiales con aprendizaje no supervisado son aquellas que no necesitan de un *profesor* o *supervisor* externo para realizar su aprendizaje. Son capaces de modificar sus parámetros internamente, adaptándose al entorno de la mejor manera posible. Biológicamente, existen claros ejemplos de aprendizaje no supervisado, así como de supervisado.

La no supervisión consiste en que la red descubra por sí sola características, regularidades, correlaciones o categorías en los datos de entrada, y se obtengan de forma codificada a la salida. Por tanto, se puede decir que estas unidades y conexiones muestran cierto grado de auto-organización.

El aprendizaje no supervisado sólo consigue resultados útiles si en los datos de entrada existe cierto tipo de redundancia. Sin redundancia sería imposible encontrar patrones o características en los datos, lo cual se asemeja necesariamente a ruido aleatorio. En este sentido, la redundancia es fuente de conocimiento [Barlow, 1989]. Dicho de una manera más precisa, el contenido total de información del conjunto de los datos de entrada es menor que el máximo que podría ser soportado por el mismo canal; esa diferencia es la redundancia.

El tipo de patrón que una red con aprendizaje no supervisado puede detectar depende de su arquitectura. Pero es interesante conocer qué tipo de proceso puede realizar la red, qué problemas puede resolver, qué representa la salida de la red. Existen varias posibilidades:

1. **Familiaridad.** Es el caso en el que mediante un único valor continuo de salida se pondera el grado de similitud entre la entrada y un *valor tipo* o media de una serie de valores presentados con anterioridad. La red, a su vez, puede ir aprendiendo lo que se entiende por *valor tipo*.

2. **Análisis de las componentes principales.** Se trata de detectar cuáles de las componentes del conjunto de entrada caracterizan en mayor grado al conjunto de datos, de forma que las demás pudieran eliminarse sin una significativa pérdida de información.
3. **Agrupamiento.** A partir de un conjunto de entrada se desea conocer si hay un conjunto de datos que están bien representados por un elemento patrón, y de qué manera se agupan los datos a estos patrones representativos que ha encontrado la red. Es decir, si se puede dividir el conjunto de datos en diferentes clases, y decidir a qué clase pertenece cada dato y qué caracteriza a cada una de las clases.
4. **Prototipado.** Igual que en el caso anterior, pero en vez de obtenerse como resultado a qué clase pertenece el dato de entrada, se obtiene un prototipo o ejemplar de la clase a que pertenece dicho dato de entrada.
5. **Codificación.** Se obtiene, a la salida de la red, una versión codificada del dato de entrada, es decir, un dato de menor dimensión que mantenga el máximo de información que le sea posible. Esto podría utilizarse para la compresión de datos antes de transmitirse por un canal de un ancho de banda limitado, por el cual la señal original no podría ser transmitida; siempre y cuando exista la posibilidad de construir una red capaz de restaurar el dato original.
6. **Extracción y relación de características.** La idea es realizar un mapa topológico de los datos de entrada, a través del diseño de la red, de tal forma que patrones de entrada parecidos, produzcan respuestas similares en células cercanas. De esta forma se espera que si existe una organización global de los patrones de entrada, ésta sea puesta de relieve a la salida.

Estos ejemplos no tienen por qué ser disjuntos, y se pueden combinar de varias maneras. Por ejemplo, el problema de codificación podría realizarse mediante análisis de componentes principales, o por *clustering*, lo cual recibe el nombre de cuantización de vectores, *vector quantization*. También se puede utilizar el método de análisis de las componentes principales para reducir la dimensión de los datos, como paso previo al *clustering* o detector de características.

A veces los métodos de aprendizaje no supervisado pueden dar mejores resultados incluso en situaciones en que es posible aplicar métodos de aprendizaje supervisado:

- El método de RETROPROPAGACIÓN multicapa es sumamente lento, en parte debido a que los mejores pesos de una capa dependen de todos los otros pesos de todas las demás capas. Esto se podría evitar, en alguna medida, mediante un método no supervisado, o una solución híbrida, en la cual algunas capas se autoorganicen justo antes de pasar sus datos a una red supervisada.

- Incluso después del entrenamiento, en una red con aprendizaje supervisado, podría ser conveniente realizar algún tipo de aprendizaje no supervisado, con objeto de que la red pueda adaptarse de forma gradual a los cambios que puedan producirse en su entorno.

La arquitectura de las redes con aprendizaje no supervisado suele ser bastante simple. La complejidad de las mismas radica en sus leyes de aprendizaje. Muchas de ellas tienen una sola capa; la mayoría son *feed-forward* (dirigidas hacia delante), con la notable excepción del modelo de Grossberg de la *Teoría de la Resonancia Adaptativa*, que se describirá al final del capítulo. Suele haber, también, bastantes menos unidades que entradas, salvo en el caso de los *mapas autoorganizativos de Kohonen* en los que normalmente ocurre lo contrario. Otra característica común a todas ellas es que suelen estar más cerca de las estructuras neurobiológicas que de las puramente computacionales; se basan en estudios cerebrales y tienden a imitar sus características y comportamiento.

Tal vez los modelos más característicos de redes no supervisadas sean los mencionados con anterioridad de *Kohonen* y *Grossberg*; son, de hecho, los más mencionados en la literatura, y los más utilizados en casos prácticos. Por ello se han elegido como muestra de lo que es y de lo que puede hacerse con los métodos de aprendizaje no supervisados.

### 6.1.1. Categorización en redes neuronales biológicas

Cualquier modelo neuronal que quiera acercar su comportamiento al de los sistemas biológicos debe contar con una serie de características esenciales, que son las que proporcionan a los sistemas biológicos gran parte de sus capacidades.

Una de las principales características de las Redes Neuronales es su dimensión. En el cerebro existen entre  $10^{11}$  y  $10^{14}$  neuronas, y cada una de ellas posee entre 1.000 y 10.000 conexiones con otras tantas células [Simpson, 1990]. Esto deja a los sistemas artificiales varios órdenes de magnitud por debajo, en cuanto a conectividad.

El funcionamiento general de los sistemas nerviosos es altamente paralelo. Es este grado de paralelismo el que permite que, con una velocidad de señal relativamente baja (1 msec), el cerebro pueda reaccionar con rapidez ante la continua necesidad de tomar decisiones, en las que existen un gran número de variables. Dichas decisiones resultan extraordinariamente complejas de plantear, y en muchos casos irresolvibles en tiempo real, incluso para los mejores computadores actuales, a pesar de que éstos poseen velocidades de alrededor de seis órdenes de magnitud superiores a las Redes Neuronales.

En los sistemas nerviosos se realizan procesos de convergencia y divergencia de la información. Mediante la existencia de múltiples conexiones de entrada y de salida para cada neurona, se produce una multiplicación del uso de una determinada información (divergencia), así como una combinación de diversas señales de entrada para producir un efecto conjunto en una sola neurona (convergencia).

Las redes neuronales<sup>1</sup> poseen una plasticidad intrínseca, mediante la cual están dotadas de la capacidad de aprender a partir de la experiencia y adaptarse a entornos nuevos con extrema facilidad.

Poseen una cierta predeterminación genética, dada por la existencia de un mapa de conexiones prefijado. Esto significa, básicamente, que el diseño fundamental de la arquitectura de las distintas Redes Neuronales está predeterminado genéticamente, y ha sido seleccionado de entre los modelos posibles por su valor adaptativo para cada especie. Así pues, la arquitectura general de las conexiones está diseñada de antemano, si bien la potencia relativa de dichas conexiones se obtendrá a lo largo del aprendizaje. Esto no entra en absoluto en contradicción con la idea anterior de plasticidad, como en un principio podría parecer.

Por su relación con los temas que serán tratados a continuación, se van a describir una serie de características importantes de las Redes Neuronales, cuando tratan con problemas de categorización.

**Aprendizaje continuo.** El aprendizaje está intrínsecamente ligado al funcionamiento del sistema. En la mayoría de los modelos simulados de aprendizaje neuronal, sin embargo, existe una separación en fases entre el funcionamiento estable del sistema y su aprendizaje. Primero se produce el aprendizaje a partir de un conjunto de ejemplos de prueba, y una vez que la red ha aprendido, el sistema se pone en funcionamiento sin volver a cambiar los valores de ninguno de sus parámetros. Los seres vivos, por el contrario, en los procesos de clasificación están constantemente adaptando sus parámetros de discriminación, y es probable que la clasificación varíe a lo largo del tiempo, en función de los estímulos que hayan recibido. Además, al contrario de los modelos artificiales, producen un resultado de clasificación desde el principio de la existencia del sistema, incluso cuando el aprendizaje ha sido aún prácticamente inexistente. La clasificación que producen en estas primeras fases puede no ser muy correcta, pero les permite reaccionar a las entradas y adaptarse en función de los nuevos estímulos.

**Determinación de entradas relevantes** La categorización es mejor cuantas más veces se haya recibido el mismo estímulo o uno parecido. Para las cosas cotidianas no tenemos problemas en asignarles una categoría: una silla enseguida podemos decir que es una silla; un aparato de cirugía tenemos dificultades en categorizarlo a no ser que seamos cirujanos. Por el contrario, cuanto mayor tiempo haga que no se repite un estímulo, será más fácil que lo olvidemos. Se pueden recordar cosas que han sucedido una sola vez, si el evento ha ocurrido en un tiempo relativamente reciente; sin embargo, a medida que transcurre tiempo sin que el estímulo se repita, su categorización se hace más complicada. Los estímulos son olvidados cuando ya no son necesarios, y la medida de necesidad de un estímulo, en términos de necesidad de categorizarlo correctamente,

<sup>1</sup>Se utilizará el término redes neuronales para hacer referencia a los sistemas biológicos, mientras que se utilizará el término Redes de Neuronas Artificiales para los sistemas computacionales.

depende del tiempo transcurrido desde su última aparición y del número de veces que ha aparecido. Éste es el mismo mecanismo que utilizan las memorias de computador. La información más reciente es almacenada en las memorias más rápidas, pequeñas y costosas de una máquina, pero como su capacidad es reducida, transcurrido un tiempo sin ser utilizada, la información es desplazada a memorias más lentas y grandes.

Esta última característica es de gran importancia para el buen funcionamiento de los sistemas nerviosos debido a las limitaciones de espacio, al igual que en las memorias de los computadores referidas anteriormente. La capacidad de almacenamiento de un sistema nervioso es muy grande, pero limitada; si tuvieran que "recordar" todos los estímulos recibidos se colapsaría en muy poco tiempo, además de estar "almacenando" información muy poco útil. Es por esto por lo que es necesario almacenar sólo aquella información que realmente pueda ser de utilidad. Mediante el mecanismo de almacenamiento y olvido descrito, los sistemas nerviosos pueden estar constantemente incorporando nueva información, imprescindible para poder sobrevivir, sin que su capacidad neuronal limitada se colapse.

**Plasticidad** Los seres vivos se adaptan con extrema facilidad a nuevas situaciones, aunque éstas sean completamente diferentes y cambien drásticamente, al mismo tiempo que son capaces de pasar por alto estímulos irrelevantes. A veces hay ciertas situaciones que es conveniente almacenar, debido a su importancia para el ser vivo que las padece, aunque no se repitan muy a menudo, o aunque no hayan ocurrido recientemente: situaciones de peligro, situaciones de supervivencia. Un elefante es capaz de conducir a la manada a una laguna que nunca se seca, en períodos de gran sequía, aunque haya estado en ella una sola vez, muchos años atrás, y tenga que recorrer grandes distancias.

Todas las características anteriores son de vital importancia para los animales, ya que les permiten la supervivencia en el mundo complejo y en constante cambio en que se desenvuelven y les dota de una capacidad de decisión crucial para optar por la mejor respuesta ante cada situación. Un sistema neuronal artificial que intentara asemejarse a la forma de categorización de los animales debería incluir al menos las características citadas anteriormente.

### 6.1.2. Regla de Hebb

Como ya se ha señalado a propósito de las Redes de Neuronas Artificiales con aprendizaje no supervisado, no existe ninguna información externa que indique si se están produciendo resultados erróneos, ni que ayude a decidir cómo y en qué grado modificar las conexiones. Entonces, ¿con qué criterio se van a modificar las conexiones?, ¿qué es lo que guiará el aprendizaje de la red?

Donald Hebb realizó una primera aproximación a la resolución de estas preguntas. En 1949 [Hebb, 1949], enunció una regla que lleva su nombre y que ha pasado a ser una aportación fundamental en las ANN. La regla se refería a cierto comportamiento biológico que Hebb observó en las células cerebrales. Su enunciado es el siguiente:

Cuando un axón de una célula A está lo suficientemente cerca para excitar a una célula B, y toma parte repetidamente en el proceso de disparo de dicha célula, se produce algún tipo de cambio metabólico en una de las células (o en las dos), que hace que la eficacia con la que A disparaba a B se vea incrementada.

Como se ve, esta regla de modificación sináptica no depende de ningún factor externo; simplemente hace que las células vayan incluyéndose unas a otras, es decir que se auto-configure la permeabilidad sináptica de la red, a partir de las reacciones a los estímulos recibidos. Esta regla ha demostrado ser de gran eficacia en las Redes de Neuronas Artificiales, y es casi universalmente utilizada en los sistemas no supervisados.

Esta regla, tal y como fue enunciada por Hebb, deja abierto un gran abanico de posibilidades, de manera que cuando una regla de aprendizaje en una red tiene forma análoga a lo expresado por Hebb, se dice que posee un aprendizaje hebbiano. Es decir, no existe una única implementación de la regla de Hebb, sino un cúmulo de reglas hebbianas.

A continuación se detalla una serie de interpretaciones matemáticas, todas ellas utilizadas en diferentes modelos, de la regla de Hebb:

$$\begin{aligned}\Delta W_{ij} &= a_i \cdot a_j & \Delta W_{ij} &= \mu(a_i - \bar{a}_i) \cdot (a_j - \bar{a}_j) \\ \Delta W_{ij} &= \Delta a_i \cdot \Delta a_j & \Delta W_{ij} &= \mu a_i W_{ij} \Delta a_j \\ \frac{dW_{ij}}{dt} &= -W_{ij} + a_i \cdot a_j & \frac{dW_{ij}}{dt} &= -W_{ij} + \sigma(a_i) \cdot \sigma(a_j) \\ \frac{dW_{ij}}{dt} &= -W_{ij} + \frac{d(\sigma(a_i))}{dt} \cdot \frac{d(\sigma(a_j))}{dt}\end{aligned}$$

donde:

- $W_{ij}$  = Peso de la conexión de las células i y j.
- $a_i$  = Valor de activación de la célula i.
- $\bar{a}_i$  = Media de los valores de activación.
- $\sigma$  = Función de umbral, tipo sigmoidal.

### 6.1.3. Modelo de interacción lateral

#### Características básicas

Una de las propiedades del cerebro más pasadas por alto en los modelos artificiales existentes es aquella del orden significativo de sus unidades de proceso. Este orden hace que unidades estructuralmente idénticas tengan una funcionalidad diferente, debida a parámetros internos que evolucionan de forma variable según sea la ordenación de las células.

Esta propiedad “topológica” del cerebro parece ser de fundamental importancia para la representación de cierto tipo de información, en las imágenes visuales, en las abstracciones, etc. Estos mapas topológicos se encuentran presentes en la corteza cerebral y se encargan de diversas tareas de tipo sensorial y motor.

Cuando pensamos, y también en el procesado de información del subconsciente, realizamos una compresión de la información formando representaciones reducidas con los aspectos más relevantes de la misma, sin que se produzca por ello ninguna pérdida de conocimiento acerca de las interrelaciones que se producen. El propósito de este procesado inteligente de la información parece ser la creación de imágenes del mundo observable, a varios niveles de abstracción.

Así pues, uno de los rasgos esenciales de las Redes Neuronales es el de disponer de un mecanismo capaz de extraer de manera automática las características más relevantes de un conjunto de datos, en principio arbitrario, y de ámbito heterogéneo. Corresponde a un sistema de los denominados de “clustering” capaz de adaptarse de forma automática a los estímulos recibidos y producir una clasificación en función de las características significativas, sin ningún mecanismo externo que determine el resultado.

Esto es en parte posible gracias a la ya mencionada propiedad “topológica” de las Redes Neuronales. Según Kohonen [Kohonen, 1982] ciertas Redes Neuronales pueden adaptar sus respuestas de tal forma que la posición de la célula que produce la respuesta pasa a ser específica de una determinada característica de la señal de entrada. Esta especificidad se da en el mismo orden topológico para la red que el que existe entre las características de las señales de entrada.

Esto quiere decir que la estructura topológica de la red absorbe a su vez aquella que se produce entre las características de los datos, y por tanto el sistema no sólo es capaz de realizar una clasificación de los estímulos, sino que además pondrá de relieve y conservará las relaciones existentes entre las diferentes clases obtenidas.

#### Descripción del modelo

Existen varios modelos de Redes de Neuronas Artificiales que tratan de incorporar las propiedades anteriores. Estudios realizados sobre el neocórtez cerebral [Hubel and Wiesel, 1962] demuestran que éste está formado esencialmente por capas de células bidimensionales interconectadas en cada capa por conexiones laterales.

Cada neurona está conectada con otras de su entorno de manera que produce una excitación en las más próximas y una inhibición en las más alejadas (Figura 6.1). Tanto la excitación como la inhibición laterales son gradualmente más débiles a medida que nos alejamos de la neurona en cuestión.

Este mecanismo hace que cuando un estímulo produce una reacción en una célula, las células de su inmediato entorno se vean influenciadas por dicha reacción, de una manera positiva las más cercanas, y negativa las más alejadas. Igualmente, a medida que la señal se aleja de la célula que la produjo, esa influencia va progresivamente debilitándose. De esta manera el orden de las neuronas influye directamente en la forma en que las señales van a ser propagadas a través de la red, y en la respuesta de la misma.

La función matemática que establece esta “interacción lateral” viene a menudo definida por una función (Figura 6.2) en forma de sombrero mexicano.

En el modelo de interacción lateral, la salida de cada célula se calcula de

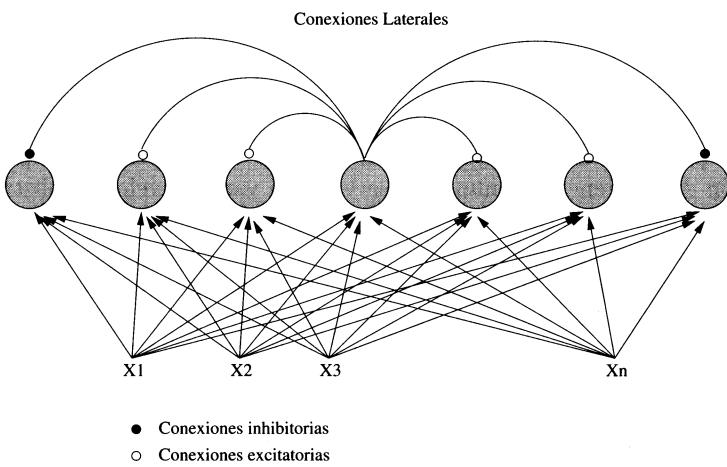


Figura 6.1: Neuronas con conexión lateral

forma similar a los modelos “feed-forward” presentados en capítulos anteriores; esto es, la salida de cada célula se obtiene aplicando una función no lineal a la suma, ponderada por los pesos de las conexiones, de las entradas a la célula, incluyendo el umbral. Como en este caso las conexiones tienen valores fijos, determinados por la función de la Figura 6.2, la salida de cada célula podría expresarse por la Ecuación 6.1.

Su formulación en el campo discreto sería:

$$\mu_i(t) = \sigma[\theta_i(t) + \sum_{k=-l}^l \gamma_k \mu_{i+k}(t-1)] \quad (6.1)$$

donde:

- $\mu_i(t)$  es la salida de la unidad  $i$ .
- $\sigma[x]$  es una función no lineal de tipo sigmoide.
- $\gamma_k$  es un coeficiente extraído de muestrear en el tiempo la señal de la Figura 6.2, de forma que para  $-l < k < l$  tenga un valor positivo, y negativo en el resto.
- $\theta_i(t)$  es el umbral de la célula  $i$ .

La interpretación de la función es la siguiente. Si una célula está próxima a la célula objeto de análisis recibe una influencia muy grande de ella, por eso  $f(x)$  es grande para valores absolutos de  $x$  pequeños. A medida que la distancia entre las células aumenta ( $|x|$  aumenta), la influencia disminuye hasta hacerse incluso negativa. Sigue aumentando su influencia negativa, hasta una determinada distancia en la que la influencia se diluye y desaparece.

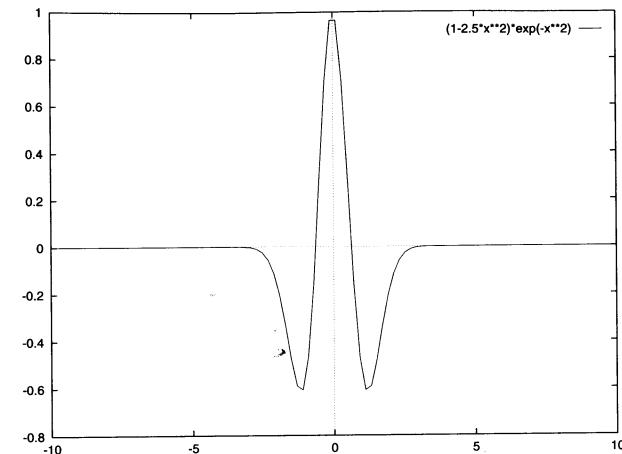


Figura 6.2: Función de sombrero mexicano para la interacción lateral

De esta forma se producen determinados grupos de activación a lo largo de la red, influidos por neuronas con altos grados de activación, que propagan su nivel de actividad. En resumen, en este modelo, el nivel de activación de una célula depende no sólo de la entrada que recibe, sino también, y gracias a las conexiones laterales, de la ubicación y la actividad general de su vecindario.

El presente modelo de interacción lateral no constituye por sí mismo un modelo de Red de Neuronas Artificiales, ya que no existe aprendizaje. Permite conseguir las propiedades de localidad, descritas anteriormente, pero siempre en unión con algún otro modelo de Red de Neuronas Artificiales con aprendizaje. El aprendizaje no se produce en la capa de interacción lateral, sino en alguna otra capa incluida para completar el modelo. Así pues, la interacción lateral es un mecanismo introducido en algún otro modelo para producir localidad y ayudar a la creación de núcleos neuronales vecinos especializados. A continuación se describirá un modelo de aprendizaje que utiliza un esquema de interacción lateral, el aprendizaje competitivo. Pero antes se describirán algunas características de los sistemas biológicos en tareas de clasificación.

#### 6.1.4. Aprendizaje competitivo

En esta sección se van a describir unos modelos de Redes de Neuronas Artificiales que se crearon con la finalidad de incorporar ciertas características, que se mencionaron con anterioridad, de las Redes Neuronales. Grossberg [S., 1980] recoge estas características, que más tarde incluirá en sus modelos ART [Grossberg, 1987, Carpenter and Grossberg, 1987, Carpenter et al., 1991a] y [Carpenter et al., 1991b].

El aprendizaje competitivo es un tipo de aprendizaje no supervisado que sirve de base para varios modelos de Redes de Neuronas Artificiales. El objetivo

de este tipo de redes es *categorizar* los datos de entrada. Se trata de que los datos parecidos sean clasificados como pertenecientes a la misma categoría. En estos modelos suele haber una capa de clasificación compuesta por tantas neuronas como categorías pueda haber en los datos. Cada categoría está representada por un prototipo cuyas características son una especie de compendio de las características de los datos pertenecientes a esa misma categoría. En la capa de clasificación, cada célula corresponde, pues, a un prototipo. El sistema debe relacionar cada célula (prototipo) con los datos de entrada que representa. En otros términos, debe agrupar los datos de entrada en categorías, por razones de similitud, y asignar a cada categoría un prototipo, que más tarde será utilizado para clasificar datos nuevos y desconocidos.

La arquitectura de una red de neuronas artificial con aprendizaje competitivo es la siguiente. Existen dos capas denominadas  $F_1$  y  $F_2$ . La capa  $F_1$  es la llamada capa de entrada y recibe los datos de entrada (señales del entorno). La capa  $F_2$  es la capa de competición y se encarga de producir la salida.

Cada célula de la capa  $F_1$  está conectada con todas las células de la capa  $F_2$  a través de conexiones ponderadas variables. Por su parte la capa  $F_2$ , además de recibir las entradas de la capa  $F_1$ , tiene conexiones laterales inhibitorias entre todas las células de su capa, excepto consigo misma, en que la conexión es excitatoria (Figura 6.3).

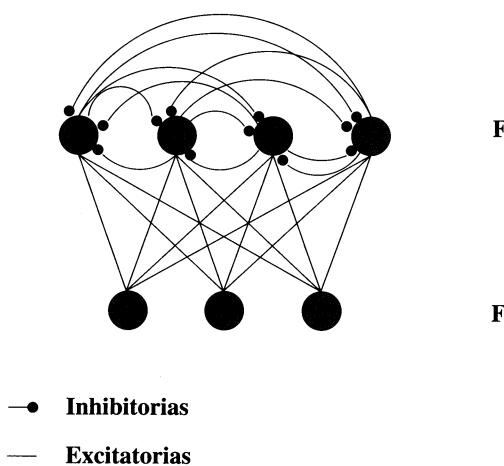


Figura 6.3: Arquitectura del Competitive Learning

Las conexiones existentes en la capa  $F_2$  son fijas y el valor es asignado en el momento de creación de la red. Esta arquitectura es exactamente un sistema de interacción lateral en el que la función de interacción de cada célula sólo toma valores positivos para un radio de cero, es decir, para la propia célula, y valores negativos constantes para el resto de las células.

En este modelo, cada célula se refuerza a sí misma, es decir, realimenta su valor de activación para compensar las inhibiciones que provienen del resto de

las células de su capa.

Cuando se recibe una entrada en la capa  $F_1$ , ésta se propaga hasta la capa  $F_2$ , cuyas células tendrán un valor de activación proporcional al valor de la entrada propagada. Esta proporción vendrá dada por los valores de las conexiones entre  $F_1$  y  $F_2$  ( $W_{ij}$ ). Una vez que las células de  $F_2$  han sido activadas por la capa  $F_1$ , ésta deja de actuar, y la señal es propagada, de forma asíncrona, a través de la capa  $F_2$ , de manera que todas las células tratarán de impedir que las demás tengan un valor de activación alto, gracias a las conexiones inhibitorias con todas sus vecinas; a la vez que intentarán tener ellas mismas un valor de activación alto, gracias a las conexiones reflexivas excitatorias.

El algoritmo que describe el funcionamiento de la red es el siguiente:

1. Se recibe el estímulo en  $F_1$ .
2. Se propaga la señal hasta  $F_2$  y se calcula el valor de excitación para cada célula de  $F_2$ .
3. Se inhiben las conexiones entre la capa  $F_1$  y la  $F_2$ .  
Se propaga la señal por la capa  $F_2$ , calculándose los nuevos valores de excitación de las células.  
Cuando sólo haya una célula (célula ganadora) con un valor de salida mayor que cero, ir al paso 5.
4. Ir al paso 3.
5. Restablecer las conexiones entre las capas  $F_1$  y  $F_2$ .  
Calcular los nuevos valores para los pesos de las conexiones entre la capa  $F_1$  y la célula ganadora en el paso 3.

La capa  $F_2$  se dice que se ha estabilizado cuando todas las salidas de las células tienen un valor de cero, excepto una, que será precisamente la que al principio ha recibido la entrada más alta de la capa  $F_1$ , ya que será la que habrá inhibido en mayor grado al resto y también se habrá reforzado a sí misma en mayor grado.

$$\forall i, i \neq j \mu(F_{2i}) = 0 \mu(F_{2j}) > 0$$

Se dice, pues, que las células de la capa  $F_2$  *compiten* por la entrada, de ahí el nombre del método. Sólo una consigue ganar la competición; a ésa se la etiqueta con el nombre de célula ganadora.

En la capa  $F_2$  no hay aprendizaje, los pesos son fijos y se asignan según la siguiente ecuación:

$$\forall i, \sum_j^N W_{ij} = 1$$

$$\forall i, j W_{ij} = W_{ji}$$

La célula ganadora representa al prototipo que se asigna al dato de entrada. Para hacer que la siguiente vez el mismo dato de entrada haga activarse aún más a su prototipo relacionado, para cada dato de entrada se realiza un ciclo de aprendizaje. Las conexiones entre la capa  $F_1$ , la de la entrada, y la célula ganadora son reforzadas. Este aprendizaje sólo modifica las conexiones de la célula ganadora, y por eso recibe el nombre de “el que gana se lo lleva todo” (Figura 6.4) (en inglés *winner takes all*). Esto hará que en el futuro cada célula tenga aún mayor facilidad para reconocer el estímulo que aprendió, e incluso estímulos parecidos.

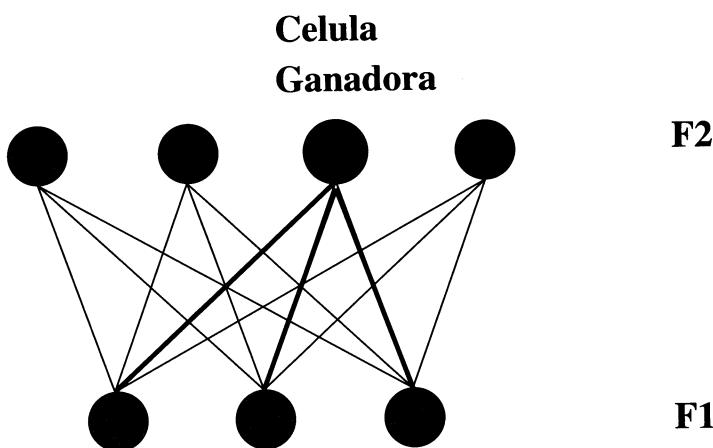


Figura 6.4: Método de aprendizaje *winner takes all*

Las conexiones de la capa  $F_2$  permanecen inmutables a lo largo de todo el proceso, y las conexiones entre la capa  $F_1$  y la célula ganadora se modifican siguiendo la ecuación:

$$\frac{dW_{ij}}{dt} = \sigma(a_i)[-W_{ij} + \sigma(b_j)]$$

Grossberg [Grossberg, 1987] ofrece una nueva versión del método en la que las ecuaciones de aprendizaje son las siguientes:

$$\frac{dW_{ij}}{dt} = (-W_{ij} + \theta_i) \cdot g(b_j) \quad (6.2)$$

donde:

$$g(b_j) = \begin{cases} 1 & \text{si } j \text{ ganadora} \\ 0 & \text{en caso contrario} \end{cases}$$

$$\theta_i = \frac{I_i}{\sum_{j=1}^M I_j}$$

siendo  $I_i$  el  $i$ -ésimo elemento del ejemplo de entrada.

En este modelo de aprendizaje competitivo de Grossberg se aprecia cómo la función  $g(x)$  se introduce para realizar aprendizaje sólo de la célula ganadora (*winner takes all*). Además, el incremento del valor de la conexión es proporcional al valor de la entrada normalizado.  $\theta_i$  se calcula como la parte proporcional de la entrada que está en esa posición  $i$ . Por ejemplo, si el vector de entrada tiene tres componentes en los que el segundo es tres veces el primero y el tercero seis veces el primero, el valor de los componentes del vector  $\theta$  seguirá ese mismo esquema, sumando todos ellos 1. Es decir:  $I = \{1, 2, 6\}$ , implicará un  $\theta = \{0, 1, 0, 3, 0, 6\}$ , y el incremento de las conexiones será también esta cantidad. Además se aprecia que el aprendizaje sigue un esquema *hebbiano*, ya que, al ser la salida de la célula ganadora 1 y la del resto 0, el aprendizaje de cada conexión es proporcional a los valores de salida de las células que une.

A pesar de todo esto el modelo de aprendizaje competitivo tiene carencias que lo alejan mucho del comportamiento de los sistemas nerviosos en tareas de categorización, descrito anteriormente. Algunas de estas limitaciones son las siguientes:

- No es capaz de producir una codificación estable ante entradas arbitrarias. Es decir, si se perturba la entrada con valores aleatorios y a continuación se introduce uno de los datos de entrada presentados con anterioridad, y que había sido convenientemente estabilizado por la red, puede producirse una respuesta incorrecta. El ruido introducido en el sistema puede perturbar lo ya aprendido pudiendo provocar respuestas incorrectas.
- Tiene una capacidad limitada de codificación, en función de las dimensiones de la red, el número de entradas que puede recibir y categorizar correctamente es restringido.
- Es necesario establecer a priori el número de categorías de la clasificación, lo cual limita el numero de entradas a almacenar, o puede producir que haya categorías no utilizadas.

Estas limitaciones y el deseo de diseñar modelos que tuvieran un comportamiento biológico más aceptable llevaron a Grossberg [Grossberg, 1987] a formular su modelo de Teoría de la Resonancia Adaptativa ART (Adaptive Resonance Theory).

A continuación se describirán los dos modelos de Redes de Neuronas Artificiales que recogen las características descritas, para tareas de clasificación: los Mapas Autoorganizativos de Kohonen, y la Teoría de la Resonancia Adaptativa, respectivamente.

## 6.2. Mapas autoorganizativos de Kohonen

### 6.2.1. Descripción del modelo

A partir de los estudios descritos anteriormente, un científico finlandés, Teuvo Kohonen, diseñó un modelo adaptable a las características expuestas de los sistemas neuronales [Kohonen, 1982]. Este modelo, llamado mapa de características de Kohonen, en inglés, *Kohonen's Feature Map*, consiste en una Red Neuronal de dos capas, una primera capa de entrada y una segunda de competición (Figura 6.5).

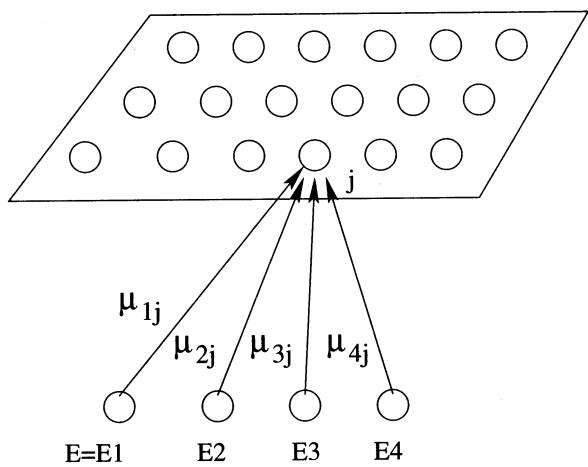


Figura 6.5: Arquitectura del mapa de Kohonen

La capa de entrada recibe la señal de entrada a la red. Como siempre la dimensión de la entrada depende de los atributos que tengan los datos de entrada, sea  $n$  dicho número atributos, cada entrada viene definida por un vector  $\epsilon = \{\epsilon_1, \dots, \epsilon_n\}$ . Cada neurona de la entrada está conectada a todas las células de la capa de competición; si hay  $m$  células en la capa de competición, entonces los pesos de las conexiones entre las dos capas se pueden definir mediante la siguiente matriz:

$$\begin{pmatrix} \mu_{11}, & \mu_{12}, & \dots, & \mu_{1m} \\ \mu_{12}, & \mu_{22}, & \dots, & \mu_{2m} \\ \dots, & \dots, & \dots, & \dots \\ \mu_{1n}, & \mu_{n2}, & \dots, & \mu_{nm} \end{pmatrix}$$

donde  $\mu_{ij}$  es el peso de la conexión entre la célula  $i$ -ésima de la capa de entrada y la  $j$ -ésima de la capa de competición. Estas conexiones son inicializadas aleatoriamente al principio de la fase de aprendizaje de la red, y son modificadas a lo largo de dicha fase, de la forma que se describirá más tarde.

Si se selecciona una de las columnas de la matriz anterior, la  $j$ -ésima por ejemplo, se estará haciendo referencia a la célula  $j$  de la capa de competición, y el vector que se obtiene sería:  $\mu_j = \{\mu_{1j}, \mu_{2j}, \dots, \mu_{ij}, \dots, \mu_{nj}\}$ , que tienen el mismo número de componentes (la misma dimensión) que el vector de entrada de  $\epsilon$ . Al tener la misma dimensión, se pueden comparar entre sí, o lo que es lo mismo, definir una función de distancia entre ellos:

$$d : ij = d(\epsilon, \mu_j)$$

El modelo de Kohonen aplica esta función distancia para calcular la salida de las células de la capa de competición. Sea esta salida  $\tau_j$ ; entonces el cálculo sería:

$$\tau_j = d(\epsilon, \mu_j)$$

Dependiendo de la función distancia elegida, el modelo se comportará de una u otra manera. Así pues, para una función distancia que sea el producto escalar:

$$d(x, y) = \sum_{i=1}^n x_i y_i$$

la salida de la capa de competición será la habitual en los modelos supervisados:

$$\tau_j = \sum_{i=1}^n \epsilon_i \mu_{ij}$$

es decir, la suma ponderada de las entradas por los pesos de las conexiones. Ahora bien, al estar definidos la mayoría de los problemas reales en estos términos, la función más utilizada es la distancia euclídea:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

En este caso el modelo quedaría como sigue:

$$\tau_j = \sqrt{\sum_{i=1}^n (\epsilon_i - \mu_{ij})^2}$$

En este apartado se tomará como referencia esta última distancia, pero no hay que olvidar que el modelo acepta cualquier otra definición de distancia, si el problema lo requiere.

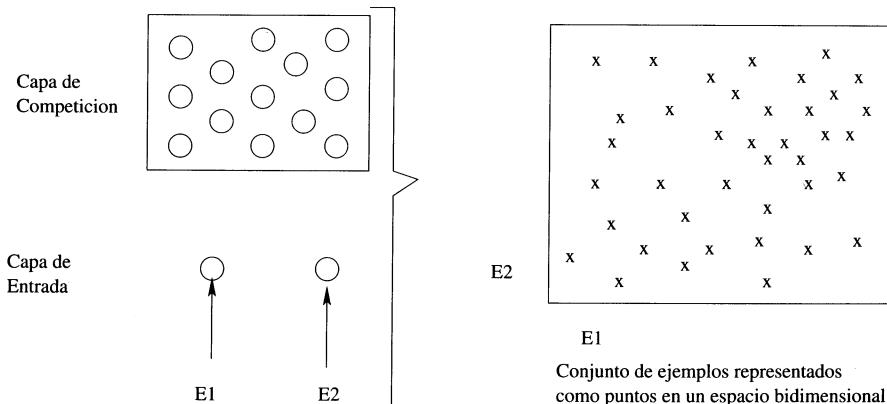
El funcionamiento del modelo es el siguiente. Se recibe el vector entrada y se propaga por las conexiones hasta llegar a la capa de competición. Dicha

propagación se realiza de manera que cada célula de la capa de competición produce una salida, que es el resultado de comparar su entrada (común para todas ellas) con los valores de los pesos de las conexiones de dichas células mediante una función de distancia definida como parámetro del sistema (normalmente la distancia euclídea).

Todas las salidas de las células de la capa de competición son comparadas entre sí, para seleccionar aquella que produzca una salida más pequeña. A dicha célula se la etiqueta con el nombre de célula ganadora para la entrada correspondiente.

Esta competición se realiza mediante un modelo de interacción lateral, como se ha descrito en la sección anterior. Esto quiere decir que las células de la capa de competición tienen conexiones a sus vecinas de su misma capa, siguiendo una asignación mediante la función "sombbrero mexicano" de la Figura 6.1. Estas conexiones son fijas y permiten retroalimentar las salidas más grandes, de forma que la red genere un núcleo de actividad alrededor de la célula que produce una salida mayor. Sin embargo, en este modelo se desea que dicha actividad sea alrededor de la célula que produce la salida menor. Simplemente cambiando el signo de la función distancia, que siempre es positiva, se obtendría el comportamiento deseado. Esto es; una alta actividad alrededor de la neurona cuya salida es menor, es decir, cuya distancia respecto de la entrada es menor.

Esto podría describirse mejor desde un punto de vista geométrico de la siguiente manera.



Cada ejemplo de entrenamiento viene definido por un vector de  $n$  componentes. Esto podría representarse mediante un punto en un espacio de  $n$  dimensiones. Si  $n$  es igual a 2, cada ejemplo sería un punto en un plano (Figura 6.6).

Ahora bien, cada célula de la capa de competición tiene conexiones a todas las células de la capa de entrada. Si la dimensión de la entrada es  $n$ , habrá,

para cada célula de la capa de competición,  $n$  conexiones con valores de pesos. Así pues, las células de la capa de competición también pueden representarse como puntos en un espacio  $n$ -dimensional (Figura 6.7).

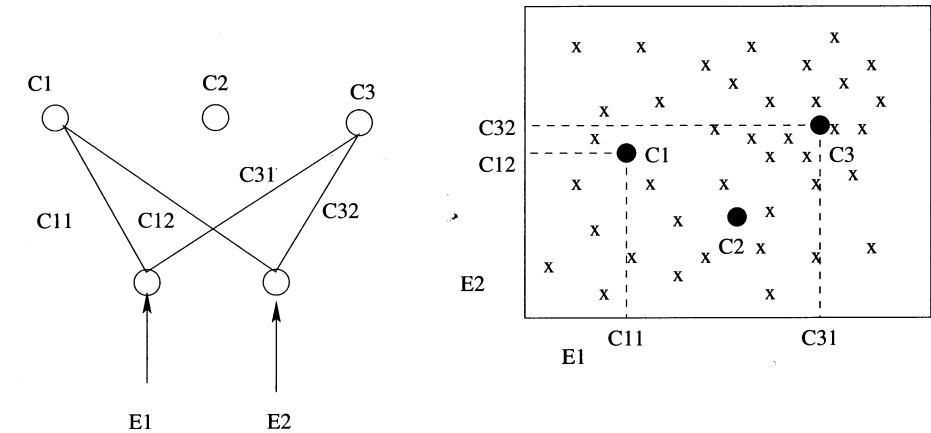


Figura 6.7: Representación geométrica bidimensional de una red de Kohonen

Una vez realizado el proceso descrito, se requiere que la red responda de manera similar a estímulos parecidos; para ello se aplica una regla de aprendizaje tipo Hebb, es decir, se refuerzan más aquellas unidades que hayan respondido en mayor grado a la entrada, de forma proporcional al valor de la entrada. Esto podría formalizarse como sigue:

$$\frac{d\mu_{ij}}{dt} = \alpha(t)\tau_j(t)(\epsilon_i(t) - \mu_{ij}(t)) \quad (6.3)$$

Analizando la ecuación se aprecia cómo se trata, efectivamente, de un aprendizaje hebbiano, ya que, para cada conexión, el incremento del valor de su peso es proporcional al valor de activación de las células que conecta ( $\tau_j \cdot \epsilon_i$ ).

Por otra parte, el método de aprendizaje utilizado sigue el esquema "el que gana se lo lleva todo" comentado anteriormente. Para ello, lo único que se necesita es que la salida de la célula ganadora sea uno y las del resto sean cero, esto es:

$$\tau_i = \begin{cases} 1 & \text{si } C_i \text{ ganadora} \\ 0 & \text{en caso contrario} \end{cases}$$

Esto se consigue siguiendo el esquema de interacción lateral comentado anteriormente. En este caso la Ecuación 6.3 se convierte en:

$$\frac{d\mu_{ij}}{dt} = \begin{cases} \alpha(t)(\epsilon_i(t) - \mu_{ij}(t)) & \text{si } C_i \text{ ganadora} \\ 0 & \text{en caso contrario} \end{cases} \quad (6.4)$$

Éste es el esquema final de aprendizaje del método de Kohonen, siendo  $\alpha$  la tasa de aprendizaje. En el método de Kohonen la tasa de aprendizaje es

decreciente con el tiempo. Es una función denominada “de olvido” que hace que los patrones ya introducidos con anterioridad sean aprendidos más intensamente por la red neuronal. El sistema utilizado para decrementar la variable  $\alpha$  forma parte del esquema de aprendizaje del método. Normalmente se utilizan dos esquemas equivalentes:

- El valor de  $\alpha$  se decrementa una cantidad constante pequeña  $\beta$ , tras cada ciclo completo de todos los patrones de aprendizaje:

$$\alpha(t+1) = \alpha(t) - \beta$$

Mediante la asignación del parámetro  $\beta$  se pueden determinar el número total de ciclos aprendizaje, que sería:

$$\text{Iteraciones} = \frac{\alpha(0)}{\beta}$$

ya que después de dicho número de ciclos el valor de  $\alpha$  sería cero y la red estaría estabilizada.

- El valor de  $\alpha$  se decrementa siguiendo un esquema logarítmico. En este caso, el decremento es muy elevado en las primeras iteraciones y se va reduciendo paulatinamente hasta que alcanza valores muy pequeños, con una asíntota en el cero. Este esquema hace que se les dé más importancia a las primeras iteraciones, que son las que tienden a formar la estructura de la red; después se ajusta con ligeros desplazamientos de las células hasta alcanzar el equilibrio.

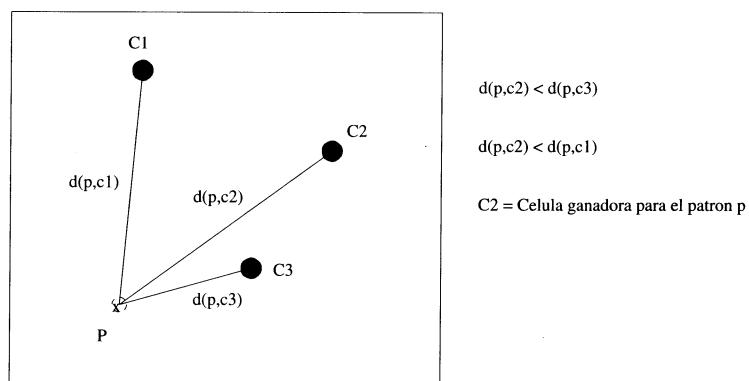


Figura 6.8: Cálculo geométrico de la célula ganadora

En este caso, el funcionamiento de la red al introducirse un patrón podría describirse como el cálculo de la distancia entre dicho patrón y cada una de las

células, que harían las veces de prototipos de los patrones de entrenamiento. Así pues, la célula ganadora sería aquella cuya distancia con respecto al patrón introducido fuese más pequeña (Figura 6.8).

Así pues, el procedimiento general sería:

- Seleccionar un patrón de los del conjunto de entrenamiento.
- Calcular la distancia de dicho patrón a cada una de las células del mapa de Kohonen.
- Etiquetar como ganadora a aquella célula cuya distancia sea menor.

Siguiendo con la analogía, el entrenamiento se puede describir de la manera siguiente: dada la fórmula 6.4

$$\Delta\mu_{ij} = \alpha(\epsilon_i - \mu_{ij})$$

donde  $\forall i = 1, \dots, n$  y donde  $C_j$  es la célula ganadora (Figura 6.9).

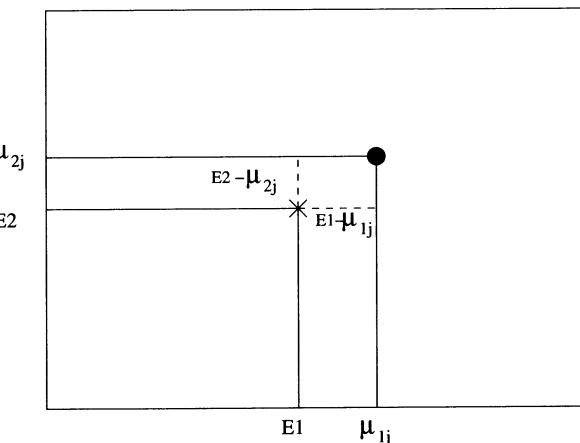


Figura 6.9: Cálculo geométrico del valores del incremento/decremento de los pesos

Se aprecia que el incremento de la conexión es proporcional ( $\alpha$ ) a la distancia entre el patrón y la célula ganadora, conservándose el signo (Figura 6.9). Además, desde un punto de vista geométrico, la modificación de los valores de las conexiones se puede interpretar como un desplazamiento del prototipo al que representa la célula ganadora; de forma que, después de moverse, se quede a una distancia menor del ejemplo introducido. El desplazamiento será tanto mayor cuanto más grande sea el valor de  $\alpha$  (Figura 6.10).

En el esquema de aprendizaje propuesto por el método, se comienza con un  $\alpha$  grande que hace que los prototipos sean atraídos con fuerza por los patrones, pero el valor de  $\alpha$  desciende con el tiempo de forma que, en sucesivas

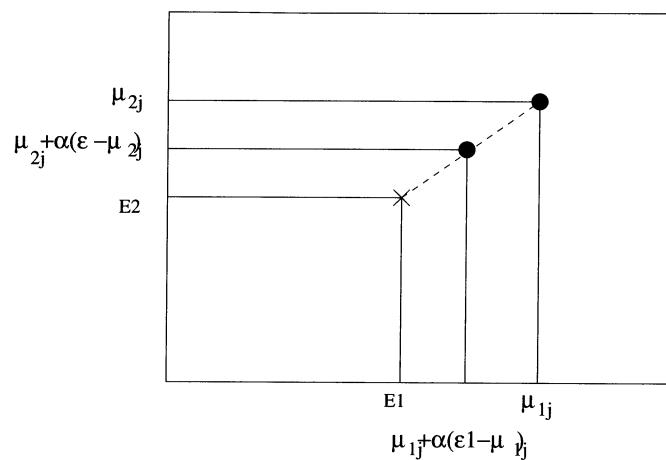


Figura 6.10: Desplazamiento del prototipo a raíz de la regla de aprendizaje

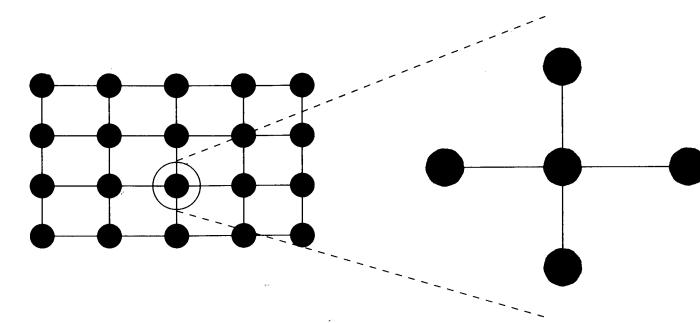
presentaciones de los patrones, dicha atracción vaya disminuyendo, hasta desaparecer cuando  $\alpha$  toma valor cero. El efecto que produce este esquema es el desplazamiento de los prototipos hacia el centro de gravedad de “nubes” más o menos compactas de patrones de entrenamiento. Éste es precisamente el efecto deseable en problemas de agrupamientos.

Además, el sistema incorpora la capacidad de relacionar los prototipos generados entre sí, por razones de similitud entre los datos a los que representa cada uno de ellos. Esto se realiza mediante el mecanismo del vecindario. Este mecanismo consiste simplemente en definir para cada célula de la capa de competición un conjunto de células a las que se denomina vecinas, siguiendo algún tipo de estructura espacial. Si se colocan las células en una superficie bidimensional y se decide que cada una tendrá cuatro vecinas (arriba, abajo, derecha e izquierda), se tendrá el vecindario bidimensional cuadrado de la Figura 6.11 arriba. Si se decide que también son vecinas las de arriba a la izquierda, arriba a la derecha, abajo a la izquierda, y abajo a la derecha, entonces se tiene el vecindario en estrella de la Figura 6.11 abajo.

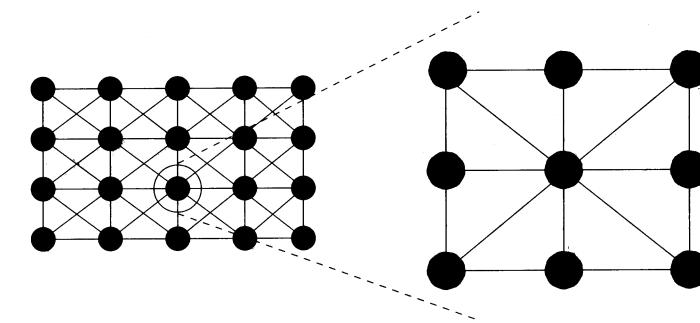
También puede haber vecindarios no bidimensionales. En la Figura 6.12 arriba aparece un vecindario unidimensional, en el que cada célula tiene tan sólo dos vecinos; y en la Figura 6.11 abajo otro tridimensional.

El vecindario debe ser definido como parte de la arquitectura de la red, siendo otro de los parámetros del sistema.

El esquema de vecindario es utilizado únicamente durante el aprendizaje. Se ha señalado que el aprendizaje en Kohonen es del tipo “winner takes all”; pues bien, esto no es exactamente así cuando se introducen vecindarios. En este caso el aprendizaje no se produce únicamente para la célula ganadora sino también para sus vecinas más próximas. La proximidad en términos de vecindario se define por el número de células que hay que atravesar para llegar desde la célula



Vecindario bidimensional cuadrado, cada celula tiene 4 vecinas



Vecindario bidimensional en estrella, cada celula tiene 8 vecinas

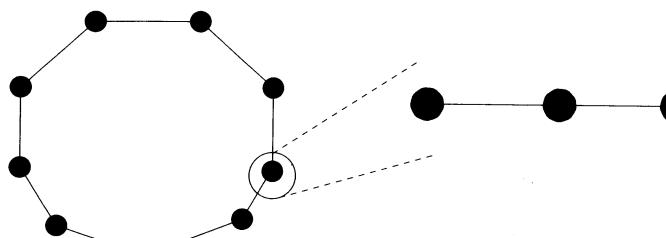
Figura 6.11: Vecindarios bidimensionales

origen a la destino, caminando por las líneas del vecindario. El vecindario define de esta forma una “distancia de vecindario”  $d(c_i, c_j)$ .

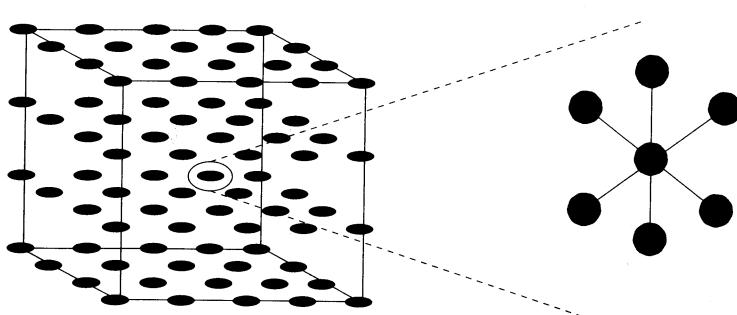
Con el vecindario, la Ecuación 6.4 que define el esquema de aprendizaje se convierte en:

$$\frac{d\mu_{ij}}{dt} = \begin{cases} \frac{\alpha(t)}{d(c_i, c_j)} (\epsilon_i(t) - \mu_{ij}(t)) & \text{si } c_i \text{ ganadora} \\ & \text{y } d(c_i, c_j) < \theta \\ 0 & \text{en caso contrario} \end{cases} \quad (6.5)$$

donde  $\theta$  es un valor entero que marca el límite del vecindario de aprendizaje, más allá del cual no se produce. Con este esquema no sólo la célula ganadora es arrastrada hacia el patrón durante el aprendizaje; también lo son todas las vecinas dentro de un radio marcado por el parámetro  $\theta$ . Como  $d(c_i, c_i) = 1$ , la Ecuación 6.5 se convierte en la anterior Ecuación 6.4 para la célula ganadora, es decir que para la célula ganadora se realiza el mismo entrenamiento que se



Vecindario unidimensional en anillo, cada celula tiene 2 vecinos



Vecindario tridimensional, cada celula tiene 4 vecinos

Figura 6.12: Vecindarios bidimensionales

ha descrito hasta ahora. El resto de las células ven disminuido su incremento, al dividirse la tasa de aprendizaje  $\alpha$  por el vecindario  $d(c_i, c_j)$ .

El efecto del vecindario es de gran utilidad. Al desplazarse para cada patrón de entrenamiento, la célula ganadora y aquellas células próximas en el vecindario serán representantes de datos parecidos, ya que estos prototipos también han sido desplazados por ellos. Al final del aprendizaje, cuando todos los patrones han sido introducidos un número suficiente de veces, ocurrirá que las células cercanas en un vecindario ocuparán posiciones cercanas en el espacio. Cuando se analiza la red, no sólo se podrá saber qué patrones son representados por qué prototipo y qué características definen a cada prototipo (su colocación en el espacio multidimensional de la entrada), sino también qué prototipos están relacionados entre sí.

Se va a ilustrar esta explicación con un ejemplo. Supóngase que se tiene un conjunto de datos sobre la población de una determinada región. De cada familia de la población se conocen diferentes características: número de miembros de la familia, renta per cápita, lugar de residencia, número de vehículos, nivel profesional de cada uno de los miembros de la familia, etc. Se desea saber si en esa sociedad existen estratos sociales definidos por los datos anteriores. Para

este ejemplo, podría utilizarse un mapa autoorganizativo de Kohonen, en el que cada patrón de entrada está constituido por las características disponibles de cada una de las familias de la población. El sistema se encargará de formar agrupaciones de datos (familias) a los que categorizará en el mismo grupo y asignará un prototipo. Las características del prototipo permitirán realizar una descripción de la categoría a la que pertenece. Así pues, podrán aparecer las categorías de las familias burguesas, aristócratas, de nivel medio, familia obrera, etc. El vecindario servirá para relacionar las categorías que han surgido entre sí, de forma que, por ejemplo, la categoría de las familias aristócratas estaría en el vecindario más próximo a la categoría de las familias burguesas que a las familias obreras.

El algoritmo general de los *mapas autoorganizativos de Kohonen* en forma de tabla quedaría de la siguiente manera:

1. Inicializar pesos.  
Asignar a los pesos valores pequeños aleatorios.
2. Presentar una nueva entrada.  
El conjunto de aprendizaje se presenta cíclicamente hasta llegar a la convergencia de la red.  
Actualizar  $\alpha$ .
3. Propagar el patrón de entrada hasta la capa de competición.  
Obtener los valores de salida de las células de dicha capa.
4. Seleccionar la célula  $C$  cuya salida sea mayor.
5. Actualizar las conexiones entre la capa de entrada y la célula  $C$ , así como las de su vecindad, según su grado de vecindad.
6. Si  $\alpha$  por encima de cierto umbral, volver al paso 2; en caso contrario FIN.

### 6.2.2. Ejemplos de aplicaciones

#### Distribuciones bidimensionales de puntos

Una demostración del funcionamiento de la red la ofrecen los *mapas topológicos bidimensionales de Kohonen*. Estos mapas simplemente distribuyen los elementos de la red a lo largo de un conjunto de ejemplos de patrones de entrada. Si además se restringe el problema a que tanto la dimensión de los patrones, como la de las células sea dos, se podrá realizar una representación gráfica de los mismos y tener una buena referencia visual de lo que está pasando en la red.

La aplicación más generalizada de los mapas autoorganizativos de Kohonen se da en tareas de clasificación no supervisada, o de extracción de características. Éste es el caso, por ejemplo, en el que, dado un conjunto de datos, se desea realizar agrupaciones a partir de una serie de características, pero se desconocen, tanto los agrupamientos que se pueden formar, como las características que

pueden definir dichos agrupamientos. Con la particularidad de que la red no sólo clasificará los datos en agrupaciones, sino que relacionará dichas agrupaciones entre sí, gracias al mecanismo de vecindario del modelo.

En el ejemplo de la Figura 6.13 se ha representado un conjunto de 2000 puntos distribuidos uniformemente a lo largo de un espacio bidimensional. Cada uno de estos puntos se corresponde con un patrón del conjunto de entrada a la red. En este caso la capa de entrada de la red consta de dos neuronas, que determinan las coordenadas  $x$  e  $y$ , respectivamente de cada uno de los puntos. En la capa competitiva se han colocado un total de 100 neuronas distribuidas en un cuadrado de 10·10, es decir que se está utilizando un vecindario cuadrado bidimensional en el cual cada célula tiene cuatro vecinos: arriba, abajo, izquierda y derecha. Cada una de las células de la capa competitiva está conectada a todas las neuronas de la capa de entrada, aquí sólo dos, y esas conexiones determinan la posición de la neurona en el espacio bidimensional de los patrones, o sea las coordenadas  $x$  e  $y$  de la neurona en dicho espacio.

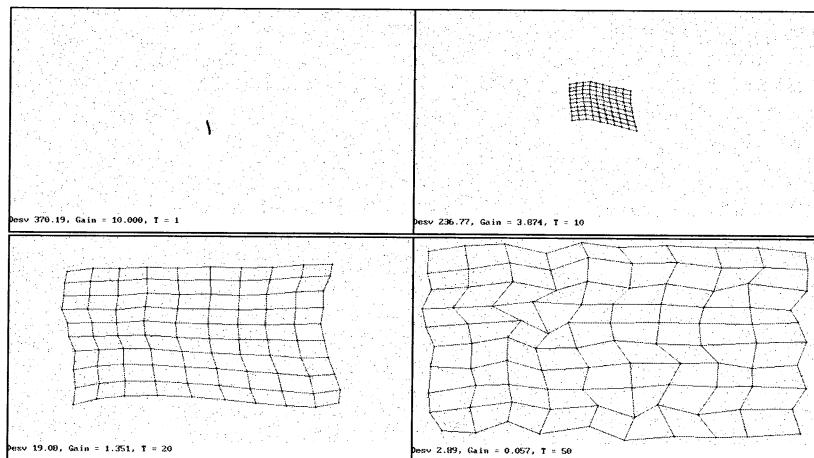


Figura 6.13: Evolución de un mapa de Kohonen para una distribución uniforme. Iteraciones 1, 15, 25 y 50, respectivamente.

Se consigue así una representación gráfica de las neuronas de la capa competitiva como puntos dentro del espacio, definidos por los valores de los pesos de las conexiones para dichas neuronas.

El significado de la distribución de las neuronas en un cuadrado es el siguiente:

- Cada célula es conectada a sus cuatro vecinas (arriba, abajo, izquierda y derecha). No se trata de una conexión física, sino de una relación entre neuronas a efectos de definición del vecindario.
- Se define un grado de vecindad, como el número de células que habrá que recorrer para llegar desde una célula a otra, incluida la de destino.

En la figura se representan las neuronas conectadas por líneas según la definición anterior.

Lo que ocurre en la simulación, como puede apreciarse en los cuatro diagramas de la Figura 6.13, es que a medida que se introducen los patrones de entrada, la red los va aprendiendo, lo cual quiere decir que varía el valor de sus conexiones. Este cambio de valor de las conexiones viene expresado gráficamente por una traslación de la neurona en cuestión. De esta manera se consigue que donde hay más densidad de puntos, exista también mayor densidad de neuronas. La conexión topológica de la capa de competición hace que la representación de las células tenga forma de rejilla, donde las líneas marcan la vecindad entre las células. Esto quiere decir que se puede comprobar si la red se está comportando de forma correcta, sólo con observar si la rejilla está extendida, es decir, si no se producen cruces entre las líneas que la componen. Esto es así debido a la influencia que cada célula tiene con sus vecinas en un grado proporcional al de su vecindad. Esta influencia provoca que el mapa se desenvuelva en forma de rejilla. No hay que olvidar que al principio la red está distribuida aleatoriamente por todo el espacio, y por tanto se encuentra totalmente "retorcida".

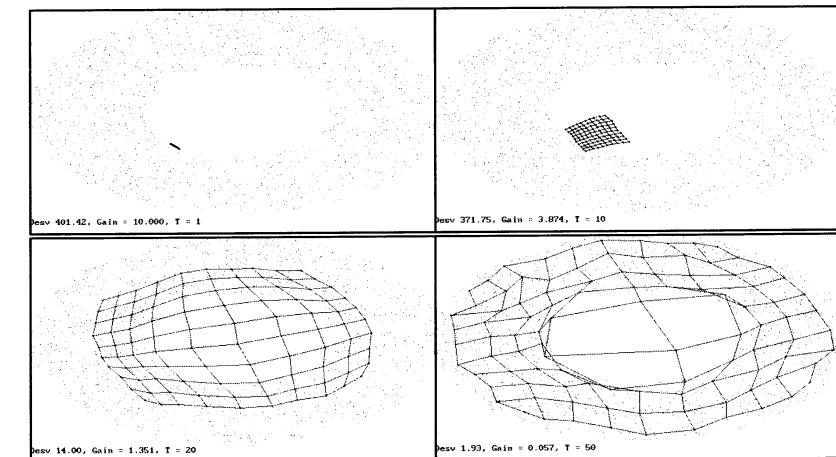


Figura 6.14: Evolución de un mapa de Kohonen para una distribución toroidal. Iteraciones 1, 10, 20 y 50, respectivamente.

Se presentan aquí tres ejemplos con otras tantas distribuciones de puntos, todas ellas de 2000 puntos y con 100 neuronas en la capa competitiva. En el primer ejemplo (Figura 6.13), se muestra el mapa topológico de Kohonen, para una distribución uniforme, en tres de sus fases, después de la primera iteración donde todas las células están en un núcleo central. Después de 15 iteraciones, se puede apreciar cómo la rejilla está totalmente desenredada, y va distribuyéndose de manera uniforme, al igual que los puntos. A continuación la rejilla aumenta de tamaño para adaptarse cada vez mejor a la distribución de puntos (iteraciones 25 y 50). Para que la rejilla fuese perfecta, es decir, en forma de rectángulos

iguales y alineados, se necesitaría un número de puntos y de células mayor.

En el segundo ejemplo se trata de una distribución toroidal con 2000 puntos. La misma red del caso anterior produjo la salida que muestra la Figura 6.14.

Aquí también se aprecia cómo la rejilla va adaptando su forma, poco a poco, a la de la distribución, apreciándose, al final, perfectamente la figura que forma la distribución de puntos. Puede darse el caso, como en este ejemplo, que haya células que no estén asociadas a ningún punto, o dicho de otro modo, que se formen prototipos que no representan a ningún ejemplo<sup>2</sup>. Estas clases se pueden pasar por alto en procesos posteriores en los que la red deba producir una respuesta a entradas desconocidas.

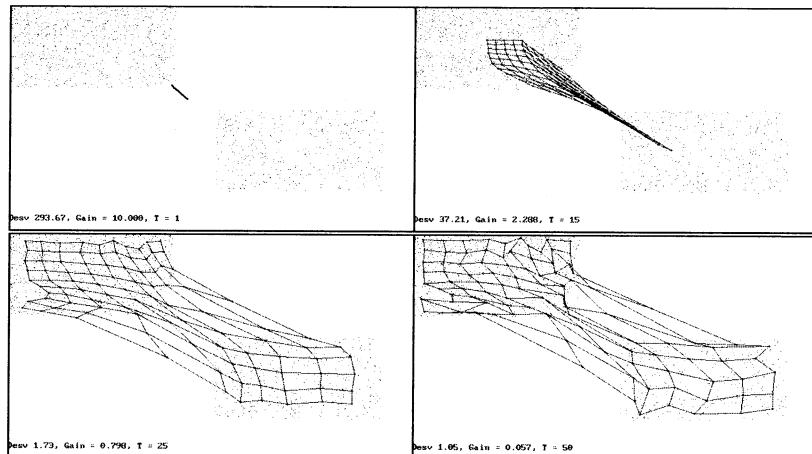


Figura 6.15: Evolución de un mapa de Kohonen paralelamente para dos distribuciones uniformes. Iteraciones 1, 15, 25 y 50, respectivamente

En el último ejemplo (Figura 6.15) se tienen dos distribuciones uniformes de diferentes tamaños: en la parte superior izquierda la distribución con mayor número de puntos y en la parte inferior derecha la de menor número de puntos.

A partir de un número no muy elevado de iteraciones, 18, se aprecia cómo la rejilla se estira formando una diagonal de izquierda a derecha y de arriba abajo, en busca de las dos distribuciones de puntos, para al final, con 88 iteraciones, comportarse como si se tratara de dos redes, una para cada distribución, sin perderse las relaciones topológicas entre los elementos de la red. Se ve, pues, claramente con estos tres ejemplos cómo una misma red de Kohonen puede clasificar de forma automática y no supervisada distintas distribuciones de puntos sin perder sus propiedades topológicas.

Se puede ver también cómo las neuronas que aparecían como vecinas al principio de la simulación, ahora ocupan posiciones contiguas dentro de la distribución (no hay cruces entre las líneas que definen el vecindario). Es decir,

<sup>2</sup>Las células que han quedado en el interior del toroide no representan a ninguna acumulación de puntos.

neuronas conectadas topológicamente adquieren, mediante el aprendizaje, características similares. De esta forma se puede saber, sin más que seguir las conexiones topológicas de las células, qué unidades comparten similares características y cuáles son, por el contrario, opuestas.

Para el caso del ejemplo esto puede parecer absolutamente trivial, pero cuando la dimensión de los datos aumenta considerablemente, el poder relacionar grupos de datos, así como poder agrupar puntos o disminuir la dimensión de los datos, es de gran importancia.

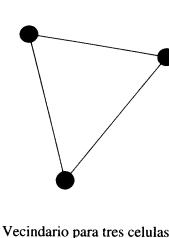
### El problema del viajante de comercio

Otro ejemplo de aplicación de los mapas de Kohonen es el descrito por Durbin y Willshaw [Durbin and Willshaw, 1987], en el que utiliza un método numérico, que más tarde Angénol [Angénol et al., 1988] modificaría aplicando una red de Kohonen, para resolver el problema del viajante de comercio, en inglés TSP (*traveling salesman problem*). El TSP es un problema clásico en el mundo de la computación debido a su complejidad, ya que muchos de los problemas de complejidad similar o incluso menor, pero de más aplicabilidad, exigen encontrar una solución al TSP.

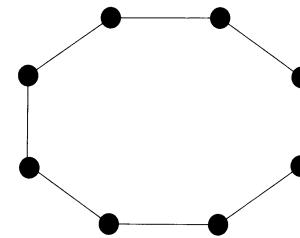
El problema del viajante consiste en describir el recorrido que debería hacer un viajante de comercio, que deba visitar  $n$  ciudades, y regresar a la ciudad de la que partió, de tal forma que recorra la menor distancia posible. Todas las ciudades tienen conexión con todas las demás, y se conocen las distancias de los diferentes tramos. La solución es una lista ordenada de ciudades, que empiecen y acaben en la misma, y que recorran el resto una sola vez. Este problema está categorizado dentro de los *np-completos* y su complejidad es factorial a la dimensión del problema. Es decir, si para resolver un TSP con 10 ciudades se tardara una milésima de segundo, para uno de 20 sería del orden de 2 años; esto, por supuesto, inspeccionando todas las soluciones posibles (búsqueda ciega). Cuando se habla de solucionar el problema, se trata de poder dar la solución óptima en un tiempo lineal al tamaño del sistema; en el caso anterior sería encontrar la solución en 2 milésimas de segundo. Actualmente no existe ningún método capaz de resolver el problema. Los métodos que existen tratan de dar buenas soluciones, en un tiempo no exponencial. La validez del método se mide por la bondad de la solución encontrada.

El método de resolución del TSP con una red de Kohonen podría ser el siguiente. Se representan las ciudades por puntos en un espacio bidimensional de la misma manera explicada en el apartado anterior. Se crea una red con sólo unos pocos nodos, por ejemplo tres, y se unen entre sí mediante una cuerda cerrada; es decir que cada célula tiene dos vecinas, una siguiente y una anterior, y esto para todas las células.

La representación de las células de la capa competitiva sería la misma que se eligió para el apartado anterior. Se aplica el método de Kohonen, con una única variante, y es que cada vez que una célula sea la ganadora de más de una ciudad, se divide a dicha célula en dos y se comprueba cuál de las dos gana la competición, y se repite el algoritmo para esta nueva arquitectura de la red (con



Vecindario para tres celulas



Vecindario para ocho celulas

Figura 6.16: Vecindario de un mapa de Kohonen para la resolución del problema del viajante

una célula más). Puede suceder que una célula que antes correspondía a una ciudad, ahora la haya perdido en favor de una ya existente o de otra que haya aparecido posteriormente. Estas células, si en un número de iteraciones no han ganado ninguna competición, se eliminan de la red. Así, la red va incorporando y eliminando células a medida que va ejecutándose el método.

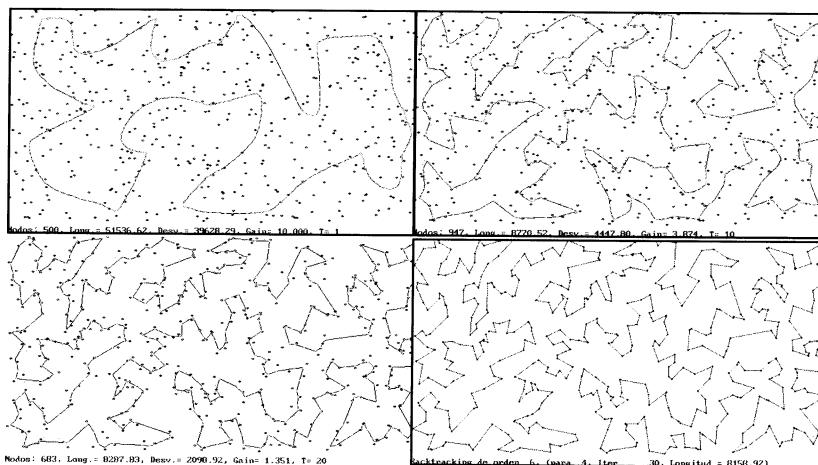


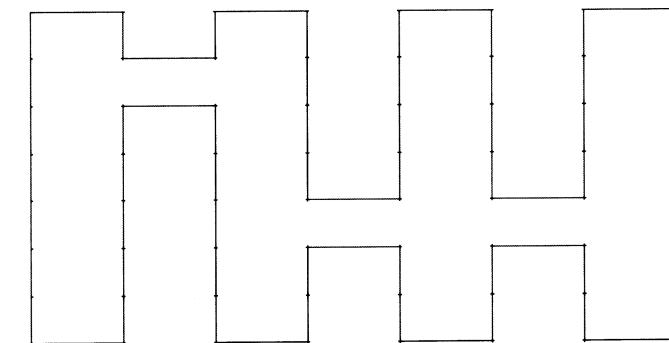
Figura 6.17: Evolución de un mapa de Kohonen para resolver un TSP de 500 ciudades. Las figuras se corresponden con 1, 10, 20 y 88 iteraciones, respectivamente

De esta forma se evita que existan más células que ciudades, y el algoritmo esté realizando cálculos innútiles; recordemos que cada vez que se presenta un patrón, éste es comparado con todas las células de la capa de competición. Al mismo tiempo, se garantiza que al final habrá exactamente el mismo número de células que de ciudades, es decir, que cada célula corresponda a cada ciudad,

y al final las coordenadas de la ciudad y de la célula sean las mismas. Esto es así debido a que el valor de la tasa de aprendizaje, o de olvido, va disminuyendo, al igual que la vecindad. De este modo, al finalizar el proceso, cuando haya el mismo número de ciudades que de células, el vecindario será uno, y no existirá arrastre por vecindario; simplemente cada célula caerá gravitacionalmente hacia su ciudad correspondiente, de la que es la ganadora cuando se presenta a la red como patrón.

Lo que ocurre en la red es que, a medida que se van presentando las ciudades, aquella célula que se encuentre más cercana a la ciudad, será la ganadora de la competición, y se acercará a la ciudad, arrastrando en su movimiento a las células vecinas. El camino se irá acercando cada vez más a las ciudades, hasta coincidir con todas ellas. Además, el camino, que vendrá dado por el mapa topológico de las células de la capa competitiva, se irá adaptando de manera elástica a una solución en que la vecindad evita que se produzcan cruces.

Como se aprecia en la Figura 6.17, dos células, a pesar de estar muy próximas físicamente, si no son vecinas topológicamente, en este caso vecinas en el recorrido, tienen escasa influencia la una respecto de la otra y se evita así que esa ciudad pueda atraer a células que aun estando más próximas no lo sean en el recorrido, lo cual daría resultados incorrectos.



Nodos: 64, Long. = 2820.00, Desv. = 89.68, Gain = 0.203

Figura 6.18: Solución exacta del problema del TSP para 64 ciudades distribuidas en rejilla, aplicando una red de Kohonen

Este es un buen ejemplo de la importancia de las características topológicas de la red. En este caso la solución no está en la lista de puntos y a qué clase pertenecen, ni tampoco en la posición (características) de cada uno de los prototipos que la red ha producido. Se hace que haya tantos prototipos como ciudades, y que se coloquen cada uno encima de una de las ciudades. La solución es precisamente el orden, en el vecindario, que tienen las ciudades; es decir, para cada ciudad la solución la dan quienes son sus vecinos. Éste es un caso extremo

en el que el vecindario es lo único importante. Existen muchos otros ejemplos en los que la clasificación es necesaria y el vecindario es muy útil, sobre todo para interpretar los resultados.

En la Figura 6.17 se pueden ver los resultados obtenidos para una dimensión de 500 ciudades, en tres fases de evolución de la simulación: al principio, con una sola iteración, en la que se aprecia que existe un número elevado de células pero es menor que el de ciudades; después de 9 iteraciones, donde “la goma” está más estirada y ya se atisba la solución que nos propondrá el método, y al final, después de 65 iteraciones, donde ya cada célula ha coincidido con una ciudad y produce la solución final. En la figura se ve cómo la solución producida es muy buena; no tiene cruces y es difícilmente mejorable a mano. Pero no tiene por qué ser la solución óptima; el método no lo garantiza, pero sí se trata de un buen método de búsqueda de solución aproximada al TSP, en un tiempo de unos minutos (para el caso del ejemplo).

Por último, en la Figura 6.18 se ve cómo el método es capaz de obtener soluciones óptimas. En este caso, al estar las ciudades distribuidas en una cuadrícula, es fácil comprobar si la solución es óptima; basta con verificar que todas las líneas que conectan las ciudades son horizontales o verticales, que no hay diagonales, como en el caso de la figura para 64 ciudades.

### 6.3. Teoría de la resonancia adaptativa

La Teoría de la Resonancia Adaptativa nace de la necesidad de abordar lo que Grossberg definió como el dilema de la estabilidad-plasticidad [S., 1980, S., 1982b, S., 1982a], y que él mismo enunció de la siguiente manera:

*Por un lado, el hecho de haberme puesto a vivir en Los Ángeles no debe impedirme el poder encontrar, sin esfuerzo, la casa de mis padres en Boston, cuando en el futuro vaya a visitarlos. Por otro, el hecho de haber crecido en Boston no debe impedirme el poder aprender lugares y direcciones de Los Ángeles al irme a vivir allí.*

Es decir, se trata de diseñar un sistema que reaccione de forma **plástica** a nuevos datos, pero de manera **estable** a aquellos que sean irrelevantes. Se debe también poder comutar entre estos dos estados, plástico y estable, cuando sea necesario, para evitar la degradación de lo ya aprendido.

Esto requiere que el sistema tenga incorporada alguna estimación que le permita decidir cuándo un dato es irrelevante y cuándo no, puesto que el sistema es de aprendizaje no supervisado, y no existe ninguna señal externa que permita discernir entre datos.

#### 6.3.1. Arquitectura ART 1

Grossberg hace una distinción, siguiendo la terminología utilizada en psicología, entre dos tipos de memoria: la memoria a corto plazo y la memoria a largo plazo:

- Memoria a corto plazo (STM, Short Term Memory). Se refiere a la capacidad de un sistema nervioso de recordar algo que ha ocurrido dentro de un intervalo reciente de tiempo, independientemente del número de veces que haya ocurrido. Se identifica en términos de Redes de Neuronas Artificiales, con los valores de activación de las neuronas.
- Memoria a largo plazo (LTM, Long Term Memory). Se refiere a la capacidad de recordar cosas que ocurrieron en un intervalo distante de tiempo, con tal de que hayan ocurrido el suficiente número de veces. Computacionalmente podría ser el equivalente a los pesos de las conexiones.

Para solucionar el problema de la estabilidad-plasticidad, Grossberg introduce en su modelo dos subsistemas y un término de control:

- Subsistema atencional. Es básicamente un modelo de Red de Neuronas con aprendizaje competitivo, que se encarga de reaccionar ante nuevas entradas y de aprenderlas en caso necesario.
- Subsistema orientador. Se encarga de distinguir entre entradas relevantes e irrelevantes. Es el encargado de comunicarle al sistema atencional que una entrada debe ser aprendida.
- Término de control. Se trata de un sistema que lleva el control de los módulos que van a actuar y gobierna las señales entre los distintos módulos.

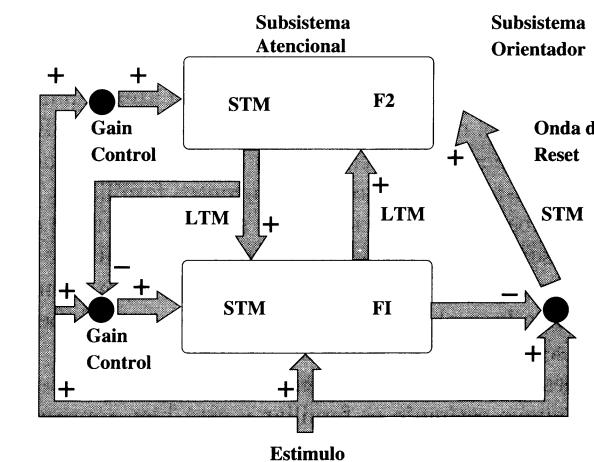


Figura 6.19: Arquitectura de un sistema ART

En la Figura 6.19 se puede ver la arquitectura general de un sistema ART. Como se aprecia, la arquitectura básica es la de un sistema de aprendizaje

competitivo con dos capas  $F_1$  y  $F_2$ . La capa  $F_1$  es, al igual que en el aprendizaje competitivo, la capa de entrada; su dimensión es la misma que la de la entrada y se encarga de propagar la señal de entrada a la capa superior  $F_2$ . Esta última es básicamente una capa competitiva donde existen los mismos tipos de conexiones que en el aprendizaje competitivo; es decir, cada célula inhibe a todas las demás con una fuerza fija e igual para todas, y recibe una señal excitatoria de sí misma también con la misma fuerza.

Uno de los principales problemas que tiene el aprendizaje competitivo es el de la inestabilidad. El sistema puede ser fácilmente perturbado por entradas aleatorias, o algo ruidosas, que afectarían a las ya aprendidas. Es decir, el sistema no permanece estable una vez aprendido un conjunto elevado de datos, sino que fácilmente puede producir salidas incorrectas si se introducen en el sistema nuevos datos irrelevantes. Grossberg ya indicaba que este problema podría solucionarse introduciendo en el sistema mecanismos de aprendizaje recurrentes y de correlación entre los estímulos aprendidos y los que se presenten al sistema.

Por tanto, en el ART se añaden a las conexiones existentes entre la capa  $F_1$  y la  $F_2$ , llamadas conexiones *bottom-up*, otras nuevas entre la capa  $F_2$  y la  $F_1$ , llamadas conexiones *top-down*. Estas nuevas conexiones, poco habituales entre los sistemas neuronales artificiales, permiten un mecanismo de aprendizaje de valores esperados o predicciones con los que poder comparar los datos de entrada y que pueden producir reacciones orientadoras en situaciones de desigualdad, como se describirá más adelante.

Usando estos mecanismos, un sistema ART puede generar códigos de reconocimiento de forma adaptativa y sin la presencia de ningún profesor externo. Puede también, de modo adaptativo, conmutar entre un estado estable y uno plástico de forma que sea capaz de aprender eventos significativamente nuevos sin perder la estabilidad ante aquellos que sean irrelevantes. Es capaz, pues, de distinguir entre eventos familiares y no familiares o esperados y no esperados.

Los eventos familiares se procesan en el subsistema atencional de forma parecida a como se realiza en el modelo de aprendizaje competitivo. Este subsistema también es capaz de aprender las predicciones *top-down* que ayudarán a estabilizar los códigos *bottom-up* aprendidos de los eventos familiares.

Sin embargo, el subsistema atencional por sí solo es incapaz de mantener al mismo tiempo representaciones estables de categorías familiares y crear nuevas categorías para estímulos no familiares que puedan ir apareciendo. Para realizar esta labor se introduce el subsistema orientador que inhibe al subsistema atencional cuando se recibe un estímulo no familiar.

Las interacciones que se producen entre estos dos subsistemas ayudan a decidir cuándo un nuevo estímulo es familiar y está bien representado por uno de los códigos de reconocimiento, y cuándo no es familiar y necesita un nuevo código de reconocimiento.

La arquitectura del modelo recuerda a la ya descrita del aprendizaje competitivo: dos capas -la de entrada,  $F_1$ , y la de competición,  $F_2$ - que tendrá tantas células como prototipos de los datos existan. En el modelo de Kohonen, comentado con anterioridad, es necesario conocer a priori el número de clases en que se quiere agrupar los datos. Este parámetro se convierte en crítico, y el resulta-

do de la clasificación es muy dependiente del mismo. Desgraciadamente, en la clasificación no supervisada no es posible conocer el número de clases a formar, ya que se desconoce el carácter de los datos. En los modelos habituales, dicho número se determina a posteriori, después de que la red ha realizado la categorización, a partir de los resultados obtenidos. En el modelo ART, sin embargo, es el propio modelo el encargado de determinar el número de categorías a partir del perfil de los datos de entrada. Como dicho número viene determinado por la cantidad de células de la capa  $F_2$ , esta capa inicialmente está vacía, y va incluyendo nuevas unidades a medida que se necesitan nuevos prototipos.

### 6.3.2. Funcionamiento del modelo

A continuación se detalla el funcionamiento del modelo, describiendo las señales que se van produciendo a la llegada de un dato de entrada arbitrario. La entrada llega a tres lugares al mismo tiempo (Figura 6.20): a la capa  $F_1$ , al término de control del subsistema atencional y al subsistema orientador. El subsistema orientador, recibe además una señal inhibitoria de la capa  $F_1$ . Dicha señal es emitida de forma constante, y habitualmente toma un valor elevado. El alto nivel de esta señal inhibitoria hace que el subsistema orientador esté inhibido siempre que sea necesario.

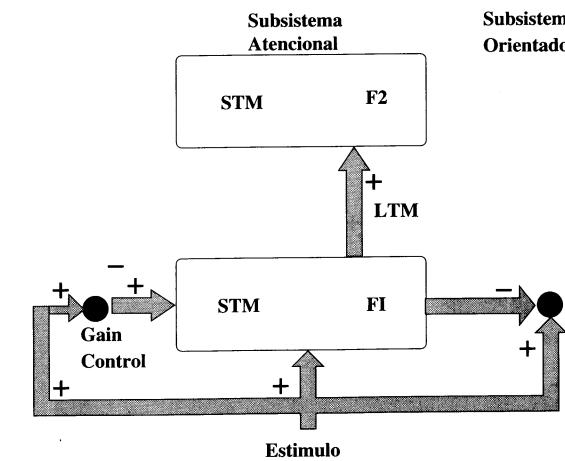


Figura 6.20: Propagación *bottom-up*

El subsistema orientador es el encargado de comunicar al subsistema atencional que el dato que acaba de llegar es nuevo, relevante, y por lo tanto debe ser aprendido. Esto se realizará cuando la señal inhibitoria procedente de la capa  $F_1$  caiga por debajo de un determinado valor  $\rho$ , llamado factor de vigilancia.

La capa  $F_1$  recibe tres entradas: una procedente de su término de control, otra de las conexiones *bottom-up*, y la otra que es directamente el dato de entrada. Sin embargo, nunca recibe más de dos a la vez; a esto se le denomina la

regla de los dos tercios (2/3). Más adelante se describirá el mecanismo de esta regla.

Por su parte, el término de control recibe dos entradas, una proveniente de la entrada directamente y la otra de las conexiones *top-down*. En un primer momento, sólo recibe el dato de entrada, y lo propaga directamente a la capa  $F_1$ . De esta forma, al introducir el patrón, la capa  $F_1$  recibirá el mismo dato dos veces. En la capa  $F_1$ , mediante un mecanismo que se describirá más adelante, se realiza siempre una comparación entre las dos entradas que recibe, que en esta primera fase son idénticas (al ser ambas el dato de entrada) y producen una salida alta, al ser la comparación exacta, que inhibe por completo al subsistema orientador.

Posteriormente, la señal es propagada hasta la capa  $F_2$  y se pondera a través de las conexiones *bottom-up* (Figura 6.20). Al ser  $F_2$  una capa de competición, funciona igual que en el modelo de aprendizaje competitivo descrito anteriormente; es decir, la señal es propagada por la capa  $F_2$ , hasta que una de sus unidades gana la competición, la célula ganadora. En el modelo de aprendizaje competitivo el proceso se detiene en este punto, y se procede al aprendizaje para la célula ganadora. En el modelo ART, la célula ganadora no representa el prototipo asignado a la entrada, sino un posible candidato. La célula ganadora es la predicción realizada por el subsistema atencional, que debe ser verificada.

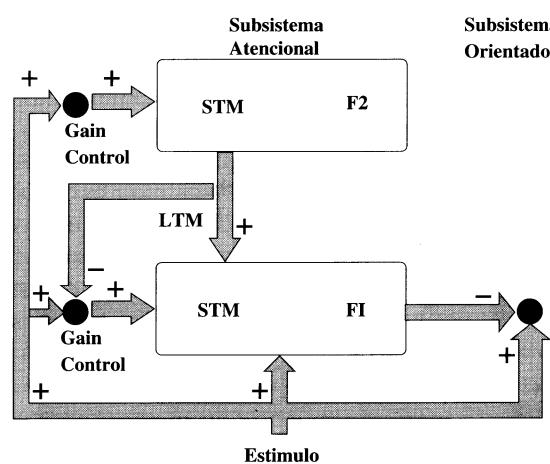


Figura 6.21: Propagación *top-down* de la señal

A continuación, se describirá el mecanismo de verificación de la predicción realizada por la capa  $F_2$ . En este punto, la célula ganadora emite una señal, a través de las conexiones *top-down*, que llega a la capa  $F_1$  (Figura 6.21). En este momento es preciso explicar el mecanismo de la regla 2/3. Al final de la propagación de la señal de  $F_2$  a  $F_1$ , a esta última cabría esperar que llegaran tres señales, la que se acaba de mencionar y la entrada por dos veces. Sin embargo, la capa  $F_2$  emite también una señal inhibitoria hacia el término de control

(Figura 6.21), que lo inhabilita, y evita que propague la señal de entrada hacia  $F_1$ . Por lo tanto, en este instante  $F_1$  recibirá de nuevo dos entradas, pero ahora en vez de ser la entrada repetida, es la entrada y la señal propagada por las conexiones *top-down*.

Después, el comportamiento es el mismo que ya ha sido descrito: la capa  $F_1$  realiza una comparación entre las dos entradas que recibe, que ya no tienen por qué ser idénticas, y produce una salida inhibitoria hacia el subsistema orientador, proporcional al resultado de la comparación. Si la señal de las conexiones *top-down* es muy similar a la entrada, entonces el subsistema orientador es inhibido. Esto quiere decir que la predicción realizada sobre el prototipo elegido para la entrada es correcta, y se procede a adaptar las conexiones, tanto las *top-down* como las *bottom-up*, utilizando el dato de entrada. Se aprende la entrada recibida siguiendo una serie de ecuaciones que se describirán más adelante. Sin embargo, si la comparación no resulta exitosa, la señal producida no será capaz de inhibir el subsistema orientador, y éste se pondrá en funcionamiento. Es este caso, será porque la predicción realizada por la red es incorrecta, y habrá que elegir otro candidato para ser el representante de la entrada. El que la comparación resulte o no exitosa depende del valor del parámetro de vigilancia. Para valores muy elevados en cuanto la comparación arroja una ligera divergencia, el resultado cae por debajo de  $\rho$  y no se acepta el prototipo seleccionado. Esto se corresponde con un sistema exigente, en el que los patrones se agrupan en la misma categoría sólo si comparten muchas de sus características, si son patrones muy parecidos. A medida que se reduce el valor de  $\rho$  el sistema se hace cada vez menos exigente, y pueden estar en la misma clase datos algo más diferentes. El usuario puede determinar la *exigencia* que desea que tenga su sistema, asignando un valor al factor de vigilancia.

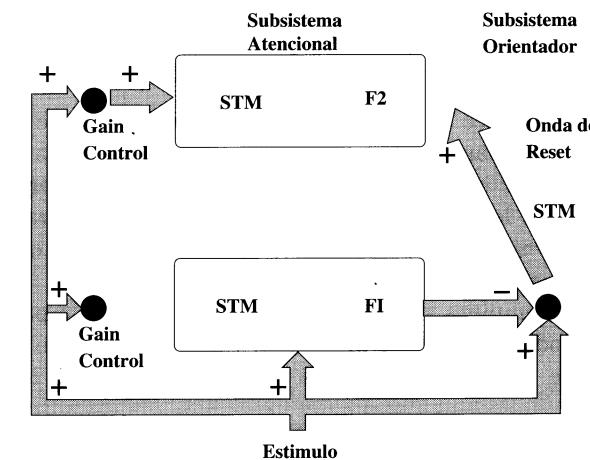


Figura 6.22: Onda de reset, producida al fallar la predicción

El funcionamiento del subsistema orientador es el siguiente. Cuando se de-

tecta que una predicción no es buena, el subsistema orientador se pone en funcionamiento enviando una señal, denominada "onda de reset", a la capa  $F_2$ . Esta señal hace que en la capa  $F_2$ , la célula ganadora quede inactivada temporalmente, hasta la llegada de una nueva entrada. Al eliminarse la célula ganadora para esta entrada se producen dos efectos: por una parte, no puede volver a ser elegida, lo cual es aceptable ya que fue una predicción errónea, y si no fuera eliminada el sistema la elegiría de nuevo; y por otra, desaparece la señal de las conexiones *top-down*. Por lo tanto, el término de control no será inhibido por dichas conexiones, y en la capa  $F_1$  se recibirá de nuevo duplicada la entrada, con lo cual se repite todo el proceso. Se dice entonces que el sistema entra en resonancia, y de ahí el nombre del sistema.

Se puede apreciar que todo el sistema funciona de forma asíncrona; no existe una unidad central que guíe todo el proceso, sino que son las propias señales las encargadas de que todo se realice según el plan previsto. Esto es debido a que el modelo nació para ser un reflejo de ciertos mecanismos que se producen en el cerebro, y a que debía implantarse en computadores con arquitecturas paralelas.

Así pues, la onda de reset produce la eliminación temporal de la célula ganadora y permite que el proceso se repita y que el sistema realice otra predicción, con la esperanza de que sea más acertada. Pero puede suceder que la nueva predicción vuelva a ser inadecuada. En este caso, la nueva célula ganadora es también eliminada temporalmente y el proceso se repite una y otra vez hasta que o bien se encuentre una célula que sea una predicción adecuada, o bien todas las células de la capa  $F_2$  hayan sido eliminadas temporalmente. Si esto ocurre, el sistema inserta una nueva célula, que se etiqueta como ganadora de la entrada que se está procesando, y se procede a su aprendizaje. Esto hace que el prototipo que acaba de ser incorporado se ajuste perfectamente a la entrada aprendida.

Cuando se introduce una nueva entrada en el sistema, las células de la capa  $F_2$  eliminadas temporalmente vuelven a estar operativas, y todo el proceso se repite como se ha descrito.

De todo esto se deduce que en las conexiones *bottom-up* se almacenan los códigos de predicción, para poder acceder de forma directa a las categorías correctas, mientras que en las conexiones *top-down* se almacenan los valores de los prototipos, con los que se va a comparar la entrada para verificar si la predicción realizada es correcta, es decir, si el prototipo elegido es o no un buen representante de la entrada recibida.

Como el sistema empieza sin células en la capa  $F_2$ , será la primera entrada la encargada de producir el primer prototipo, y a medida que se van recibiendo nuevas entradas, se van creando nuevos prototipos según las necesidades del sistema.

### 6.3.3. Ecuaciones del modelo ART 1

La inicialización del sistema se realiza siguiendo las siguientes ecuaciones:

$$t_{ij}(0) = 1$$

$$b_{ij}(0) = \frac{1}{1 + N}$$

$$\begin{aligned} 0 &\leq i \leq N - 1 & N &= \text{número de nodos de } F_1 \\ 0 &\leq j \leq M - 1 & M &= \text{número de nodos de } F_2 \\ 0 &\leq \rho \leq 1 \end{aligned}$$

En estas ecuaciones  $b_{ij}$  es el peso de la conexión *bottom-up*, y  $t_{ij}$  es el peso de la conexión *top-down*, entre el nodo de entrada  $i$  y el nodo de salida  $j$  en el tiempo  $t$ . Estos pesos definen el ejemplar especificado por el nodo de salida  $j$ . Por último,  $\rho$  es el factor de vigilancia.

Más formalmente, el funcionamiento del sistema se puede describir según el mismo esquema de seguimiento de la propagación de una entrada cualquiera  $X$ . La entrada  $X$  se transforma en un patrón  $S$  de activaciones a través de los nodos, o de forma más abstracta *directores de características*, de  $F_1$  (Figura 6.20). Esta señal se propaga a través de las conexiones *bottom-up* mediante la suma ponderada de la señal  $S$  por los valores de los pesos de las conexiones *bottom-up*, hasta transformarse en una nueva señal  $T$ . A la transformación de  $X$  en  $T$  se la llama filtro adaptativo. Esta señal de entrada  $T$  se transforma rápidamente en otra señal  $\mu$  en la capa  $F_2$  a través de las conexiones excitatorias e inhibitorias existentes en dicha capa. En el modelo ART 1 todo este proceso podría describirse mediante la ecuación:

$$\mu_j = \sum_{i=0}^{N-1} b_{ij}(t)x_i, \quad 0 \leq j \leq M - 1$$

siendo  $\mu$  la salida de la célula  $j$  de la capa  $F_2$ , y  $x_i$  el elemento  $i$ -ésimo de la entrada, que para el modelo ART 1 toma valores binarios (1 o 0).

Tan pronto como se produce la transformación de  $X$  en  $\mu$ , la capa  $F_2$  envía esta señal hacia abajo a través de las conexiones *top-down*, a partir, exclusivamente, de la célula ganadora que se determina mediante:

$$\mu_j^* = \max_j(\mu_j)$$

De la misma forma que ocurría en la transformación de  $X$  en  $T$ , ahora se obtiene una nueva señal  $V$  que será el resultado de realizar la suma ponderada de la señal  $\mu$  por los valores de los pesos de las conexiones *top-down*. La transformación de  $\mu$  en  $V$  será, por tanto, un nuevo filtro adaptativo. La señal  $V$  se llamará valor predictivo.

$$V_i = g(\mu_j^*) \cdot t_{ji}$$

donde:

$$g(\mu_j) = \begin{cases} 1 & \text{si } j \text{ ganadora} \\ 0 & \text{en caso contrario} \end{cases}$$

y  $t_{ji}$  es el valor de la conexión entre la célula  $j$  de la capa  $F_2$  y la  $i$  de la capa  $F_1$ .

Ahora  $F_1$  se verá perturbada por dos fuentes: la entrada  $X$  y el valor predictivo  $V$  (Figura 6.20). Esto hará que los valores de activación de la célula  $F_1$  varíen de los que tenía al principio al recibir la entrada  $X$ ; de esta forma, se mide el grado de semejanza entre la entrada  $X$  y el valor predictivo  $V$ , lo que permite resolver el dilema de la estabilidad-plasticidad.

La capa  $F_1$  está relacionada con el subsistema orientador a través de una conexión inhibitoria que calcula todos los valores de activación de las células de  $F_1$  (Figuras 6.19, 6.20, 6.21 y 6.22).

Al recibir la entrada, puesto que ésta también se introduce en el término de control que está conectado a su vez con la capa  $F_1$ , en  $F_1$  se produce una comparación del estímulo consigo mismo, lo cual da un valor alto de la señal inhibitoria hacia el subsistema orientador que quedará por lo tanto inactivo, como se ha explicado.

La comparación se realiza en el modelo ART 1 mediante la ecuación:

$$I = C(X, Y) = \frac{1 + \|X \cdot Y\|}{1 + \|X\|} = \frac{1 + \sum_{i=0}^{N-1} x_i \cdot y_i}{1 + \sum_{i=0}^{N-1} x_i}$$

Como se puede ver, la señal de inhibición  $I$  es máxima cuando se compara la entrada consigo misma, ya que al ser valores binarios  $x_i \cdot x_i = x_i$ ,  $\forall i$ , y por lo tanto:

$$I = C(X, X) = \frac{1 + \|X \cdot X\|}{1 + \|X\|} = \frac{1 + \sum_{i=0}^{N-1} x_i \cdot x_i}{1 + \sum_{i=0}^{N-1} x_i} = \frac{1 + \sum_{i=0}^{N-1} x_i}{1 + \sum_{i=0}^{N-1} x_i} = 1$$

Más adelante, cuando  $F_1$  reciba la señal predictiva  $V$ , el término de control quedará inhibido por las señales *top-down* que recibe (Figuras 6.20 y 6.21). Con lo cual, en  $F_1$  se producirá la comparación entre la entrada  $X$  y el valor predictivo  $V$ , reduciéndose la inhibición de la capa  $F_1$  hacia el subsistema orientador.

$$I = C(X, V) = \frac{1 + \|X \cdot V\|}{1 + \|X\|} = \frac{1 + \sum_{i=0}^{N-1} x_i \cdot v_i}{1 + \sum_{i=0}^{N-1} x_i} <= 1$$

Si dicha señal inhibitoria cae por debajo del factor de vigilancia, el valor predictivo  $V$  se considera erróneo, y no debe ser asumido por el código de reconocimiento. En este caso, el subsistema orientador envía una señal a la capa  $F_2$  del subsistema atencional comunicándole el error, lo que produce la temporal desaparición de la capa  $F_2$  del código de reconocimiento elegido, es decir, la célula ganadora.

$$I = \frac{1 + \|X \cdot V\|}{1 + \|X\|} < \rho$$

Entonces se dispara el subsistema orientador. Una vez producido esto se repite todo el proceso tantas veces como sea necesario para que uno de los valores predictivos sea correcto; el sistema entra en **resonancia**. En el proceso de resonancia puede ocurrir que se eliminan todos los códigos de reconocimiento

de la capa  $F_2$ , y por tanto no se pueda continuar con la búsqueda de nuevos códigos. Recuérdese que el sistema en este caso crea un nuevo código para la entrada, de tal forma que la correlación entre dicha entrada y el valor predictivo que se acaba de crear sea total. Se dice que se ha producido una nueva situación relevante que ha dado lugar a la creación de un nuevo código de reconocimiento para posteriores entradas con características similares.

Ya sólo queda describir el mecanismo de adaptación de las conexiones *top-down* y *bottom-up*. Estas últimas siguen un esquema similar al del modelo de aprendizaje competitivo:

$$\Delta b_{ig} = \frac{t_{gi} \cdot x_i}{0,5 + \sum_{i=0}^{N-1} t_{gi} \cdot x_i}$$

$$i = 0, 1, \dots, N - 1$$

donde  $g$  es la célula ganadora.

La Ecuación 6.2 del aprendizaje competitivo podría formularse en este caso como:

$$\Delta b_{ig} = \frac{x_i}{\sum_{i=0}^{N-1} x_i}$$

La diferencia es aquí la introducción del valor 0,5 en el denominador para evitar divisiones por infinito y suavizar el aprendizaje, además de la introducción como multiplicando del valor del elemento correspondiente de las conexiones *top-down*, que en la ecuación clásica del aprendizaje competitivo no existía. Como se ha señalado anteriormente, el vector de conexiones *top-down* almacena el valor del prototipo correspondiente. En este caso es incluido en el aprendizaje *bottom-up*, cuyas conexiones son las encargadas de realizar las predicciones de códigos de reconocimiento, para que futuras predicciones den con prototipos más acertados, y evitar así la resonancia. De esta forma, al final de muchos ciclos de aprendizaje, el sistema puede llegar a estabilizarse y producir predicciones exactas, eliminar la resonancia, y que todos los prototipos correctos sean seleccionados de forma directa, evitando la búsqueda y comparación con el resto de prototipos.

Para ello es necesario que cada elemento del vector de conexiones no se actualice proporcionalmente al valor de su elemento del dato de entrada correspondiente, sino que se actualicen únicamente aquellos elementos cuya entrada coincide con la correspondiente en el vector del prototipo, almacenado en el vector  $t$ . Esto se verá mejor con un ejemplo como el de la Figura 6.23.

En este caso se tienen los siguientes vectores: el vector de entrada es  $X = \{1, 1, 0, 1\}$ , el de conexiones *bottom-up* entre la entrada y la célula ganadora es  $b_g = \{0,8, 0,4, 0,2, 0,1\}$  y el de conexiones *top-down* también con la célula ganadora es  $t_g = \{0, 1, 1, 1\}$ . Los únicos valores que coinciden entre la entrada  $X$  y las conexiones *top-down* son el segundo y el último, y serán los únicos aprendidos en sus conexiones *bottom-up*:

$$\sum_{i=0}^{N-1} t_{gi} \cdot x_i = 0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 = 2$$

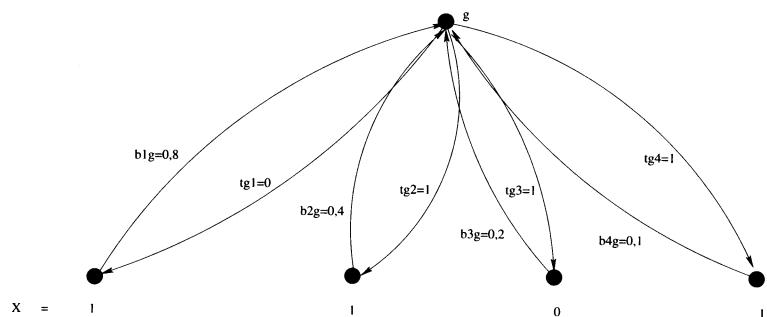


Figura 6.23: Ejemplo de una red ART 1 para la célula ganadora, con 4 células en la entrada

$$\Delta b_{1g} = \frac{t_{g1} \cdot x_1}{0,5 + 2} = \frac{0}{2,5} = 0$$

$$\Delta b_{2g} = \frac{t_{g2} \cdot x_2}{0,5 + 2} = \frac{1}{2,5} = 0,4$$

$$\Delta b_{3g} = \frac{t_{g3} \cdot x_3}{0,5 + 2} = \frac{0}{2,5} = 0$$

$$\Delta b_{4g} = \frac{t_{g4} \cdot x_4}{0,5 + 2} = \frac{1}{2,5} = 0,4$$

Este esquema tiene la ventaja de que al actualizar únicamente aquellas conexiones en las que coinciden los valores de la entrada y el prototipo, se está reforzando que esta coincidencia se dé en el futuro. Es decir que en posteriores apariciones de entradas en las que se den dichas coincidencias, el prototipo seleccionado tendrá aún más probabilidades de ser seleccionado a la primera, ya que producirá valores de activación mayores en la capa  $F_2$ .

El nuevo vector de conexiones *bottom-up* sería  $b_g = \{0,8,0,8,0,2,0,5\}$ .

Las conexiones *top-down* se actualizan siguiendo la ecuación:

$$t_{gi}(t+1) = t_{gi}(t) \cdot x_i$$

$$i = 0, 1, \dots, N - 1$$

Esto es realmente aplicar una función AND entre la entrada y las conexiones *top-down* del instante de tiempo anterior. En el ejemplo anterior, el nuevo vector *top-down* quedaría  $t_g = \{0,1,0,1\}$ .

### 6.3.4. Características de los modelos ART

Los sistemas ART tienen cuatro características esenciales que los distinguen de otros modelos de categorización:

1. Unidades computacionales autoescalables. El contexto de una entrada puede de afectar a su modo la clasificación, de tal manera que las señales que aparezcan en un dato pueden ser tratadas como ruido en unos casos, pero pueden ser parte del propio dato en otro contexto. En la Figura 6.24 se aprecia cómo partes de la señal de entrada que son tratadas como ruido, cuando afectan a un sistema en un cierto grado de autoorganización, pueden ser tratadas como señal cuando perturban al mismo sistema en un grado de diferente autoorganización. En el primer caso hay dos entradas que difieren en dos puntos, pero en este contexto esa diferencia hace que las entradas puedan ser consideradas como diferentes, y aprendidas de forma independiente. En el segundo caso también hay dos entradas que difieren también en dos puntos, pero ahora el sistema reconoce esa diferencia como ruido y considera que las dos son realmente la misma entrada.

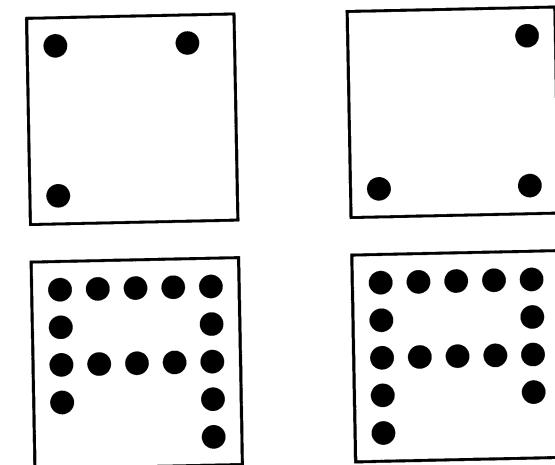


Figura 6.24: El ruido es tratado de forma independiente al contexto

El modelo, de forma automática, autoescala sus unidades computacionales para incluir al contexto y ser sensible a definiciones de señal y de ruido dependientes del aprendizaje.

2. Acceso a memoria autoajustable. No existe ningún método de búsqueda de las señales en el sistema; por el contrario, el modelo es capaz de realizar una búsqueda paralela, que de forma adaptativa actualice el orden de búsqueda para mantener la eficiencia a medida que los códigos de reconocimiento se hacen más complejos debido al aprendizaje. En otros términos,

el aumento en el número de prototipos no ralentiza el proceso de categorización de nuevos datos, ya que como parte del sistema está el realizar una búsqueda en paralelo y ordenada del prototipo más representativo del dato de entrada.

3. Acceso directo a los códigos de reconocimiento. Una de las características más importantes del funcionamiento de un sistema neuronal es la velocidad a la que es capaz de reconocer objetos familiares de entre un número ingente de posibilidades. En un modelo ART, a medida que el aprendizaje se convierte en autoconsistente y predictivamente exitoso, el proceso de búsqueda es más directo. No importa cómo sea de grande y de complejo el código aprendido; los datos familiares acaban por disparar de forma directa el prototipo que los representa, sin búsqueda, así como aquellos datos no familiares cuyo parecido sea suficientemente alto.
4. El entorno como profesor. En un sistema ART, si a un error de reconocimiento se lo acompaña de una penalización, el sistema se hará más vigilante. Esto se traduce por un cambio en el estado atencional del sistema, lo cual incrementará la sensibilidad entre las diferencias de los datos y los valores predictivos.

Se va a ilustrar la descripción del modelo con un par de ejemplos: un reconocedor binario de caracteres y un reconocedor de caras.

### 6.3.5. Ejemplo, reconocedor de caracteres

El ejemplo consiste en comprobar si una red ART 1 es capaz de crear, a partir de un conjunto de datos que son la representación de las letras del alfabeto latino, una categoría para cada una de las letras, de forma que sea capaz de reconocerla, incluso cuando después se introduzca una letra con ruido. Las entradas a las redes ART 1 han de ser binarias, por lo que habrá que elegir una representación para los caracteres binaria. En la Figura 6.25 se muestra una posible representación.

En esta representación, cada carácter está definido por una rejilla de  $5 \times 7$ , 35 componentes, que pueden estar encendidos o apagados. Esta combinación produce como salida el carácter deseado y genera un vector de 35 componentes, el que aparece al pie de cada carácter en la Figura 6.25 y que se utilizará como entrada a la red. De esta forma, al introducir el carácter *A* a la red, se estará introduciendo el vector de 35 componentes:

$$A = \{ \begin{array}{l} 1,1,1,1,1, \\ 1,0,0,0,1, \\ 1,0,0,0,1, \\ 1,1,1,1,1, \\ 1,0,0,0,1, \\ 1,0,0,0,1, \\ 1,0,0,0,1 \end{array} \}$$

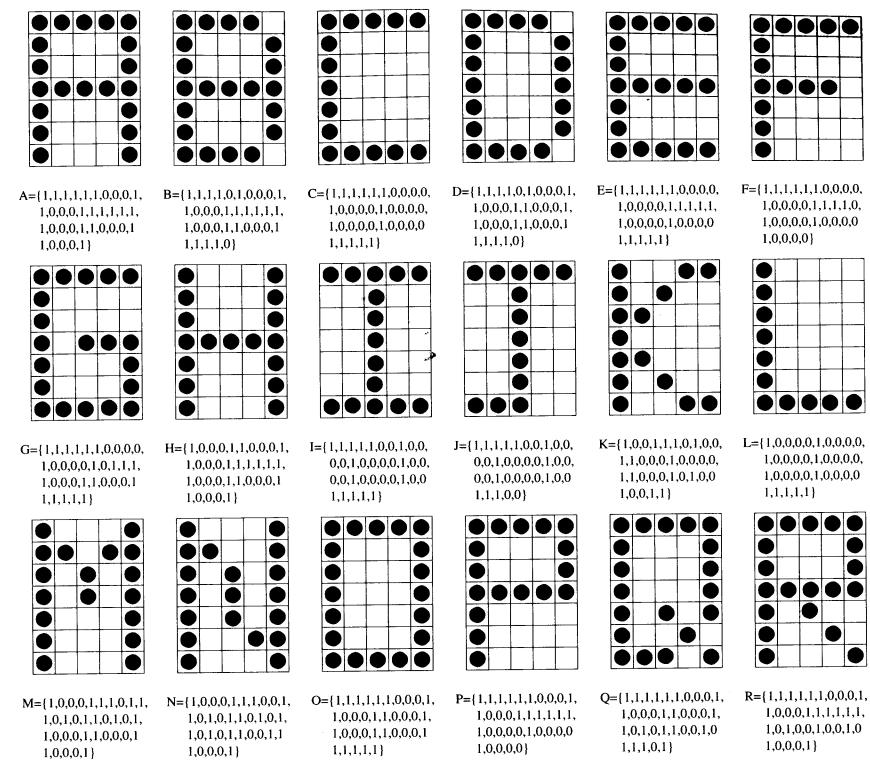


Figura 6.25: Datos de entrada para reconocedor de caracteres binario

A continuación se va a mostrar la evolución de una red ART 1 a medida que se le van introduciendo los vectores de datos descritos. En primer lugar se introduce el carácter *A*; al estar la red vacía, se crea una célula en  $F_2$  a la que se llamará  $c_1$ . Luego se introduce el carácter *B*. La red da como ganadora a la única categoría existente, como no podría ser de otra forma, y las conexiones de arriba abajo corroboran esta predicción, con lo cual *B* es asignado a  $c_1$ . Lo mismo ocurre con los caracteres *C* y *D*. Sin embargo, al introducirse el carácter *E*, la predicción de la red no es corroborada, se activa el subsistema orientador, y como no hay más categorías en  $F_2$ , se crea una nueva  $c_2$ . El carácter *F* produce como salida  $c_1$ . El *G* primero se asocia a  $c_1$ , pero el subsistema orientador no lo verifica se asocia con  $c_2$ ; que sí es verificado por el subsistema orientador, y por tanto es la salida finalmente producida para este carácter, y así sucesivamente. En el Cuadro 6.1 se muestra la evolución de la red tras la introducción de todos los caracteres del conjunto de entrenamiento. En cada fila, hay información acerca del carácter introducido, y en cada columna información acerca de las categorías de la capa  $F_2$ . El código utilizado es el siguiente. Un asterisco indica que el carácter ha sido asociado a esa categoría existente previamente, la *N*; que

la categoría en cuestión acaba de ser creada para reconocer a dicho carácter. Una  $N$  sólo puede aparecer si en esa misma fila las casillas anteriores están llenas. Un número indica que el carácter había sido asociado en primer término con la categoría, pero que después fue descartado; el valor indica en qué lugar fue asociado.

Cuadro 6.1: Resultados de la primera iteración del reconocedor de caracteres

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
N	*	*	*	1	*	1		1		*	*	2	1	1	*	1	1		
				N		*	*	2		*	*	1	1	2		2	*	*	
					N	*						3							
												N	*	*	*	*	*	*	

Así, en el Cuadro 6.1, para el carácter  $M$ , resulta que en primer lugar fue asociado a la categoría  $c_2$  (el 1 está en la segunda fila) y fue descartado; luego a la categoría  $c_1$  (el 2 está en la primera fila) y posteriormente a la  $c_3$  (el 3 está en la tercera fila), y en ambos casos fue descartada dicha asociación, y como no había más categorías en  $F_2$ , se creó una nueva,  $c_4$ , para asociarlo (la  $N$  está en la cuarta fila).

Cuadro 6.2: Resultados de la segunda iteración del reconocedor de caracteres

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
1	*	*	1	*	1							*	2						
3		3		3															
2		2		2		*	*	*	*	*	*	*	3			1	*	*	
4				*									1						
N				*									*			*			

Al final de la introducción de todos los patrones la red ha creado cuatro categorías. En gran parte de ellas se ha producido resonancia. La categoría predicha en primer lugar no ha sido la que al final se ha dado como salida. Además, el número de categorías de la red ha aumentado; al principio no había ninguna y pasa a cuatro categorías. Todo esto son pruebas de que la red no está estabilizada y es necesario volver a introducir todos los patrones. En el Cuadro 6.2 se muestra el resultado de una nueva iteración de todos los ejemplos de entrenamiento.

Después de esta nueva iteración se pueden apreciar varias cosas. En primer lugar, se ve cómo los caracteres que antes producían una salida ahora han cambiado de elección. Por ejemplo, la  $A$  en la iteración anterior fue asociada a la

categoría  $c_1$  y después de la segunda iteración se la asocia a la categoría  $c_4$ . Esto es debido a que cada vez que una entrada es asociada definitivamente con una categoría, la entrada es aprendida, modificándose las conexiones existentes, y abriendo la posibilidad de que más tarde una entrada asociada a ella produzca peor salida que otra categoría próxima. Además, se ha creado una nueva categoría,  $c_5$ , concretamente para asociar al carácter  $B$  que ya no es recogido por ninguna de las categorías que existían.

Después de esta última iteración todavía existen algunos casos en los que se sigue produciendo resonancia, por lo que será necesario volver a iterar.

Cuadro 6.3: Resultados de la tercera iteración del reconocedor de caracteres

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
		*	*			*				*	*	*	*	*	*	*	*	*	*
	*				*		*			*	*	*	*	*	*	*	*	*	*

Después de la tercera iteración ya no se han creado nuevas categorías, ni se han producido resonancias (Cuadro 6.3), como lo demuestra el hecho de que sólo hay asteriscos en el cuadro. Ahora, la red está estabilizada y su salida es definitiva. Se han creado, por tanto, cinco categorías: la categoría 1 para los caracteres  $C, D, F$  y  $P$ ; la categoría 2 para los caracteres  $K$  y  $L$ ; la categoría 3 para los caracteres  $I, J, M, N, O$  y  $T$ ; la categoría 4 para los caracteres  $A, H$  y  $R$ ; y la categoría 5 para los caracteres  $B, E, G, Q$  y  $S$  (Cuadro 6.4).

Cuadro 6.4: Resultados final de la clasificación con un factor de vigilancia de 0,6

Clase	Caracteres				
$C_1$	$C$	$D$	$F$	$P$	
$C_2$	$K$	$L$			
$C_3$	$I$	$J$	$M$	$N$	$T$
$C_4$	$A$	$H$	$R$		
$C_5$	$B$	$E$	$G$	$Q$	$S$

Se comprueba que esta clasificación es coherente, en una misma categoría están caracteres próximos en la representación, sin embargo no es el objetivo planteado en un primer lugar. Esto es debido a que el modelo ART 1 es no supervisado, y por lo tanto no existe un objetivo a priori preciso para la red, en un ejemplo como el que se trata se deben utilizar modelos de clasificación

Cuadro 6.5: Resultados de la primera iteración del reconocedor de caracteres con factor de vigilancia 0.9

supervisados, de los que se hablará en el capítulo de clasificación. Este ejemplo se ha elegido únicamente para ilustrar el modelo. Recuérdese que es posible cambiar la clasificación realizada por la red para hacerla más exigente. En este caso, sólo hay que incrementar el valor del factor de vigilancia. En el caso anterior este parámetro valía 0,6.

Para ilustrar la influencia del factor de vigilancia en la clasificación, se va a repetir el proceso anterior para una red con un factor de vigilancia de 0, 9. Los resultados después de la primera iteración son los que muestra en el Cuadro 6-5.

Lo primero que se aprecia es que se han producido un mayor número de categorías, como resultado de haber indicado a la red mayor exigencia. Ahora que el factor de vigilancia es mucho mayor, para que dos ejemplos pertenezcan a la misma clase, se deben parecer mucho más que antes. Es por esto por lo que, en la mayoría de los casos, al introducir un nuevo ejemplo se crea una categoría nueva para él, como se ve en el hecho de que se hayan producido 12 categorías, habiéndose introducido 20 datos de entrada. Esto también hace que haya más resonancias y que la red deba ser iterada más veces. Los resultados tras la segunda iteración son los que se muestran en el Cuadro 6.6.

El número de categorías ha aumentado hasta 15, sólo tres más que antes, y las resonancias también han disminuido. Ya hay muchas entradas reconocidas de forma directa: *D*, *F*, *H*, *K*, *L*, *M*, *N*, *O*, *P*, *R*, *S* y *T*. De todos modos es necesaria una nueva iteración (Cuadro 6.7).

En este caso, todos los caracteres han sido reconocidos de forma directa excepto el *I*, para el que ha sido necesario crear una nueva categoría. Esta categoría sólo reconocerá a este carácter, debido a la forma en que se realiza el aprendizaje y a que el resto de los caracteres ya están asociados de forma directa sin resonancia. Por lo tanto, una nueva iteración produciría la misma

Cuadro 6.6: Resultados de la segunda iteración del reconocedor de caracteres con factor de vigilancia 0.9

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
1	3			1	*	5		9							*	10			
	1	2	*			6		7								3			
	2	1		3		2		5								5			
	5	*			2	1	*	2								4			
	9						*	13								11			
	11							1	1							14			
	12							11		*						12			
	10							12				*				6			
	6							6								1			
	8						*	3								2			
2	7							10								9			*
	4							4								7			*
	N							8		*						8			
								N								13			
																N			

Cuadro 6.7: Resultados de la tercera iteración del reconocedor de caracteres con factor de vigilancia 0,9

tabla, pero esta vez sin resonancia para el carácter *I*. Éste será, por tanto, el resultado de la clasificación para un factor de vigilancia de 0,9:

Clase	Caracteres	
$C_1$	<i>F</i>	<i>P</i>
$C_2$	<i>D</i>	
$C_3$	<i>L</i>	
$C_4$	<i>C</i>	
$C_5$	<i>H</i>	
$C_6$	<i>T</i>	
$C_7$	<i>K</i>	
$C_8$	<i>M</i>	<i>N</i>
$C_9$	<i>O</i>	
$C_{10}$	<i>G</i>	
$C_{11}$	<i>A</i>	<i>R</i>
$C_{12}$	<i>E</i>	<i>S</i>
$C_{13}$	<i>B</i>	
$C_{14}$	<i>J</i>	
$C_{15}$	<i>Q</i>	
$C_{16}$	<i>I</i>	

Se aprecia que se han creado 16 categorías, y sólo cuatro contienen más de un carácter; es decir, el aumento del factor de vigilancia ha hecho que la red sea más sensible a las entradas y sea capaz de distinguir entre casi la totalidad de los caracteres. Por otra parte, en el caso de las categorías 1, 8, 11 y 12, que son las que recogen dos caracteres, estos caracteres son muy parecidos; por ejemplo, la *F* y la *P*, o la *M* y la *N*. Aumentando aún más el factor de vigilancia se produciría una separación total de los caracteres de entrada. La clasificación no supervisada tiene como objetivo crear categorías numerosas. Vemos cómo esto se logra en el ART 1, ya que para conseguir separarlas es necesario alcanzar valores del parámetro de vigilancia anormalmente elevados.

## Capítulo 7

# PREDICCIÓN DE SERIES TEMPORALES

## 7.1. Introducción

Una serie temporal se puede definir como una colección de datos o valores de un suceso determinado a lo largo del tiempo,  $\{x(t)\}_{t \in \mathbb{R}^+}$ , donde  $t$  representa la variable tiempo y  $x(t)$  el valor de la serie en dicho instante de tiempo. Las series temporales se caracterizan porque su evolución temporal no depende explícitamente de la variable tiempo, sino de los valores de la serie en instantes anteriores de tiempo o incluso de otras variables temporales que pudieran afectar a la evolución de la serie.

En la Figura 7.1 se muestran dos ejemplos de series temporales. La Figura 7.1(a) muestra la serie temporal conocida como serie logística cuyo comportamiento viene dado por la ecuación en diferencia que se indica en dicha figura. Se caracteriza porque para  $a = 3,97$  y  $x(0) = 0,5$ , la serie presenta un comportamiento fuertemente caótico. En la Figura 7.1(b) se muestra otra serie temporal que representa la evolución del número de manchas solares observadas desde 1700 hasta 1995. En este caso, no se conoce la ecuación en diferencias que describe el comportamiento de dicha serie, y cada muestra indica el promedio del número de manchas solares observadas anualmente por diferentes observadores.

La predicción del comportamiento en el futuro de una serie temporal es un objetivo importante en muchas áreas de aplicación, como puede ser la física, la biología, negocios, etc. Así, por ejemplo, la predicción de la serie que describe la evolución del número de manchas solares observadas anualmente (Figura 7.1(b)), permite estudiar la actividad solar, que a su vez influye en las transmisiones radioeléctricas o en fenómenos como el descenso térmico global de la Tierra.

En ocasiones, es posible construir modelos mediante la aplicación de ciertas leyes físicas, los cuales permiten describir el comportamiento o evolución tem-

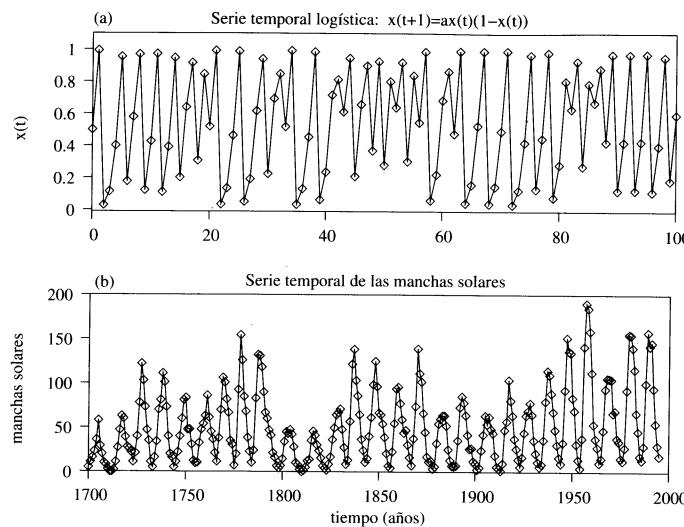


Figura 7.1: Ejemplos de series temporales: (a) Serie temporal logística; (b) Serie temporal de las manchas solares

poral del fenómeno estudiado. Dichos modelos consisten, generalmente, en un conjunto de ecuaciones diferenciales o ecuaciones en diferencia que expresan el valor de la serie temporal en un instante de tiempo como una función de sus valores anteriores. Estos modelos pueden utilizarse para predecir el comportamiento de la serie en el futuro. Sin embargo, para la mayor parte de las series temporales que describen fenómenos reales -manchas solares, consumo eléctrico, precio del oro en el mercado, temperatura, etc.- no es fácil ni inmediato conocer las ecuaciones que describen la relación explícita entre el valor de la serie en un instante de tiempo y sus valores anteriores, sino que sólo se dispone de un conjunto de datos observados. Estos datos requieren una interpretación para poder construir un modelo que aproxime la evolución de la serie temporal y permita predecir su comportamiento en el futuro.

Las redes de neuronas artificiales han sido ampliamente utilizadas en los últimos años en el marco de predicción de series temporales. Esto es debido a características propias de las redes de neuronas, como son:

- La capacidad de las redes de neuronas para aproximar y capturar relaciones a partir de un conjunto de ejemplos, sin necesidad de conocer información adicional sobre la distribución de los datos.
- La capacidad de las redes de neuronas para construir relaciones no lineales.
- La capacidad de las redes de neuronas para construir relaciones a partir de información incompleta o información con ruido.

- Los modelos basados en redes de neuronas son fáciles de construir y utilizar.

En este capítulo se van a presentar modelos de predicción de series temporales basados en redes de neuronas artificiales supervisadas. El interés se centra en modelos construidos con redes de neuronas estáticas, como el PERCEPTRON multicapa o las redes de base radial, pues son modelos fáciles de construir y utilizar.

## 7.2. Problema de predicción

De manera genérica, el problema de predicción de series temporales se puede formular del siguiente modo: a partir de un conjunto de muestras de la serie, conocer los valores en el futuro, es decir, su evolución o comportamiento a lo largo del tiempo. En este contexto, se suelen distinguir dos casos:

### ■ Predicción en un paso de tiempo

La predicción en un paso de tiempo consiste en predecir el valor de la serie en el instante de tiempo inmediatamente siguiente al instante actual  $t$ , a partir de las muestras disponibles hasta dicho instante de tiempo. Es decir, predecir el valor  $x(t + 1)$  utilizando un cierto número de muestras anteriores  $x(t)$ ,  $x(t - 1)$ ,  $x(t - 2)$ , ..., número que depende de la serie temporal.

### ■ Predicción en múltiples pasos de tiempo

La predicción en múltiples pasos de tiempo consiste en predecir el comportamiento de la serie, no únicamente en el instante inmediatamente siguiente al instante actual  $t$ , sino en un futuro más lejano, concretamente en el llamado intervalo de predicción  $[t + 1, t + h + 1]$ , siendo  $h$  un número natural que representa el horizonte de predicción. Es decir, consiste en predecir los valores de la serie  $x(t + 1)$ ,  $x(t + 2)$ , ... y  $x(t + h + 1)$  a partir de la información disponible en el instante de tiempo actual  $t$ .

Como se ha comentado en la introducción, cuando no se dispone de las ecuaciones que describen el comportamiento de la serie temporal, es necesario recurrir a métodos aproximativos para construir modelos que permitan resolver el problema de predicción, bien se trate de predicción en un paso de tiempo o en múltiples pasos de tiempo. La dificultad de predicción de series temporales depende, obviamente, del comportamiento dinámico de dicha serie. Existen series que poseen comportamientos periódicos y estables que pueden ser explicados y predichos utilizando modelos basados en técnicas clásicas y lineales. Sin embargo, existen otras series temporales más complejas, para las cuales las técnicas clásicas podrían resultar deficientes.

En este capítulo el interés se centra en series temporales cuyo comportamiento puede ser descrito por modelos NAR o modelos no lineales de regresión. Estos modelos se caracterizan porque recogen el comportamiento temporal de

la serie expresando el valor de la serie en el instante de tiempo  $t + 1$  como una función no lineal de  $r + 1$  valores de la serie temporal en instantes anteriores de tiempo, es decir:

$$x(t+1) = F(x(t), x(t-1), \dots, x(t-r)) + \varepsilon(t) \quad (7.1)$$

donde  $t$  es la variable discreta tiempo;  $\varepsilon(t)$  es un error residual que se asume ruido blanco Gausiano; y  $F$  es una función no lineal desconocida y que, por tanto, debe ser estimada o aproximada a partir de un conjunto de datos observados de la serie temporal,  $\{x(t)\}_{t=0, \dots, N}$ .

Por tanto, la construcción de modelos NAR involucra la determinación de la función  $F$ , a partir del conjunto de muestras disponibles, mediante técnicas de aproximación, entre las cuales se encuentran las redes de neuronas artificiales.

### 7.3. Modelos neuronales para la predicción de series temporales

Como se ha comentado en el Capítulo 5, tanto el PERCEPTRON multicapa como las redes de base radial son modelos de redes de neuronas considerados estáticos, ya que su arquitectura está diseñada para encontrar relaciones entre patrones de entrada y salida independientes de la variable tiempo. Sin embargo, esto no impide que puedan utilizarse para tratar información temporal y abordar así el problema de predicción de series temporales. Para ello, basta que la entrada a la red esté compuesta no sólo del valor del patrón en un determinado instante de tiempo, sino también de una cierta historia de dicho patrón, la cual se representa utilizando una secuencia finita temporal en el pasado. Los modelos de predicción construidos con redes de neuronas estáticas son modelos simples y fáciles de construir, de manera que el interesado en analizar el comportamiento de las redes de neuronas en la predicción de series temporales puede recurrir a dichos modelos.

El problema de predicción de series temporales puede abordarse también con estructuras dinámicas de redes de neuronas recurrentes. Éstas se caracterizan porque la propia arquitectura de la red está diseñada para el procesamiento de información temporal, permitiendo que la salida de la red no sólo dependa de la entrada, sino de estados anteriores de la propia red. Esto hace que las redes recurrentes estén preparadas para capturar comportamientos dinámicos, sin necesidad de presentar a la red una secuencia de valores anteriores de la serie. Sin embargo, las redes recurrentes suelen ser modelos más complejos de construir y, en general, menos utilizados en el contexto de predicción de series temporales. No obstante, en la literatura aparecen trabajos en los que se utilizan redes de neuronas recurrentes o con ciertas componentes recurrentes para abordar el problema de predicción de series temporales, obteniendo resultados satisfactorios. Así, por ejemplo, en [Hertz et al., 1991] y [Yu and Bang, 1997] el problema de predicción se aborda utilizando el modelo conocido con el nombre

de red de FIR (Finite Impulse Response); en [Stage and Sendhoff, 1997] se propone una modificación de la red de Elman para predicción de series temporales; en [Galván and Isasi, 2001] se presenta una arquitectura de red parcialmente recurrente diseñada con el propósito de predicción en múltiples pasos de tiempo, etc.

A continuación, se van a presentar los mecanismos para abordar el problema de predicción de series temporales utilizando los modelos de redes de neuronas estáticas, PERCEPTRON multicapa y redes de base radial. Como se ha comentado en capítulos anteriores, ambas arquitecturas de redes son aproximadores universales, por lo que pueden utilizarse para la construcción de modelos NAR (Ecuación 7.1), si bien el PERCEPTRON multicapa ha sido más utilizado que las redes de neuronas de base radial.

#### 7.3.1. Predicción en un paso de tiempo

Considerando el vector  $(x(t), x(t-1), \dots, x(t-r))$  como patrón de entrada, las redes de neuronas estáticas pueden utilizarse para aproximar la función  $F$  en la Ecuación (7.1), obteniendo el siguiente modelo de predicción:

$$\tilde{x}(t+1) = \tilde{F}(x(t), x(t-1), \dots, x(t-r)) \quad (7.2)$$

donde  $\tilde{F}$  representa una aproximación neuronal de la función  $F$  y  $\tilde{x}(t+1)$  es la predicción proporcionada por el modelo neuronal en el instante de tiempo  $t + 1$ .

Como se muestra en la Figura 7.2, la red de neuronas tiene  $r + 1$  neuronas de entrada que reciben los valores de la serie temporal en los  $r + 1$  instantes anteriores, y una neurona de salida que representa la predicción en un paso de tiempo de la serie temporal.

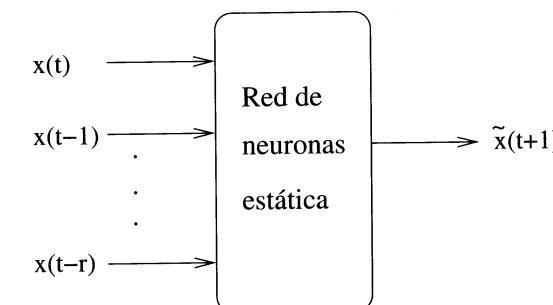


Figura 7.2: Predicción en un paso de tiempo

Los patrones para el entrenamiento de la red de neuronas se obtienen del conjunto de muestras disponibles de la serie temporal del siguiente modo: dado el conjunto  $\{x(t)\}_{t=0, \dots, N}$  y dado el valor de  $r$ , cada patrón se construye utilizando los  $r + 1$  valores anteriores de la serie, como se muestra en el cuadro 7.1. De este

Cuadro 7.1: Patrones de entrenamiento: Predicción en un paso de tiempo

	Entrada	Salida deseada
Patrón 1	$x(r), x(r-1), \dots, x(1), x(0)$	$x(r+1)$
Patrón 2	$x(r+1), x(r), \dots, x(2), x(1)$	$x(r+2)$
Patrón 3	$x(r+2), x(r+1), \dots, x(3), x(2)$	$x(r+3)$
Patrón 4	$x(r+3), x(r+2), \dots, x(4), x(3)$	$x(r+4)$
...	...	...
Patrón $N-r$	$x(N-1), x(N-2), \dots, x(N-r), x(N-(r+1))$	$x(N)$

modo, cada patrón de entrada recoge la información temporal necesaria para predecir el valor de la serie en el instante de tiempo  $t+1$ ,  $\forall t = r, \dots, N-1$ .

Al realizar el entrenamiento, se pretende que la red capture la relación entre el valor de la serie en un instante de tiempo y los  $r$  valores pasados. El ajuste de los pesos se realiza para minimizar el error cuadrático medido en la salida, es decir:

$$e(t+1) = \frac{1}{2} (x(t+1) - \tilde{x}(t+1))^2 \quad \forall t = r, \dots, N-1 \quad (7.3)$$

Una vez realizado el entrenamiento de la red, el modelo puede utilizarse con el propósito de predecir en un paso de tiempo; basta presentar a la red los valores de la serie en los  $r+1$  instantes anteriores de tiempo.

Uno de los problemas del uso de estos modelos estáticos para la predicción de series temporales es la definición de la longitud de la secuencia temporal que hay que presentar como patrón de entrada a la red, es decir, la definición del valor de  $r$ . Dicho valor dependerá de cada serie temporal y será necesario realizar una estimación. Esta estimación puede llevarse a cabo realizando un análisis previo de la serie temporal o de modelos existentes que expliquen parcialmente el comportamiento de la serie. En último caso, y ante la falta de información, mediante prueba y error, lo cual implica entrenar diferentes arquitecturas de redes de neuronas variando el número de entradas a la red, hasta conseguir una arquitectura capaz de resolver el problema de predicción.

### 7.3.2. Predicción en múltiples pasos de tiempo

Partiendo de que el interés en este capítulo se centra en series temporales que pueden ser descritas mediante modelos NAR (Ecuación 7.1), existen dos modos sencillos de abordar el problema de predicción en múltiples pasos de tiempo, ambos basados en el uso de redes de neuronas estáticas, PERCEPTRON multicapa y redes de base radial.

Sea  $h > 1$  el horizonte de predicción. Como se ha comentado anteriormente, la predicción en múltiples pasos de tiempo consiste en predecir el valor de la serie en el instante de tiempo  $t+h+1$  o en el intervalo de tiempo  $[t+1, t+h+1]$

a partir de los valores de la serie hasta el instante de tiempo actual  $t$ . Los esquemas para la predicción de series temporales en múltiples pasos de tiempo son los siguientes:

- **Esquema de predicción 1**

Este primer esquema consiste en utilizar de manera recurrente el modelo neuronal construido para la predicción en un paso de tiempo dado por la Ecuación (7.2). Dicho modelo necesita como entrada los valores de la serie en los  $r+1$  instantes de tiempo inmediatamente anteriores. Por tanto, si el objetivo es predecir el valor de la serie en el instante de tiempo  $t+h+1$ , dicha predicción viene dada por la siguiente expresión:

$$\tilde{x}(t+h+1) = \tilde{F}(x(t+h), x(t+h-1), \dots, x(t+h-r))$$

Sin embargo, en el instante actual de tiempo  $t$  no toda la información de entrada a la red está disponible, pues los valores de la serie  $x(t+h), x(t+h-1), \dots, x(t+1)$  para  $h > 1$ , no se conocen. Para afrontar este problema se utilizan como entradas al modelo neuronal los valores predichos por la red de neuronas en instantes anteriores de tiempo  $\tilde{x}(t+h), \tilde{x}(t+h-1), \dots, \tilde{x}(t+1)$  en lugar de los valores de la serie temporal  $-x(t+h), x(t+h-1), \dots, x(t+1)$ . De este modo, el modelo dado por la Ecuación (7.2) puede utilizarse con el propósito de predicción en múltiples pasos de tiempo retroalimentando la salida de la red hacia la entrada, como se muestra en la Figura 7.3.

Por tanto, las predicciones de la serie temporal en el futuro se obtienen del siguiente modo:

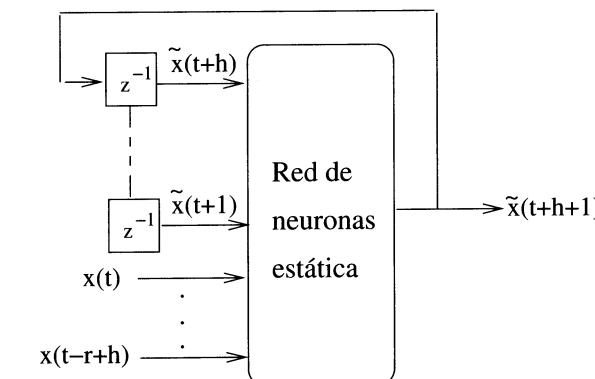


Figura 7.3: Predicción en múltiples pasos de tiempo: Esquema 1

$$\tilde{x}(t+h+1) = \tilde{F}(x(t), x(t-1), \dots, x(t-r)) \quad (7.4)$$

$$\tilde{x}(t+2) = \tilde{F}(\tilde{x}(t+1), x(t), x(t-1), \dots, x(t-r+1)) \quad (7.5)$$

$$\tilde{x}(t+3) = \tilde{F}(\tilde{x}(t+2), \tilde{x}(t+1), x(t), \dots, x(t-r+2)) \quad (7.6)$$

⋮

$$\tilde{x}(t+h+1) = \tilde{F}(\tilde{x}(t+h), \tilde{x}(t+h-1), \dots, \tilde{x}(t+1), x(t), \dots, x(t-r+h)) \quad (7.7)$$

El principal inconveniente de este esquema de predicción es que el aprendizaje de la red de neuronas ha sido realizado con el propósito de predecir en un paso de tiempo, es decir, para minimizar los errores locales dados por la Ecuación (7.3). Durante el entrenamiento, la red ha capturado las relaciones entre la muestra actual y las muestras observadas en instantes anteriores de tiempo. Sin embargo, cuando el modelo se utiliza para predecir en múltiples pasos de tiempo, un grupo de neuronas de entrada reciben los valores de la serie temporal predichos por la red en instantes anteriores de tiempo, de manera que los errores cometidos en la predicción de dichos valores son propagados hacia las predicciones futuras, pudiendo influir negativamente en la calidad de dichas predicciones.

#### ■ Esquema de predicción 2

Otra forma de abordar el problema de predicción en múltiples pasos de tiempo utilizando redes de neuronas estáticas consiste en construir un modelo NAR para predecir, directamente, el valor de la serie en el instante  $t + h + 1$  a partir de la información disponible en el instante actual  $t$ :

$$x(t+h+1) = G(x(t), x(t-1), \dots, x(t-d)) \quad (7.8)$$

donde  $G$  es una función no lineal que relaciona el valor de la serie en el instante  $t + h + 1$  con una secuencia finita de valores de la serie temporal disponibles en el instante de tiempo actual  $t$ . A diferencia del modelo para predecir en un paso de tiempo (Ecuación 7.1), en este caso, el valor de la serie en el instante  $t + h + 1$  no se predice utilizando los  $r + 1$  valores inmediatamente anteriores, sino información disponible en el instante actual.

Utilizando como entrada el vector  $(x(t), x(t-1), \dots, x(t-d))$ , las redes de neuronas estáticas se pueden utilizar para aproximar la relación  $G$ , obteniendo el siguiente modelo de predicción:

$$\tilde{x}(t+h+1) = \tilde{G}(x(t), x(t-1), \dots, x(t-d)) \quad (7.9)$$

En este caso, la salida de la red representa la predicción de la serie en el instante  $t + h + 1$ , como se muestra en la Figura 7.4.

Los patrones para el entrenamiento de la red se obtienen de las muestras disponibles de la serie temporal,  $\{x(t)\}_{t=0, \dots, N}$ , y considerando como salida deseada para el patrón en el instante de tiempo  $t$ , el valor de la serie en el

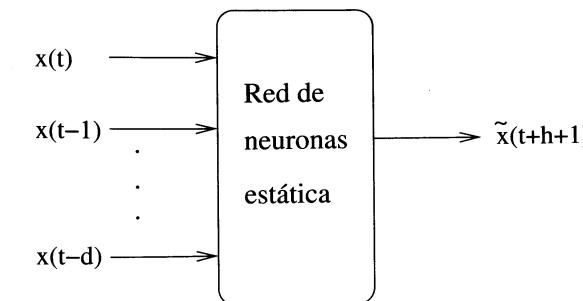


Figura 7.4: Predicción en múltiples pasos de tiempo: Esquema 2

Cuadro 7.2: Patrones de entrenamiento: Predicción en múltiples pasos de tiempo utilizando el Esquema 2

	Entrada	Salida deseada
Patrón 1	$x(d), x(d-1), \dots, x(1), x(0)$	$x(d+h+1)$
Patrón 2	$x(d+1), x(d), \dots, x(2), x(1)$	$x(d+h+2)$
Patrón 3	$x(d+2), x(d+1), \dots, x(3), x(2)$	$x(d+h+3)$
Patrón 4	$x(d+3), x(d+2), \dots, x(4), x(3)$	$x(d+h+4)$
...	...	...
Patrón $N-d-h$	$x(N-h-1), x(N-h-2), \dots, x(N-1-h-d)$	$x(N)$

tiempo  $t + h + 1$ , como se muestra en el cuadro 7.2. El aprendizaje de la red se realiza para minimizar el error medido en la salida en dicho instante, es decir:

$$e(t+h+1) = \frac{1}{2} (x(t+h+1) - \tilde{x}(t+h+1))^2 \quad \forall t = d, \dots, N-h-1 \quad (7.10)$$

La formulación de este modelo de predicción tiene sentido siempre que exista una relación, aunque desconocida a priori, entre el valor de la serie en el instante de tiempo  $t + h + 1$  y la secuencia  $(x(t), x(t-1), \dots, x(t-d))$ . Hay que destacar también que a medida que aumenta el horizonte de predicción, las relaciones entre los valores temporales de la serie se pierde y, como consecuencia, no tendrá sentido plantear esquemas de predicción de este tipo.

## 7.4. Predicción del nivel del agua en la laguna de Venecia

Generalmente, el fenómeno de marea alta en el mar resulta de la combinación de elementos estables o periódicos y de elementos climáticos caóticos de los sistemas de marea en un área particular. La predicción de tales fenómenos naturales ha sido siempre de interés, no sólo desde un punto de vista humano, sino también desde un punto de vista económico. Un ejemplo claro del fenómeno de marea alta ocurre en la laguna de Venecia.

En esta sección se va a utilizar como caso práctico de predicción de series temporales el nivel de la marea en la laguna de Venecia. El interés en dicha serie temporal viene dado por el fenómeno conocido como agua alta, que consiste en grandes subidas del nivel de la marea, hasta niveles de alrededor de 110 cm.

La predicción de este fenómeno ha sido abordada con diferentes modelos o aproximaciones. Entre ellos, se encuentran modelos basados en un conjunto de ecuaciones hidrodinámicas, modelos estocásticos lineales como modelos ARMA o modelos basados en análisis no lineal de series temporales o basados en redes de neuronas artificiales [Zaldívar et al., 2000]. A continuación, se van a utilizar los esquemas de predicción presentados en el apartado anterior para predecir el nivel del agua en la laguna de Venecia. Dichos esquemas se van a construir utilizando el PERCEPTRON multicapa. Se pretende abordar tanto la predicción en un paso de tiempo, como en varios pasos de tiempo; concretamente se plantean horizontes de predicción de  $h = 1, 4, 12, 24, 28$ .

### Estructura del modelo NAR

El primer paso a realizar es la determinación de la estructura del modelo NAR, es decir, la estimación del número de muestras de la serie en instantes anteriores de tiempo necesario para predecir su comportamiento. Un estudio del espectro de frecuencia de la serie temporal mostró que existen dos periodicidades, una diurna y otra semidiurna, con un periodo de 12 y 24 horas, respectivamente [Zaldívar et al., 1997]. Esta información se ha utilizado para determinar la estructura de los modelos NAR, considerándose modelos con la siguiente estructura:

$$x(t+h) = F(x(t), x(t-1), \dots, x(t-24)) \quad (7.11)$$

siendo  $h = 1, 4, 12, 24, 28$ . Por tanto, las redes de neuronas entrenadas van a tener 25 neuronas de entrada que reciben los valores de la serie  $(x(t), x(t-1), \dots, x(t-24))$  y una neurona de salida, que representará la predicción en 1, 4, 12, 24 y 28 pasos de tiempo.

### Obtención de los ficheros de entrenamiento y validación

Se dispone de un conjunto de datos que corresponden a medidas en centímetros del nivel del agua en la laguna de Venecia, tomadas cada hora durante la década de los 90. Esto supone una gran cantidad de datos, y la mayor parte de ellos

corresponden a niveles normales del nivel del agua.

Del conjunto total disponible se extraen dos conjuntos de datos -uno para entrenamiento y otro para validación- que contengan muestras que representen tanto situaciones normales como situaciones de marea alta, de manera que la red disponga de ejemplos representativos del problema a resolver. Se obtienen, entonces, dos conjuntos que corresponden a las muestras tomadas durante los meses de noviembre y diciembre de 1990 y durante los meses de noviembre y diciembre de 1991. El primer conjunto se utilizará para entrenar la red y el segundo para validar. Como se observa en la Figura 7.5, ambos contienen muestras que se corresponden con situaciones de marea alta.

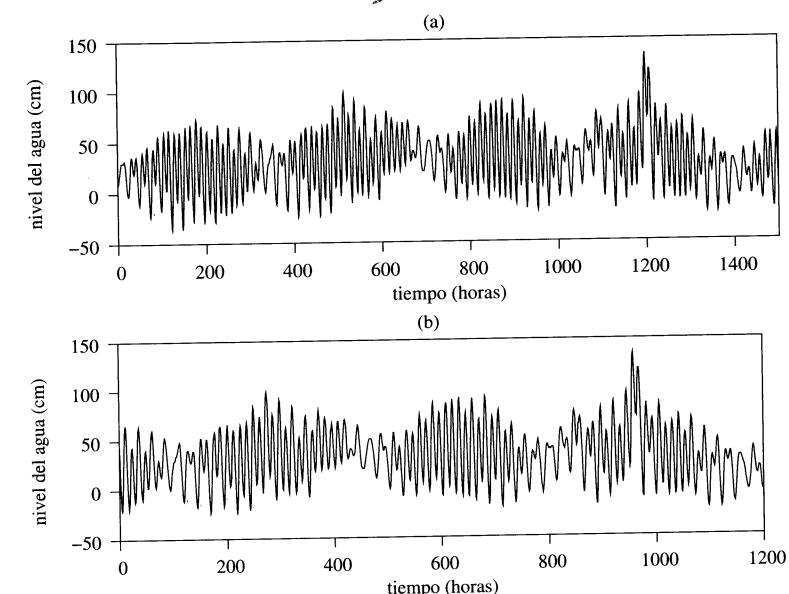


Figura 7.5: Muestras del nivel del agua en la laguna de Venecia. (a) Datos para el entrenamiento: (b) Datos para la validación

Las muestras se normalizan o escalan en el intervalo  $[0, 1]$  y se generan cinco pares diferentes de ficheros para cada horizonte de predicción,  $h = 1, 4, 12, 24, 28$ , según se ha especificado en los cuadros 7.1 y 7.2. Así, el primer par de ficheros será utilizado para entrenar y validar el modelo de predicción en un paso de tiempo (ver Ecuación 7.2), y los restantes pares de ficheros para entrenar y validar los modelos del tipo dado por la Ecuación (7.9) para la predicción en los horizontes  $h = 4, 12, 24, 28$ , respectivamente.

### Entrenamiento de los diferentes modelos de predicción

En primer lugar, se realiza el entrenamiento del modelo de predicción para un paso de tiempo. Dicho modelo se utiliza, posteriormente, para la predicción

Cuadro 7.3: Error cuadrático medio (cm) para los esquemas de predicción

Horizonte	Esquema de predicción 1	Esquema de predicción 2
$h = 1$	3,30	3,30
$h = 4$	9,75	9,55
$h = 12$	12,38	11,38
$h = 24$	13,15	11,64
$h = 28$	16,91	15,74

en múltiples pasos de tiempo, siguiendo el esquema de predicción 1, es decir, retroalimentando la salida hacia la entrada. Se entranan también los modelos para la predicción en  $x(t+4)$ ,  $x(t+12)$ ,  $x(t+24)$  y  $x(t+28)$  siguiendo el esquema de predicción 2.

Todos los modelos son entrenados durante 5000 ciclos de aprendizaje y con razón de aprendizaje  $\alpha = 0,01$ .

### Resultados de predicción

El cuadro 7.3 recoge los errores cuadráticos medios obtenidos sobre el conjunto de validación para la predicción en un paso de tiempo ( $h = 1$ ) y para múltiples pasos de tiempo ( $h = 4, 12, 24, 28$ ) con los dos esquemas de predicción presentados en el apartado anterior.

La Figura 7.6 muestra la predicción en un paso de tiempo realizada por la red de neuronas para una porción del conjunto de validación. Y las Figuras 7.7 y 7.8 muestran las predicciones para 4 y 12 horas, respectivamente, obtenidas con los diferentes esquemas de predicción para el mismo conjunto de muestras.

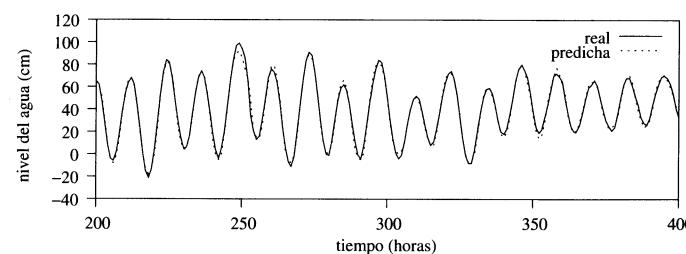


Figura 7.6: Predicción del nivel del agua en la laguna de Venecia con 1 hora de antelación

Se observa en los resultados obtenidos que el PERCEPTRON multicapa es capaz de predecir con bastante exactitud el nivel del agua en la laguna de Venecia con una hora de antelación y que dicha predicción empeora a medida que aumenta el horizonte de predicción, de modo que no se consigue una predicción adecuada con 12, 24 y 28 horas de antelación. En lo referente a los diferentes

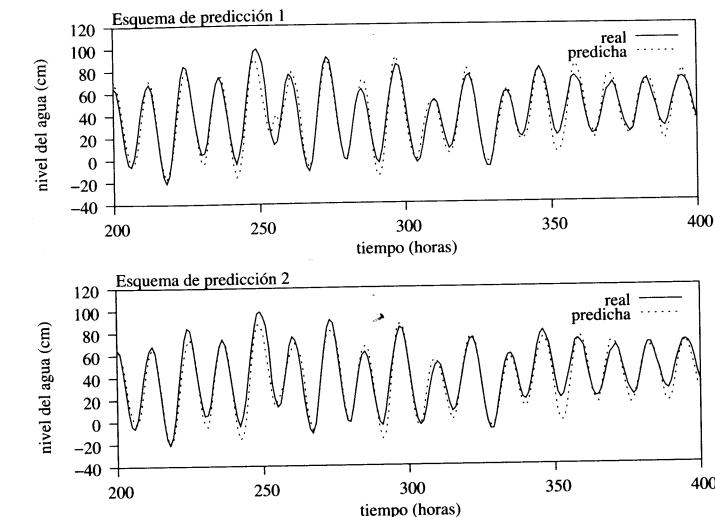


Figura 7.7: Predicción del nivel del agua en la laguna de Venecia con 4 horas de antelación

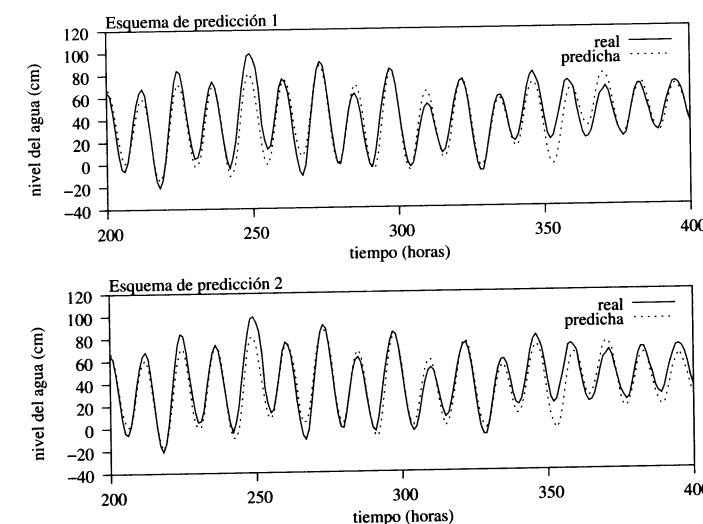


Figura 7.8: Predicción del nivel del agua en la laguna de Venecia con 12 horas de antelación

esquemas de predicción para múltiples pasos de tiempo, los resultados muestran que, en el caso de la serie temporal que se trata, ambos esquemas poseen un comportamiento bastante similar, si bien el segundo esquema es ligeramente superior.

## Capítulo 8

# CONTROL DE PROCESOS DINÁMICOS

## 8.1. Introducción

Un proceso o sistema dinámico puede definirse como una combinación de elementos o componentes relacionados entre sí y que actúan para alcanzar una determinada meta. El estudio de un proceso dinámico es el estudio de la variación a lo largo del tiempo de las variables de salida de dicho proceso, a partir de las variables de entrada que afectan al comportamiento dinámico del proceso.

Desde un punto de vista matemático, un proceso dinámico se suele representar mediante una relación (directa o indirecta) entre las variables de entrada y salida del proceso, es decir, entre las variables medibles y observables. Para la mayor parte de los procesos dinámicos reales, no es fácil conocer estas relaciones o representaciones matemáticas. Generalmente, la información disponible viene dada por un conjunto de datos observados, los cuales requieren una interpretación para obtener una representación matemática del proceso. El problema de determinar dicha representación, también llamada *modelo del proceso*, es conocido como el problema de modelización e identificación de procesos dinámicos.

En el contexto de procesos dinámicos es habitual perseguir que la salida de dicho proceso evolucione en una dirección deseada, como mantener una cierta condición estable o seguir una trayectoria especificada, para conseguir el mejor rendimiento del proceso y una actuación adecuada. Con este propósito, los procesos dinámicos requieren lo que se conoce como *sistemas de control*, es decir, sistemas que mediante la manipulación de ciertas variables de entrada al proceso -aquellas que puedan ser manipuladas-, regulan el comportamiento de las variables de salida para alcanzar el objetivo deseado.

En toda la literatura se hace una distinción entre procesos dinámicos lineales y procesos dinámicos no lineales. Se dice que un proceso es lineal cuando la relación entre las variables que intervienen en el comportamiento de dicho proceso

es lineal. En caso contrario, se dice que el proceso es no lineal. En lo que sigue se tratarán procesos dinámicos no lineales, y aunque existen diferentes técnicas para su tratamiento, es decir, para la modelización y el control, el interés en este capítulo se centra en el uso de métodos basados en redes de neuronas supervisadas. Propiedades como la capacidad de las redes para aprender a partir de ejemplos o capturar relaciones no lineales, han hecho que se hayan utilizado ampliamente en el estudio y análisis de procesos dinámicos. Algunos de los métodos más generales serán presentados, de manera esquemática, en este capítulo.

## 8.2. Modelización y control de procesos dinámicos

La **modelización de un proceso dinámico** implica la construcción de un modelo capaz de representar o simular el comportamiento dinámico del proceso. Para algunos procesos dinámicos, estos modelos pueden obtenerse aplicando las leyes físicas que gobiernan el comportamiento del proceso. La principal ventaja de estos modelos es que los parámetros y variables que intervienen poseen una interpretación física. Sin embargo, en muchas ocasiones estos modelos son complicados de construir, pues puede llegar a ser extremadamente difícil considerar todas las leyes físicas que intervienen en el comportamiento del proceso.

Existen otros tipos de modelos que relacionan directamente las variables de entrada y salida del proceso y cuya construcción se basa en los datos observados. Estos modelos se conocen como modelos NARMA (Nonlinear Auto-Regressive Moving Average) para procesos dinámicos no lineales y su valor radica en que son simples y fáciles de construir y tratar. Por simplicidad se asume que se tratan sistemas con un única variable de entrada y salida; en este caso, los modelos NARMA vienen dados por la siguiente expresión:

$$y(t+1) = F(y(t), \dots, y(t-n_y), u(t), \dots, u(t-n_u)) \quad (8.1)$$

donde  $t$  representa la variable discreta tiempo;  $u$  y  $y$  son las variables de entrada y salida del proceso, respectivamente;  $n_u$  y  $n_y$  son números naturales que indican la longitud de las secuencias discretas en el pasado de dichas variables, respectivamente; y  $F$  es una relación, desconocida a priori, que tendrá que ser estimada mediante el uso de técnicas de aproximación a partir de los datos observados.

Para la construcción de modelos NARMA pueden utilizarse diferentes clases o familias de funciones, como los polinomios, las funciones trigonométricas, los splines, las redes de neuronas artificiales, etc., que tienen la propiedad de ser aproximadores universales. Cada uno de estos métodos tiene sus propias características, que hacen que, en ciertas condiciones, unos sean mejores que otros. Así, por ejemplo, los modelos construidos con polinomios tienen la ventaja de que, aun siendo modelos no lineales, siguen presentando linealidad respecto a los parámetros, lo cual permite el uso de técnicas lineales para la determinación de dichos parámetros. Sin embargo, presentan el inconveniente de que pueden

ser inestables respecto a pequeñas perturbaciones de sus coeficientes. Los modelos basados en redes de neuronas artificiales suelen ser más tolerantes al ruido, aunque la determinación de los parámetros o pesos de la red se lleva a cabo mediante el uso de técnicas no lineales de optimización.

El **control de un proceso dinámico** consiste en regular el comportamiento del proceso, regulación que se lleva a cabo manipulando la variable de entrada al proceso, también referida en este contexto como señal o acción de control. De este modo, un sistema de control se encarga de proporcionar la acción que hay que aplicar al proceso para que la salida de dicho proceso tienda al valor deseado, también llamado objetivo de control.<sup>1</sup>

Si  $y$  representa la variable de salida,  $u$  la variable de entrada o acción de control y  $y^{des}$  el objetivo de control, el control del proceso consiste en determinar la acción  $u(t)$  tal que  $\|y^{des} - y(t+1)\| \approx 0$ , para todo instante de tiempo  $t$ .

El control de procesos dinámicos se puede llevar a cabo utilizando diferentes técnicas, como son, las técnicas de control óptimo [Brogan, 1985], control adaptativo [Goodwin and Sin, 1984], control predictivo ([Clarke et al., 1987a], [Clarke et al., 1987b]) o los clásicos controladores PID [Brogan, 1985]. Estas técnicas de control fueron principalmente desarrolladas para procesos lineales, aunque ya en la actualidad aparece en la literatura adaptaciones para procesos dinámicos no lineales [Isidori, 1989], [White and Sofge, 1992], [Miller et al., 1991], en las cuales han intervenido las redes de neuronas artificiales.

## 8.3. Diferentes esquemas de control utilizando redes de neuronas

Las redes de neuronas artificiales ocupan un lugar importante en el desarrollo de técnicas de control para procesos dinámicos no lineales. Cuando se habla de control de procesos utilizando redes de neuronas, generalmente, se entiende que es una red la encargada de calcular la acción de control que hay que aplicar al proceso para que se alcance el objetivo de control deseado. La diversidad de los métodos o estrategias de control radica en el modo de realizar el entrenamiento de la red, así como en el tipo de red de neuronas a utilizar.

Cuando se utilizan redes de neuronas supervisadas para el control de procesos dinámicos es necesario disponer de una respuesta deseada para realizar el entrenamiento de la red. Los algoritmos de aprendizaje se basan, generalmente, en el método de descenso del gradiente, de manera que los pesos o parámetros de la red que actúa como controlador se adaptan siguiendo la dirección negativa del gradiente de una determinada función error.

A continuación, se presentan diferentes esquemas de sistemas de control basados en redes supervisadas, los cuales difieren en la función error a minimizar durante el entrenamiento de la red. Para la construcción de estos esquemas

<sup>1</sup> Nótese que se ha asumido, por simplicidad, que el proceso dinámico tiene una única variable de entrada. En el caso de poseer varias variables de entrada, el control se realiza a través de aquellas entradas que puedan ser manipuladas.

pueden utilizarse tanto redes con conexiones hacia adelante, como redes recurrentes, aunque las más utilizadas han sido el PERCEPTRON multicapa y las redes parcialmente recurrentes.

### 1. Copia de un controlador ya existente

Uno de los métodos más simples consiste, básicamente, en copiar un controlador ya existente. En este caso, el entrenamiento se realiza para que la red aprenda el comportamiento de un controlador, utilizando como salida deseada para la red, la salida de dicho controlador. Este esquema puede parecer carente de sentido, pues el controlador ya está disponible. Sin embargo, puede ocurrir que por razones prácticas, como un alto coste computacional, el controlador no pueda actuar en tiempo real, en cuyo caso es útil disponer de una estructura equivalente.

### 2. Control inverso

El esquema de control inverso consiste en aproximar mediante una red de neuronas la dinámica inversa del proceso. Suponiendo que la salida de un proceso dinámico se puede expresar de la forma  $y(t+1) = S(u(t))$ , siendo  $u$  la entrada al proceso, la relación inversa en una expresión de la forma  $u(t) = S^{-1}(y(t+1))$ . Por tanto, en una estrategia de control inverso la red de neuronas se utiliza para aproximar la relación  $S^{-1}$ .

Cuando se plantea un esquema de control inverso utilizando redes de neuronas, el aprendizaje de la red puede llevarse a cabo mediante dos formas diferentes, denominadas aprendizaje generalizado y aprendizaje especializado, las cuales se describen a continuación.

#### ■ Aprendizaje generalizado

En este caso, la red de neuronas se entrena para que aprenda la dinámica inversa del proceso en su totalidad. Para ello, se utiliza un conjunto de datos representativo de dicha dinámica inversa, los cuales se obtienen manipulando la acción de control en sus diferentes rangos de operación y observando o midiendo la salida del proceso para dicha señal.

En la Figura 8.1 se muestra el esquema general del aprendizaje generalizado. Se observa que la entrada al proceso actúa como salida deseada para la red, mientras que la salida del proceso es la entrada a la red. Por tanto, el aprendizaje de la red se realiza siguiendo la dirección negativa del gradiente de la siguiente función error:

$$e_g(t) = \frac{1}{2} (\tilde{u}(t) - u(t))^2 \quad (8.2)$$

Una vez realizado el entrenamiento de la red, ésta puede utilizarse para el control; basta sustituir la entrada a la red  $y(t+1)$  por el objetivo de control,  $y^{des}$ , como se muestra en la Figura 8.2.

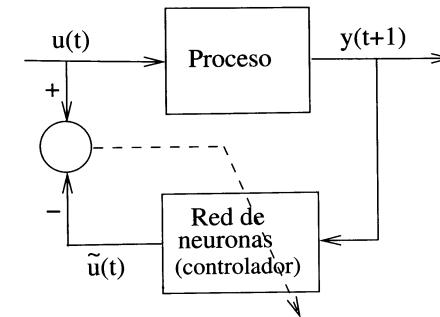


Figura 8.1: Esquema de control inverso con aprendizaje generalizado

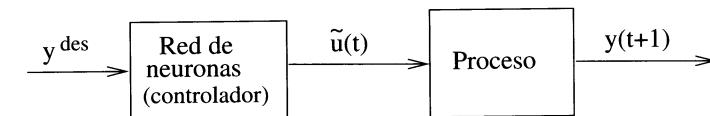


Figura 8.2: Red de neuronas actuando como controlador

#### ■ Aprendizaje especializado

Mediante el aprendizaje especializado, la red approxima la dinámica inversa local del proceso, es decir, la dinámica inversa en una determinada región de interés y no en todo el rango de operación de la acción de control.

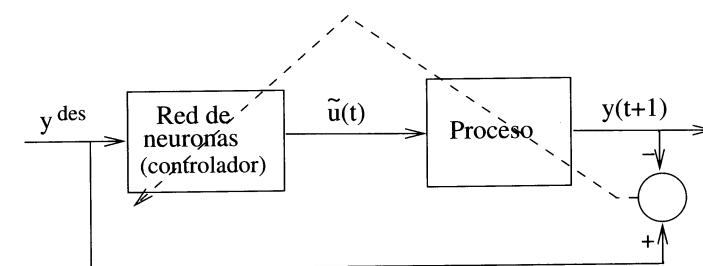


Figura 8.3: Esquema de control inverso con aprendizaje especializado

Como se muestra en la Figura 8.3, la entrada a la red es el objetivo de control y la salida de la red es la acción de control que se aplica al proceso dinámico. El aprendizaje de la red se realiza para minimizar la diferencia entre la salida del proceso y el objetivo de control deseado, es decir:

$$e_s(t+1) = \frac{1}{2} (y^{des} - y(t+1))^2 \quad (8.3)$$

Para poder realizar el aprendizaje de la red siguiendo la dirección negativa del gradiente del error dado por la Ecuación 8.3, es necesario conocer el Jacobiano del proceso, es decir, la variación de la salida del proceso respecto de la entrada,  $\frac{\partial y}{\partial u}$ . Esto es debido a que el proceso se coloca entre la red que actúa como controlador y la función error a minimizar (ver Figura 8.3). En el caso de que dicho Jacobiano sea desconocido, es necesario utilizar una aproximación. Dicha aproximación se puede obtener, bien utilizando un modelo del proceso, cuya estructura es conocida, y por tanto es posible derivar la salida respecto de la entrada, o bien approximando dicha derivada mediante una ecuación en diferencia con dos puntos, es decir:

$$\frac{\partial y}{\partial u} \approx \frac{y(u+h) - y(u)}{h} \quad (8.4)$$

Por tanto, si  $w$  es un parámetro de la red, la ley para su adaptación siguiendo la dirección negativa del gradiente de la función error dada por la Ecuación (8.3) es:

$$w(t+1) = w(t) + \alpha(y^{des} - y(t+1)) \frac{\partial y}{\partial u} \frac{\partial \tilde{u}}{\partial w} \quad (8.5)$$

Como  $\tilde{u}$  es la salida de la red,  $\frac{\partial \tilde{u}}{\partial w}$  se calcula utilizando el algoritmo de aprendizaje asociado al tipo de red que actúa como controlador.

Cuando se utiliza una estrategia de aprendizaje especializado no es necesario disponer de un conjunto de patrones de entrenamiento, pues los datos para el aprendizaje de la red proceden de la evolución directa del proceso. Se trata, por tanto, de una estrategia de control en tiempo real y el aprendizaje se lleva a cabo mientras la red actúa como controlador del proceso. Es conveniente, sin embargo, inicializar la red con parámetros o pesos que posean cierta información sobre la dinámica inversa del proceso, y no de pesos aleatorios, pues esto podría provocar un comportamiento no adecuado del controlador. Dicha inicialización se puede obtener realizando un aprendizaje especializado previo con un modelo del proceso, en lugar de utilizar directamente el proceso real.

Se ha comentado en la sección anterior que, generalmente, la salida del proceso en un instante de tiempo  $t+1$  no sólo depende de la entrada en el instante anterior  $t$ , sino que puede también depender de los valores de la entrada y la salida en un conjunto de instantes anteriores de tiempo (ver Ecuación 8.1). En estos casos, cuando se trata de aproximar la dinámica inversa del proceso la situación es similar. Por tanto, es necesario señalar que la red que actúa como controlador inverso no sólo recibe como entrada

la salida del proceso en un cierto instante de tiempo, sino que será necesario utilizar también como entrada a la red una secuencia finita de valores en instantes anteriores de tiempo, como se verá en la Sección 8.4.

### 3. Control predictivo

Las estrategias de control predictivo se basan en la idea de que la acción de control aplicada al proceso en un determinado instante de tiempo podría influir en la evolución en el futuro. Desde un punto de vista general, estas estrategias de control se caracterizan porque utilizan el comportamiento dinámico del proceso en el futuro para determinar la acción de control en el instante actual. Para predecir dicho comportamiento futuro y poder calcular la acción de control actual se utiliza un modelo del proceso. Las estrategias de control predictivo han sido desarrolladas y estudiadas, principalmente, para procesos dinámicos lineales, aunque es posible aplicarlas también a procesos no lineales.

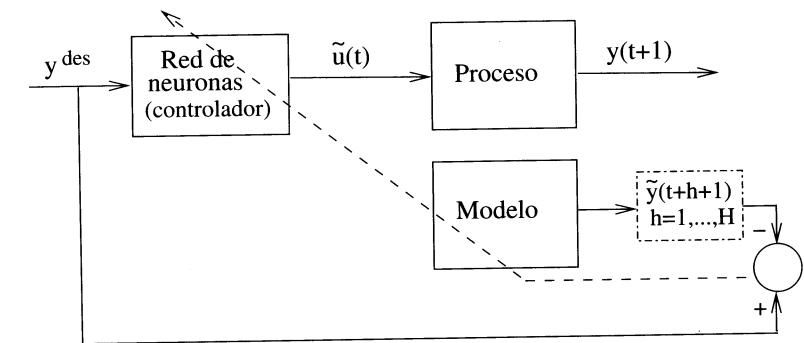


Figura 8.4: Esquema de control predictivo

En el contexto de control mediante redes de neuronas es posible plantear una estrategia de control predictivo [Galván, 1997], lo que da lugar al esquema de control que se muestra en la Figura 8.4. El esquema que resulta es muy similar al esquema de control inverso con aprendizaje especializado (ver Figura 8.3), pues la red de neuronas recibe como entrada el objetivo de control y su salida es la acción de control que hay que aplicar al proceso dinámico. Sin embargo, en este caso, y es lo que lo diferencia de una estrategia de control inverso, el aprendizaje de la red se realiza para minimizar las diferencias entre el objetivo de control y la salida de un modelo del proceso en el futuro, es decir:

$$e_p(t+1) = \frac{1}{2} \sum_{h=0}^H (y^{des} - \tilde{y}(t+h+1))^2 \quad (8.6)$$

De este modo, la acción de control no sólo se calcula a partir de la respuesta

del proceso en el siguiente instante de tiempo, sino también a partir del proceso en un futuro más lejano, dado por el horizonte de predicción  $H$ .

En este caso, el modelo se interpone entre la red de neuronas que actúa como controlador y la función error a minimizar (ver Figura 8.4). Por tanto, para realizar el aprendizaje de la red siguiendo la dirección negativa del gradiente del error dado por la Ecuación (8.6) es necesario calcular la variación de la salida del modelo a lo largo del horizonte de predicción respecto de la acción de control, es decir  $\left\{ \frac{\partial \tilde{y}(t+h+1)}{\partial u} \right\}_{h=0,1,\dots,H}$ .

La ley para modificar los parámetros de la red viene dada por:

$$w(t+1) = w(t) + \alpha \frac{\partial \tilde{u}}{\partial w} \sum_{h=0}^H (y^{des} - \tilde{y}(t+h+1)) \frac{\partial \tilde{y}(t+h+1)}{\partial u} \quad (8.7)$$

#### 8.4. Control de la temperatura en un reactor químico

Como caso práctico de aplicación de redes de neuronas supervisadas al control de procesos dinámicos se va a utilizar un ejemplo de proceso en el campo de la ingeniería química. Existe una clase de reactores químicos, conocidos como reactores batch, que se caracterizan porque permiten la producción de productos químicos en pequeñas cantidades, lo cual es interesante desde el punto de vista de la investigación de procesos químicos, y permiten cambiar de reacción tan sólo con pequeñas modificaciones en la instalación. Esto ha producido un incremento del uso de este tipo de reactores. Sin embargo, estos procesos suelen ser muy complejos, con sistemas de reacción que normalmente no son conocidos en su totalidad, y con componentes altamente no lineales.

En esta sección se van a describir los pasos para abordar el control de la temperatura del fluido para el intercambio de calor que circula en la camisa de un reactor químico batch utilizando los esquemas de control inverso y predictivo presentados en la Sección 8.3. La temperatura de este fluido influye en la dinámica de la temperatura del reactor y, por tanto, en la seguridad de éste. Dicho reactor fue estudiado dentro del marco del proyecto de investigación FIRE (Facility for Investigating Runaway Events Safely) ([Hernández and Zaldívar, 1990], [Zaldívar et al., 1993]).

No es de interés en este punto conocer las características específicas ni las propiedades del reactor químico, y sólo se hará mención de aquellas que se necesiten para la construcción de los sistemas de control, aunque el lector interesado puede encontrar una descripción detallada del reactor [Galván and Zaldívar, 1997].

El primer paso en la construcción de sistemas para controlar la temperatura del fluido utilizando redes de neuronas es conocer las variables que intervienen en la dinámica de dicho proceso. Éstas son: la temperatura ambiente,  $T_a$ , la temperatura de un depósito que contiene líquido frío,  $T_c$ , la temperatura del

reactor,  $T_m$ , y una señal digital,  $u$ , que actúa para enfriar o calentar el fluido. De todas las variables de entrada al proceso, sólo la señal  $u$  puede ser manipulada, y el resto son salidas de otros procesos dinámicos o simplemente variables del entorno. Por tanto, el control de la temperatura del fluido se realizará mediante la señal  $u$ , que será la salida de la red de neuronas que actuará como controlador.

Según el diseño del circuito de enfriamiento y calentamiento del reactor, un valor de la señal  $u$  entre el 11 y el 100 % (correspondientes a valores entre 450 y 4095, respectivamente) producirá un enfriamiento en el fluido, siendo el 100 % el máximo enfriamiento. Para ello, se producirá la apertura de una válvula, con el correspondiente porcentaje, del depósito con líquido frío. Valores de la señal  $u$  entre 0 y el 10 % (equivalentes a valores entre 0 y 450, respectivamente) producen el modo de calentamiento, correspondiendo el 0 % al máximo calentamiento. Esto se realiza mediante la aplicación directa de una potencia generada por una resistencia eléctrica.

#### Modelo del proceso

Para el diseño y construcción de los controladores basados en redes de neuronas, es necesario disponer de un modelo del proceso, ya que, tanto el esquema de control inverso con aprendizaje especializado, como el esquema de control predictivo, hacen uso de dicho modelo. Es posible construir un modelo de tipo NARMA (Ecuación 8.1) para modelizar la dinámica de la temperatura del fluido, el cual, y teniendo en cuenta las variables de entrada al proceso, vendría dado por la siguiente expresión:

$$T_e(t+1) = F(T_e(t), \dots, T_e(t-n_e), T_c(t), \dots, T_c(t-n_c), T_a(t), \dots, T_c(t-n_c), \\ T_m(t), \dots, T_m(t-n_m), u(t), \dots, u(t-n_u)) \quad (8.8)$$

donde  $T_e$  representa la temperatura del fluido en la camisa del reactor.

A partir del conocimiento empírico sobre el proceso se puede elegir  $n_c = n_a = 0$ , puesto que, tanto la temperatura del depósito de frío, como la temperatura ambiente, permanecen prácticamente constantes durante la realización de un experimento o reacción química. Respecto a la longitud de las secuencias para las variables  $T_e$ ,  $T_m$  y  $u$ , la experiencia del experto proporciona valores aproximados:  $0 \leq n_e \leq 2$ ,  $0 \leq n_m \leq 1$  y  $0 \leq n_u \leq 20$ . Eligiendo entonces  $n_e = 2$ ,  $n_m = 1$  y  $n_u = 20$ , se puede construir un modelo NARMA que explique la dinámica del proceso:

$$T_e(t+1) = F(T_e(t), T_e(t-1), T_e(t-2), T_c(t), T_a(t), T_m(t), T_m(t-1), \\ u(t), \dots, u(t-20)) \quad (8.9)$$

Para la aproximación del funcional  $F$  es necesario disponer de un conjunto de datos experimentales, el cual se obtiene manipulando directamente la señal  $u$  y midiendo el resto de las variables observables cada cierto intervalo de muestreo (10 segundos). Para que el conjunto de datos sea representativo de la dinámica del proceso, la señal  $u$  debe cubrir todo su rango de operación, [0, 4095].

La aproximación de  $F$  puede realizarse utilizando el PERCEPTRON multicapa; basta considerar como entrada a la red el siguiente vector:

$$(T_e(t), T_e(t-1), T_e(t-2), T_c(t), T_a(t), T_m(t), T_m(t-1), u(t), \dots, u(t-20))$$

Para reducir el número de variables de entrada podrían utilizarse técnicas de análisis de sensibilidad. En [Galván and Zaldívar, 1997] se obtiene un modelo NARMA simplificado capaz de simular el comportamiento de la dinámica de la temperatura del fluido de calentamiento/enfriamiento. Dicho modelo se construye utilizando una arquitectura particular de red parcialmente recurrente, la cual proporciona, en este caso, mejores resultados que las redes con conexiones hacia adelante (PERCEPTRON multicapa).

### Estructura del controlador

En este contexto, una red de neuronas va a ser la encargada de proporcionar la acción  $u$  -porcentaje de calentamiento o enfriamiento- que hay que aplicar al proceso para que la temperatura del fluido tienda hacia el objetivo de control deseado. En el caso de estudio, el objetivo de control se suele traducir en conseguir que la temperatura del fluido alcance un valor deseado,  $T_e^{des}$ , en el menor tiempo posible, y se establezca cuando alcance dicho valor.

Antes de pasar al aprendizaje de la red para el control, es necesario definir la estructura del controlador, es decir, las variables que serán utilizadas como entrada a la red de neuronas. Suponiendo que la dinámica del proceso puede ser explicada mediante el modelo NARMA dado por la Ecuación (8.9), la señal de control en un instante de tiempo  $t$  puede expresarse de la siguiente forma:

$$u(t) = G(T_e(t+1), T_e(t), T_e(t-1), T_e(t-2), T_c(t), T_a(t), T_m(t), T_m(t-1), \\ u(t-1), \dots, u(t-20)) \quad (8.10)$$

donde  $G$  relaciona la acción de control actual y el resto de las variables que intervienen en el proceso. Si dicha relación  $G$  fuera conocida, sería posible calcular la acción de control que hay que aplicar al proceso en un determinado instante de tiempo  $t$  para que la temperatura del fluido en el instante  $t+1$  tome el valor  $T_e(t+1)$ . De este modo, si el objetivo de control es que la temperatura alcance el valor  $T_e^{des}$ , la acción de control vendrá dada por:

$$u(t) = G(T_e^{des}, T_e(t), T_e(t-1), T_e(t-2), T_c(t), T_a(t), T_m(t), T_m(t-1), \\ u(t-1), \dots, u(t-20)) \quad (8.11)$$

Si se utiliza el PERCEPTRON multicapa para aproximar la relación  $G$ , cuando se sigue una estrategia de control inverso con aprendizaje generalizado la red de neuronas va a tener como variables de entrada el vector  $(T_e(t+1), T_e(t), T_e(t-1), T_e(t-2), T_c(t), T_a(t), T_m(t), T_m(t-1), u(t-1), \dots, u(t-20))$ . Cuando se realice una estrategia de control inverso o predictivo, estrategias en las que se

utiliza el objetivo de control deseado, el vector de entrada a la red está dado por  $(T_e^{des}, T_e(t), T_e(t-1), T_e(t-2), T_c(t), T_a(t), T_m(t), T_m(t-1), u(t-1), \dots, u(t-20))$ .

Se observa que, en ambos casos, la red posee un alto número de variables de entrada, por lo que de nuevo puede ser interesante aplicar técnicas de análisis de sensibilidad para reducir el número de variables. En este caso, es posible también partir de un modelo NARMA más simple, que aunque no explique la dinámica del proceso en su totalidad, sí sea capaz de capturar las variaciones más importantes de la dinámica del proceso.

### Aprendizaje del controlador neuronal

El aprendizaje del controlador consiste en aproximar la relación  $G$ , y dicha aproximación se llevará a cabo para minimizar un cierto error, dependiendo de la estrategia de control utilizada, como se ha descrito en la Sección 8.3.

- **Control inverso generalizado.** En este caso, la red será entrenada para minimizar el error dado por la Ecuación (8.2) utilizando el algoritmo de RETROPROPAGACIÓN estudiado en el capítulo 3. Para el entrenamiento de la red, los datos se obtienen manipulando la señal  $u$  en todo su rango de operación y midiendo el valor de las variables observables ( $T_e, T_c, T_a, T_m$ ). La acción de control  $u$  actuará como salida deseada para la red.
- **Control inverso especializado.** El aprendizaje de la red se lleva a cabo en tiempo real para minimizar el error dado por la Ecuación (8.3). Los datos para el aprendizaje proceden de la evolución del proceso y los pesos de la red se modifican siguiendo la ley dada por la Ecuación (8.5). En este caso, y al tratarse de un aprendizaje en tiempo real, es conveniente partir de una red que contenga en sus pesos cierta información sobre el proceso. Para ello, y como se ha comentado en la Sección 8.3, se puede realizar un aprendizaje previo utilizando un modelo del proceso, en lugar del proceso real, o bien partir de los pesos obtenidos después del aprendizaje generalizado.
- **Control predictivo.** Se define un horizonte de predicción  $H$  y se realiza el aprendizaje de la red siguiendo la ley dada por la Ecuación (8.7) para minimizar las diferencias entre las salidas del modelo y el objetivo de control deseado,  $T_e^{des}$ , a lo largo del horizonte de predicción (Ecuación 8.6).

En [Galván and Zaldívar, 1998] se muestran resultados de control de la temperatura del fluido que circula en la camisa del reactor químico para diferentes objetivos de control. Estos resultados fueron obtenidos utilizando el proceso real y con los esquemas de control descritos en este capítulo, los cuales fueron construidos con una red parcialmente recurrente.

# Capítulo 9

## CLASIFICACIÓN

### 9.1. Tarea de clasificación

Una de las tareas más comunes en que las Redes de Neuronas Artificiales han demostrado su eficacia es la tarea de **clasificación automática**. En vida cotidiana existen muchos ejemplos de tareas de clasificación. Se puede decir que nos pasamos gran parte de nuestra vida haciendo clasificación. Desde un punto de vista más elemental, los seres vivos necesitan categorizar el mundo que los rodea, para poder reaccionar de forma eficaz a estímulos desconocidos. Los animales distinguen, por ejemplo, entre depredadores y presas, entre machos y hembras, entre enemigos y colaboradores, entre familiares y desconocidos, y otras muchas clasificaciones fundamentales para su comportamiento diario. En el ser humano la categorización es aún más sofisticada y más abundante; es parte fundamental de todas las culturas. Si se desea realizar sistemas automáticos capaces de “imitar” ciertos comportamientos de seres vivos, será inevitable incluir tareas de clasificación automática. Por ejemplo, los sistemas de visión artificial tienen una parte importante de reconocimiento de la señal visual, lo cual puede ser entendido desde un punto de vista de clasificación de imágenes. Los sistemas de reconocimiento de caracteres, los sistemas de reconocimiento del habla, los sistemas de navegación de robots, los sistemas de conducción automática de vehículos, todos ellos incorporan tareas de clasificación. Para comprender el habla será necesario clasificar los diferentes fonemas y distinguirlos; para conducir un vehículo será imprescindible distinguir la calzada del arcén, o de lo que no es carretera, distinguir las señales de tráfico, y esto para todos los ejemplos mencionados y otros muchos que empiezan a ser objeto de estudio en laboratorios de Inteligencia Artificial.

Desde un punto de vista más pragmático existen muchas aplicaciones que comprenden tareas de clasificación y que son realizadas por operarios poco especializados.

lizados. Se van a citar varios ejemplos de aplicaciones reales, en las que se han utilizado Redes de Neuronas Artificiales en tareas de clasificación:

- Examen de radiografías. Para examinar las radiografías de pacientes es necesario un médico especialista capaz de detectar problemas en las mismas. Normalmente el análisis de la radiografía se realiza como parte de un conjunto de pruebas en las que el médico tiene indicios de la dolencia, y necesita corroborar o descartar la conjectura con la que trabaja. Se sabe qué es lo que se está buscando, y en muchos casos es fácil de detectar en la radiografía. También se utilizan en clínicas veterinarias. Un ejemplo de este tipo son las pruebas realizadas con una raza de perros, el mastín leonés. En estos perros una señal de la pureza de la raza la indica la forma de la cabeza del fémur. Se han realizado sistemas de clasificación de dicha pureza, a partir de las radiografías del fémur, utilizando Redes de Neuronas Artificiales.
- Clasificación de jamones. En algunos casos las aplicaciones son bastante curiosas. Es el caso del sistema de clasificación de jamones. El problema en este caso consiste en determinar el grado de grasa de un jamón, a partir de una fotografía de éste. Esto permitirá clasificar automáticamente los jamones en una fábrica, y asignarles calidades según la grasa que posean.
- Clasificación de documentos. En ocasiones existe la necesidad de clasificar documentos como pertenecientes a un determinado tema. Por ejemplo, si alguien quiere hacer un análisis sobre un acontecimiento reciente, pongamos un asesinato, puede serle de mucha utilidad disponer de todos los artículos publicados sobre el tema en los distintos periódicos. Una Red de Neuronas Artificial puede ser capaz de realizar esta tarea de forma automática, al clasificar los artículos en temas.
- Etiquetado de libros. En las bibliotecas es necesario introducir una referencia en cada uno de los libros del depósito. Esta referencia es de gran importancia para almacenar los libros, y posteriormente facilitar las consultas de los usuarios. Un mal etiquetado puede hacer que el libro esté infrautilizado y que el usuario se vea afectado. Existen sistemas neuronales capaces de realizar un buen etiquetado a partir de la clasificación del libro en temas.

Éstos son sólo un conjunto de ejemplos reales de problemas de clasificación que pueden ser resueltos mediante Redes de Neuronas. El gran número de aplicaciones posibles de esta tarea hace que su estudio requiera un apartado especial. En este capítulo se hablará en general de tareas de clasificación automática y de algunas técnicas que producen soluciones eficientes.

### 9.1.1. Características

En este capítulo se trata el tema del diseño de sistemas de clasificación automática a partir de ejemplos. En este caso se dispone siempre de un conjunto

de ejemplos significativo que será utilizado para obtener una serie de reglas generales de clasificación, o bien un conjunto de características de cada una de las clases, o simplemente un sistema al que presentar nuevos ejemplos para su clasificación.

Desde este punto de vista, la tarea de clasificación automática puede dividirse en:

- Clasificación. A partir del conocimiento de la existencia de un conjunto de clases, determinar la regla, o el conjunto de reglas, que permita asignar, a cada nueva observación, a la clase a la que pertenece. Es este caso el tipo de aprendizaje a partir de ejemplos que ha de utilizarse es el **aprendizaje supervisado**, ya que se dispone de la información sobre la clase para cada ejemplo.
- Agrupamiento. A partir de una serie de observaciones, determinar si existen clases en las que dichas observaciones puedan ser agrupadas. En este caso se trata de **aprendizaje no supervisado**, ya que no existen categorías "a priori", y se desconoce a qué clase pueden pertenecer los ejemplos. Hay que determinar cuántas categorías puede haber para los ejemplos, a cuál de las categorías recién generadas pertenecen y las reglas que distinguen unas categorías de otras.

La clasificación automática es una tarea que tiene grandes ventajas. En primer lugar es mucho más rápida, y esta propiedad puede ser muy importante en muchos casos, bien porque el tiempo de respuesta sea crítico -en un caso médico puede ser necesario clasificar los datos existentes para la realización de un buen diagnóstico en el menor tiempo posible-, o bien porque sólo así resulte útil la clasificación -en un clasificador de piezas en un proceso de fabricación, la clasificación ha de realizarse al ritmo en que las piezas pasan por la cinta para no tener que detener todo el proceso-. Los sistemas automáticos tienen la ventaja de que toman decisiones basándose únicamente en la información existente, pasando por alto datos irrelevantes. Por ejemplo, un sistema de clasificación de clientes solicitantes de créditos bancarios tendrá en cuenta la información que se le aporte, pero no se dejará influir por otro tipo de factores, subjetivos o no, que no son de importancia en la tarea en cuestión.

Además, la tarea de clasificación se puede extender a otras muchas tareas de ámbito general, por ejemplo, a tareas de predicción. Así, si se desea realizar predicción de bolsa, se pueden dividir los posibles resultados futuros en clases: estable, baja moderada, baja intensa, alza moderada, alza intensa, etc. Si se tiene un sistema capaz de clasificar una situación, por ejemplo la situación de la bolsa hoy, en función de la previsión para el día siguiente, se podrá realizar la predicción correspondiente. Si el sistema clasifica la situación como de alza moderada, eso es lo que se esperará para el día siguiente, y se podrá actuar en consecuencia (vender unas pocas acciones, por ejemplo). La tarea de clasificación será adecuada si la predicción que subyace es correcta.

Uno de los parámetros más importantes en toda tarea de clasificación es la calidad. Este parámetro se utilizará en muchos casos para guiar el proceso

de aprendizaje, y en todos ellos definirá la bondad del resultado obtenido. Sin embargo es difícil encontrar una buena medida de calidad para un clasificador. En un primer lugar podría definirse como la capacidad de acierto del clasificador. Esto se calcula sencillamente mediante el número de veces que acierta del total de pruebas realizadas, y podría darse en tanto por ciento. Sin embargo, no todos los errores son iguales; hay errores más importantes que otros. Por ejemplo, si se trata de un sistema para reconocer si un enfermo padece una enfermedad altamente contagiosa, si el error cometido es haber dicho que el paciente tenía la enfermedad, y estaba sano (a esto se lo denomina falso positivo), el inconveniente es haber tenido al paciente aislado y en observación, hasta que seguramente se detecte el error; esto se traduce en la utilización de unos medios de forma innecesaria, y la consiguiente pérdida de dinero, o falta de atención a un paciente más necesitado. En cambio, si el error es al revés, es decir que se ha clasificado al paciente como sano, estando enfermo (falso negativo), esto podría hacer que la enfermedad se propagara y se convirtiera en una epidemia de graves consecuencias. Esto ilustra la necesidad de ponderar cada uno de los errores que pueda cometer un sistema en función de su importancia objetiva, y estimar la calidad del clasificador de esta manera. En el caso anterior el objetivo es minimizar los falsos negativos, independientemente de cuántos falsos positivos cometa el sistema, y sólo en caso de misma eficacia en los falsos negativos, pasar a minimizar los falsos positivos.

Encontrar una ponderación justa de los errores no es sencillo, pero no es el único problema a la hora de definir la calidad del clasificador. Si la clasificación es supervisada, el objetivo es minimizar el número de fallos. Para ello se define un conjunto de validación y se miden los fallos cometidos por el sistema sobre dicho conjunto. Recordemos que, en aprendizaje no supervisado, ni siquiera es posible disponer de este conjunto de validación, pero esto se verá más adelante. El problema de valorar la calidad de un clasificador de este tipo (supervisado) es que es necesario saber a priori el número de prototipos que se van a utilizar en la solución. Aunque haya solamente dos clases, por ejemplo, no se puede saber si todos sus elementos están agrupados en dos espacios disjuntos; puede que formen cuatro agrupaciones, dos de cada clase, o en general un número arbitrario de ellas. En los sistemas de neuronas, es necesario saber cuántas agrupaciones existen, ya que hay que incluir una célula por cada agrupación. Como se desconoce totalmente la estructura de los datos, es necesario utilizar el método de prueba y error. Se han de realizar experimentos con diferente número de prototipos. Es aquí donde surge la dificultad de valorar los resultados. Por regla general, el número de errores disminuye al aumentar el número de prototipos, pero el poder de generalización del sistema disminuye. Si se tiene un sistema clasificador con pocos prototipos pero un error alto, y otro con menor error pero con muchos prototipos, la comparación entre ambos, en cuanto a calidad, es difícil de realizar, pues no existe ningún criterio objetivo para decidir que uno es mejor que otro.

En los sistemas no supervisados, esto se complica aún más ya que no existe medida de error. El sistema realiza agrupamientos de los datos y les asigna categorías, pero ¿cómo se puede saber si las categorías obtenidas son buenas?

La medida comúnmente utilizada es la varianza de las agrupaciones obtenidas. Si esta varianza es baja, los agrupamientos serán más compactos y, teóricamente, el resultado del agrupamiento mejor. El problema aquí es también el número de prototipos, ya que la varianza se puede disminuir hasta el valor que se desee sin más que aumentar el número de prototipos. Hay que utilizar otras medidas que combinen la varianza con la densidad de puntos de las agrupaciones. La clasificación será mejor si se disminuye la varianza, aumentando el número de puntos de los agrupamientos. El problema radica en que ambos factores suelen ir inversamente correlacionados: al aumentar la densidad se disminuye la varianza, y viceversa. Determinar si una clasificación es mejor que otra es, pues, difícil, y vuelve a no haber una medida objetiva.

### 9.1.2. Tipos de clasificadores

Existen varios tipos de clasificaciones posibles para los distintos sistemas de clasificación automática existentes. Aquí se mencionará una posible división en tres bloques: los discriminantes lineales o no lineales, los métodos de estimación de densidades y los métodos basados en reglas. A continuación se describe cada uno de ellos, incluyendo un ejemplo de un método significativo.

#### Discriminantes lineales

Los discriminantes lineales consisten en dividir el espacio de estados, es decir, el conjunto de todos los posibles puntos en los que pueden ser ubicados los ejemplos<sup>1</sup>, en regiones, definidas mediante ecuaciones lineales. En un espacio  $n$ -dimensional las fronteras de las regiones estarán determinadas por las intersecciones de los hiperplanos descritos en las ecuaciones que se obtienen como solución al problema. Estas regiones han de tener la particularidad de contener ejemplos únicamente de una clase.

El discriminante lineal más conocido y más utilizado por su sencillez y rapidez es el discriminante de Fisher. Este procedimiento consiste en dividir el espacio en hiperplanos, como se ha dicho anteriormente. Se parte de un conjunto de datos, que se representarán en el espacio como puntos, de los cuales se conoce la clase a la que pertenecen. Cada conjunto de puntos pertenecientes a la misma clase forman un agrupamiento. Se determina en primer lugar el centroide de cada agrupamiento, y a partir de los centroides se determina un nuevo hiperplano de forma que divida el espacio en dos partes biseccionando la línea que une los centros de cada clase.

Para ilustrar estas ideas se incluirá un ejemplo sobre diferentes variedades de la flor del lirio. Existen tres variedades del lirio: Setosa (etiquetada con una S), Versicolor (etiquetada con una E) y Virginica (etiquetada por una A), que pueden distinguirse por el ancho y la longitud de sus pétalos. Además, se dispone del conjunto de muestras ya contrastadas que aparece en la Figura 9.1.

<sup>1</sup>Por ejemplo, si los ejemplos están representados por vectores bidimensionales en los intervalos  $[0, 1]$  y  $[-1, 1]$ , el espacio de estados sería el plano:  $[0, 1], [-1, 1]$ .

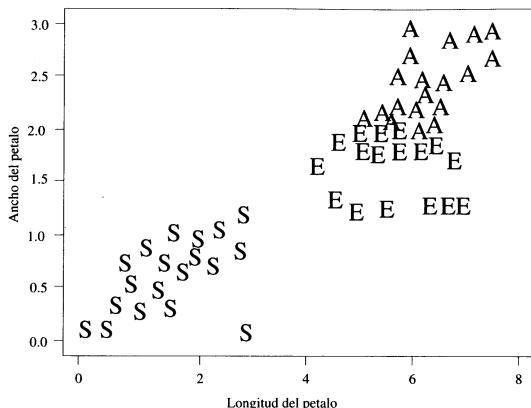


Figura 9.1: Conjunto de datos para clasificación

Se desea obtener una regla que discrimine, en función de la longitud y el ancho, las flores futuras. En este caso, al haber sólo dos características discriminantes, el espacio de estados es bidimensional, y se puede representar en un plano como se aprecia en la figura; al ser un espacio bidimensional las superficies discriminantes no serán hiperplanos sino rectas, pero el sistema no varía para un número mayor de dimensiones.

El discriminante de Fisher calcula los centroides de cada una de las clases S, E y A; a continuación se unen los centros de las clases adyacentes con sendas rectas, y se calcula la mediatrix de los segmentos anteriores. Estas mediatrixes serán las reglas discriminantes obtenidas por el método de Fisher (Figura 9.2), definidas por las ecuaciones correspondientes a dichas rectas.

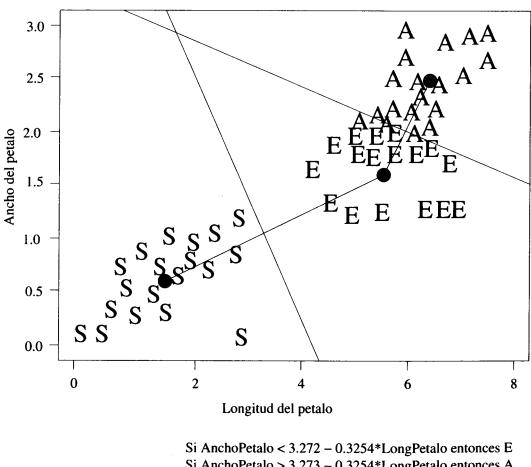


Figura 9.2: Clasificación mediante discriminantes de Fisher

En la figura se aprecia cómo las clases que están próximas son discriminadas más difícilmente, por lo que se producen errores. Además, una discriminación lineal no tiene por qué ser una solución adecuada, y en el caso de haber ejemplos de la misma clase en regiones muy diferentes del espacio, y en varios agrupamientos, las soluciones mediante estos métodos pueden ser muy disparatadas.

### Métodos basados en reglas

Están basados en la partición recursiva del espacio de datos. El espacio se divide en dos bloques a partir de la selección de uno de los atributos. El atributo se elige minimizando cierta función medida para todos los atributos disponibles. Cada bloque se comprueba si puede ser dividido a su vez, eligiendo un nuevo atributo. El proceso se reitera hasta que la calidad de la clasificación no aumente haciendo más bloques. Se genera una regla por cada atributo, y la unión de todas las reglas constituye el clasificador.

En el caso del ejemplo anterior de la Figura 9.1, se elige en primer lugar el atributo longitud del pétalo. Si éste es menor que 3,45, entonces la flor corresponde a la clase S; si no es así, habrá que observar el ancho del pétalo. En la Figura 9.3 se ve la estructura final del clasificador.

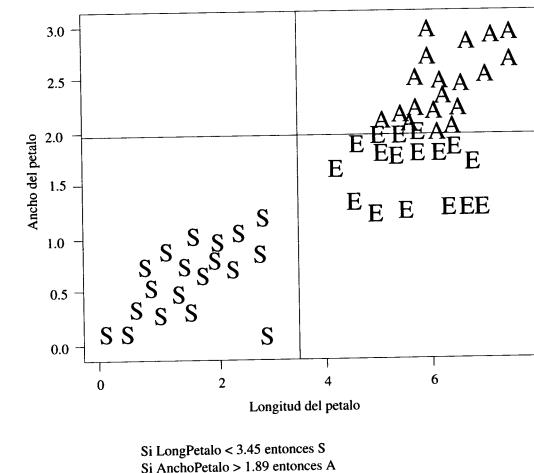


Figura 9.3: Clasificación mediante árboles de decisión

### Estimación de densidades

En este caso se determina para cada región del espacio la probabilidad de que un elemento que esté situado en ella pertenezca a cada una de las clases existentes. Este tipo de clasificadores será objeto de estudio en la siguiente sección; sólo se describirá aquí el caso de un clasificador por vecindad. En este caso no hay reglas prefijadas como en los casos anteriores, sino que la clasificación se irá haciendo para cada caso nuevo en particular. Cuando aparece un caso

nuevo, se calcula un círculo con centro en dicho punto y un radio prefijado como parámetro del sistema. Se calcula cuántos ejemplos caen dentro del círculo y se dice que el nuevo caso pertenece a la clase más numerosa dentro del círculo. En la Figura 9.4 se aprecia cómo en un primer caso, con un radio reducido, dentro del círculo hay dos ejemplos de la clase A y uno de la clase E, luego diremos que el nuevo dato pertenece a la clase A.

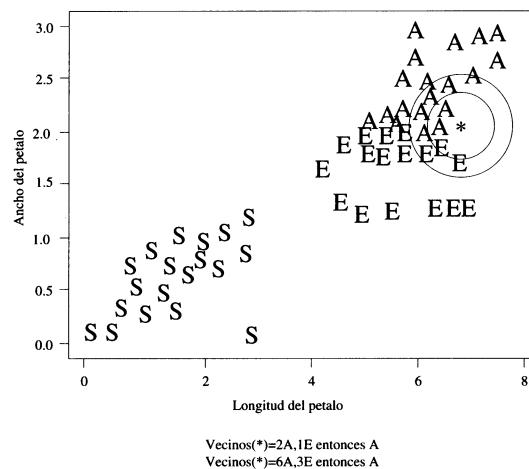


Figura 9.4: Clasificación mediante vecino más cercano

En la figura se aprecia también que el radio elegido influye en la clasificación. Si se aumenta o disminuye el número de ejemplos que caen dentro del círculo puede variar, y la predicción realizada también. Por lo tanto, este radio será un parámetro crítico a tener en cuenta.

## 9.2. Métodos de clasificación

Como se ha dicho anteriormente, existen dos tipos de sistemas clasificadores dependiendo de la naturaleza de los ejemplos disponibles, o de la clasificación a realizar. En los casos en los que se dispone de ejemplos etiquetados, que se corresponden con el aprendizaje supervisado en Redes Neuronales, se hablará simplemente de clasificación. Y en el caso en el que los ejemplos no están etiquetados, que corresponden a lo que en Redes Neuronales se llama aprendizaje no supervisado, se hablará de agrupamiento.

La clasificación tiene como objetivo el diseño automático de un sistema capaz de realizar buenas predicciones de clasificación para datos desconocidos. Para ello se basa únicamente en la información aportada por un conjunto de ejemplos ya clasificados de forma correcta, del que se deberán extraer las reglas que permitan hacer buenas generalizaciones. En este apartado se va a estudiar un método de clasificación basado en las Redes de Neuronas Artificiales no supervisadas de Kohonen, estudiadas previamente. Dicho método se conoce

con el nombre de Redes de Cuantización Vectorial (en inglés, *Learning Vector Quantization, LVQ*).

### 9.2.1. Redes de cuantización vectorial

Se puede decir que es una versión supervisada del método de Kohonen. En dicho método se introduce cada uno de los ejemplos de entrenamiento en la red. Para cada ejemplo se calcula la célula ganadora de la capa  $F_2$ . En la capa  $F_2$  están los prototipos de las clases que producirá como salida la red. Así pues, la célula ganadora corresponderá al prototipo asignado al ejemplo de entrada, la clase a la que se le hará pertenecer. A continuación se realiza el aprendizaje para dicha célula ganadora y las de su vecindario. Esto no puede hacerse en un aprendizaje supervisado, pues ya se conoce a priori a qué clase pertenece el ejemplo de entrenamiento. Así pues, se ha modificado el método para tener en cuenta esta información. El procedimiento es similar al de Kohonen, ya explicado en detalle en el Capítulo 3, de modo que aquí se describirá únicamente qué es lo que varía entre ambos métodos.

En primer lugar, a partir del conjunto de entrenamiento se conoce el número de categorías existentes; por lo tanto, en la capa  $F_2$  se introduce ese mismo número de células. Éstas constituyen los prototipos de la clasificación, y se distribuyen inicialmente de forma aleatoria por el espacio de estados. La solución al problema de clasificación será la colocación final de dichos prototipos, o células de la capa  $F_2$ , junto con la regla del vecino más cercano. La colocación de los prototipos vendrá especificada por los valores de los pesos de las conexiones entre la capa de entrada y la capa de competición de la red. Mientras que la regla del vecino más cercano funciona de la siguiente manera:

1. Se introduce el nuevo ejemplo a clasificar.
2. Se calcula la distancia de este nuevo ejemplo a todos los prototipos existentes.
3. Se etiqueta al nuevo ejemplo como de la clase del prototipo cuya distancia calculada en el paso 2 sea más pequeña.

En un modelo de Red de Neuronas Artificiales esta regla equivale simplemente a introducir el ejemplo en la red, propagarlo hasta obtener una salida y asignarle la clase a la que pertenece la célula ganadora.

Una vez distribuidos los prototipos de forma aleatoria, se van desplazando a medida que se van introduciendo los ejemplos de entrenamiento. Para ello se utiliza la misma regla de aprendizaje que en el método de Kohonen:

$$\frac{d\mu_{ij}}{dt} = \alpha(t)\tau_j(t)(\epsilon_i(t) - \mu_{ij}(t)) \quad (9.1)$$

Pero en este caso se varía la función  $\tau_i$  de la siguiente forma:

$$\tau_i = \begin{cases} 1 & \text{si } C_i \text{ ganadora y pertenece a la misma clase que } \epsilon_i \\ -1 & \text{si } C_i \text{ ganadora y no pertenece a la misma clase que } \epsilon_i \\ 0 & \text{en caso contrario} \end{cases}$$

Esta modificación de la función  $\tau_i$  cambia sustancialmente el procedimiento. En primer lugar, al ser aprendizaje supervisado se ha eliminado el concepto de vecindario. Ahora sólo se modificará la célula ganadora, y las restantes no sufrirán ningún cambio. La modificación de la célula ganadora no es siempre en la misma dirección. En el método de Kohonen, la célula ganadora se aproximaba siempre al ejemplo introducido; de esta forma las células acababan en el centro de aglomeraciones de ejemplos. En el método LVQ, si la célula ganadora pertenece a la misma clase que el ejemplo, la salida de la red es correcta, y para reforzarla se aproxima la célula al ejemplo; de esta forma también se colocará cerca de los ejemplos a los que representa, y que son de su misma clase. Sin embargo, si la célula ganadora no pertenece a la misma clase que el ejemplo, se estará produciendo una salida incorrecta, y habrá que penalizar dicha salida. Para ello lo que se hace es **alejar** (de ahí el signo negativo de la función  $\tau_i$  en el segundo supuesto) la célula del ejemplo para que en presentaciones posteriores de este mismo ejemplo haya otras células que lo representen, y no ésta. Así se consigue que los prototipos se vayan paulatinamente acercando a los ejemplos a cuya clase representan, y alejando de aquellos ejemplos a los que no representan.

En este caso, el valor de la tasa de aprendizaje  $\alpha$  también se decrementa con el tiempo, al igual que ocurría en el método de Kohonen. Se recomienda que este parámetro se inicie a valores pequeños, 0,1 y se decremente de forma lineal muy ligeramente.

### 9.2.2. K medias

Se trata de una clasificación por vecindad no supervisada en la que se parte de un número determinado de prototipos, y de un conjunto de ejemplos a agrupar, sin etiquetar. La idea es situar a los prototipos en el espacio, de forma que los datos pertenecientes al mismo prototipo tengan características similares.

Todo ejemplo nuevo, una vez que los prototipos han sido correctamente situados, es comparado con éstos y asociado a aquél que sea el más próximo, en los términos de una distancia previamente elegida. Normalmente, se trata de la distancia euclídea.

El método tiene una fase de entrenamiento, que puede ser lenta, dependiendo del número de puntos a clasificar y de la dimensión del problema. Pero una vez terminado el entrenamiento, la clasificación de nuevos datos es muy rápida, gracias a que la comparación de distancias se realiza sólo con los prototipos.

El procedimiento es el siguiente:

- Se calcula para cada prototipo el ejemplo más próximo y se incluye en la lista de ejemplo del mismo.
- Despues de haber introducido todos los ejemplos, cada prototipo  $A_k$  tendrá un conjunto de ejemplos a los que representa.
- Se desplaza el prototipo hacia el centro de masas de su conjunto de ejemplos.

- Se repite el procedimiento hasta que ya no se desplazan los prototipos.

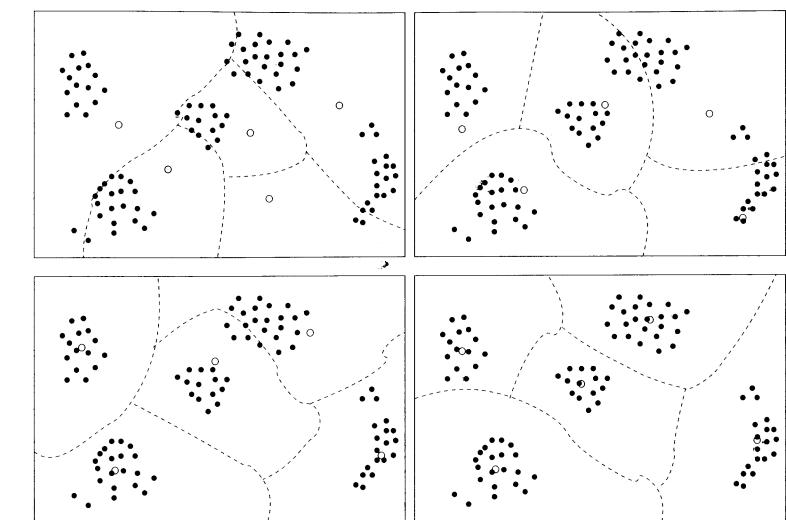


Figura 9.5: Ejemplo de evolución de un k medias

Mediante este algoritmo el espacio de ejemplos de entrada se divide en  $k$  clases o regiones (Figura 9.5), y el prototipo de cada clase estará en el centro de la misma. Dichos centros se determinan con el objetivo de minimizar las distancias cuadráticas euclídeas entre los patrones de entrada y el centro más cercano, es decir minimizando el valor  $J$ :

$$J = \sum_{i=1}^k \sum_{n=1}^m M_{in} \|x_n - A_i\|^2$$

donde  $m$  es el conjunto de patrones,  $\|\cdot\|$  es la distancia euclídea,  $x_n$  es el ejemplo de entrada  $n$ ,  $A_i$  es el prototipo de la clase  $i$ , y  $M_{in}$  es la función de pertenencia del ejemplo  $n$  a la región  $i$  de forma que vale 1 si el prototipo  $A_i$  es el más cercano al ejemplo  $x_n$  y 0 en caso contrario, es decir:

$$M_{in} = \begin{cases} 1 & \text{si } \|x_n - A_i\|^2 < \|x_n - A_s\|^2 \forall s \neq i, s = 1, 2, \dots, k \\ 0 & \text{en caso contrario} \end{cases}$$

Este es un caso parecido al de los mapas de Kohonen. En este caso la diferencia es que los prototipos no son desplazados después de la presentación de cada ejemplo, sino al final de cada iteración, con lo cual hay que determinar una regla para mover los prototipos. Esta regla consiste simplemente en moverlos hasta el centro de masas de los ejemplos a los que representan. Como tras una nueva iteración, los ejemplos son reasignados debido al desplazamiento de todos los prototipos al final de la iteración anterior, es necesario volver a calcular los centros de masas de los nuevos ejemplos. A medida que los prototipos

vayan situándose, se producirán menos cambios en sus conjuntos de ejemplos, y su desplazamiento será menor. Así, poco a poco los desplazamientos se van haciendo más ligeros, hasta desaparecer. La colocación final de los prototipos definirá la solución encontrada por el método. En la Figura 9.5 hay un ejemplo de este proceso.

### 9.3. Clasificación de enfermos de diabetes

La enfermedad conocida como diabetes está asociada a una proteína llamada insulina. Esta enfermedad puede deberse a una ausencia o baja producción de insulina por parte del páncreas debida a un ataque al sistema inmunológico, o a la resistencia a la insulina asociada típicamente a pacientes maduros u obesos. En cualquier caso, este problema con la insulina produce grandes repercusiones metabólicas. Sin embargo, una vez que un paciente ha sido diagnosticado y recibe el tratamiento adecuado, estos problemas pueden paliarse y no revestir ninguna gravedad. Así pues, el diagnóstico precoz y correcto de la enfermedad es importante para paliar los posibles problemas derivados de la misma.

En este ejemplo se va a tratar de diagnosticar si un paciente tiene o no diabetes a partir de un conjunto de datos y pruebas realizadas, mediante una Red de Neuronas LVQ. Existe información de 768 mujeres con más de 21 años de edad. Para cada mujer se han recogido los siguientes datos:

- Número de veces que ha estado embarazada.
- Concentración de glucosa en la sangre dos horas después de haber efectuado una prueba oral de tolerancia a la glucosa.
- Presión sanguínea diastólica.
- Grosor de la piel en el tríceps.
- Concentración de insulina.
- Índice de masa corporal. Peso en kilos entre altura en metros al cuadrado.
- Historial diabético.
- Edad.
- Si tiene o no la enfermedad.

El último dato es el relativo a si la mujer es o no diabética. Es precisamente la clase en la que habrá que clasificar los datos. Una vez entrenada, la red debe ser capaz de decidir si un individuo nuevo, para el que se disponen los datos especificados, padece de diabetes o no.

En primer lugar se debe dividir el conjunto de datos disponibles en dos nuevos conjuntos; uno se utilizará como conjunto de aprendizaje de la red y el otro para realizar la validación del comportamiento de la red ya entrenada. En este

caso se ha dividido de la siguiente forma: de los datos disponibles, 576 seleccionados aleatoriamente se utilizarán para entrenamiento y 192 para validación. El objeto del conjunto de validación es comprobar cuál sería el comportamiento de la red ya entrenada al enfrentarse a datos nuevos. En algunas ocasiones puede ocurrir que la red dé muy buenos resultados en entrenamiento, sea capaz de clasificar casi la totalidad de los datos, pero no produzca buenos resultados ante casos desconocidos. Esto es debido a que la red se adapta demasiado a los ejemplos con que ha sido entrenada; se puede decir que tiende más a "memorizar" los ejemplos que a representar una regla que discrimine de forma general los casos. El objetivo de la experimentación será, por tanto, alcanzar buenos resultados de clasificación en el conjunto de validación.

La distribución de ejemplos de clases en cada conjunto es la siguiente:

Clase	Entrenamiento	Validación
Sí	198 (34,37 %)	70 (36,46 %)
No	378 (65,63 %)	122 (63,54 %)

Se aprecia que las distribuciones por clases en ambos casos son casi idénticas, por lo que la división es aceptable. En caso de haberse obtenido distribuciones dispares de las clases sería conveniente repetir el proceso.

A continuación habrá que diseñar la red. La dimensión de la entrada es ocho, ya que hay ocho características discriminantes; la novena característica de los datos es la clase que se utiliza en el proceso de aprendizaje del LVQ, como ya se ha mencionado. Por lo tanto, la red deberá tener ocho células en la capa de entrada. En cuanto a la capa de competición, dependerá del número de prototipos que se desee obtener. En principio, al haber sólo dos clases parece razonable que haya también sólo dos prototipos. Esto sería ideal si las clases estuvieran separadas en dos bloques; sin embargo, esto no tiene por qué ser así. Se desconoce la forma de las distribuciones de ejemplos y, por lo tanto, también el número óptimo de prototipos a generar. En la Figura 9.6 se ilustra esto con un ejemplo.

En el primer caso se aprecia cómo la disposición de las clases hace que una solución con dos prototipos sea siempre poco eficiente. No existe una disposición de los prototipos que, con la regla del vecino más próximo, reduzca el error de clasificación por debajo de un determinado límite razonable. En éste se necesitarán cinco prototipos para dar una solución óptima. Sin embargo, el ejemplo anterior es un caso ideal ilustrativo de lo que se está explicando: en la mayoría de los casos, las distribuciones no están tan separadas y son difíciles de representar al no ser bidimensionales. En el caso de los enfermos de diabetes, el espacio de estados tiene ocho dimensiones, y es imposible de representar gráficamente.

La solución es hacer una estimación genérica del número de prototipos y realizar varios experimentos mediante ensayo y error. Es decir, probar con diferente número de prototipos y elegir la red que mejores resultados de validación produzca. Una buena regla es comenzar con el doble de prototipos que de clases existentes; en este caso se empieza con cuatro.

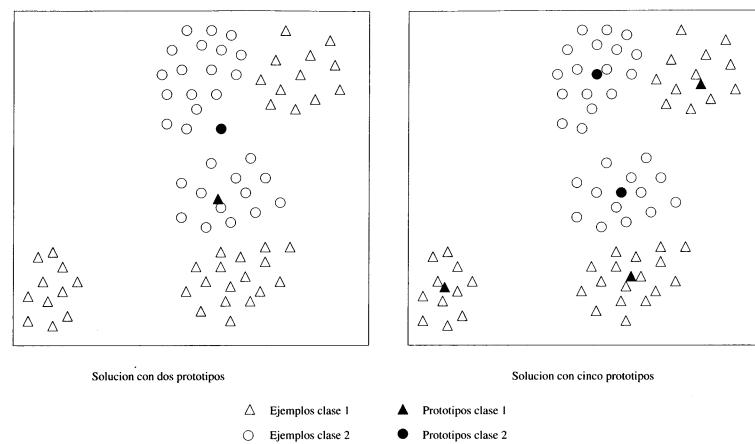


Figura 9.6: Ejemplo de distribución de dos clases

La red tendrá ocho células en la capa de entrada, todas ellas conectadas con las cuatro células de la capa de competición (Figura 9.7).

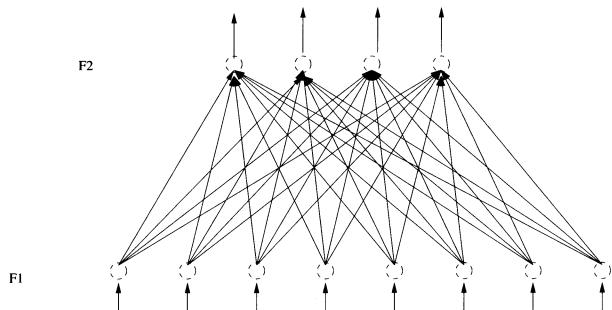


Figura 9.7: Red LVQ con cuatro prototipos para el problema de la diabetes

Se han realizado varios experimentos cambiando parámetros de la red, hasta conseguir los mejores resultados posibles (o viables). También se ha experimentado cambiando dos de los posibles parámetros. En primer lugar se ha mantenido la configuración anterior de cuatro prototipos, pero se han realizado dos experimentos, uno balanceando los prototipos y otro sin balancearlos. Balancear los prototipos quiere decir que los prototipos serán asignados a clases de forma proporcional al número de ejemplos de la clase que existan en el conjunto de ejemplos. En este caso existen dos clases etiquetadas con "SÍ" y "NO". La clase NO contiene 378 elementos y la clase SÍ 198; esto se traduce en que el 65,62 % de los ejemplos son de la clase NO y el 34,38 % son de la clase SÍ. Si se balancea habrá que crear, del total de prototipos, esa misma proporción de las dos clases. En el caso de cuatro prototipos se corresponde con tres para la clase NO

y uno para la clase SÍ. Si no se balancea, simplemente se crearán la mitad de prototipos de cada clase.

Estos dos mismos experimentos se han realizado para 8 y 12 prototipos, y finalmente para dos prototipos, para comprobar si las clases están separadas en dos cúmulos, o se produce lo mismo que ocurre en la Figura 9.6, en la que no es posible separar los datos de forma precisa utilizando únicamente el mismo número de prototipos que de clases.

Los resultados son los que aparecen en el cuadro 9.3. En él existen tres columnas principales para indicar, respectivamente, los parámetros con los que se ha realizado el experimento: el error cometido en el conjunto de entrenamiento, contabilizado en porcentaje de datos que no han sido clasificados correctamente, y el error cometido en el conjunto de validación, medido de la misma manera. En cuanto a los parámetros, se ha variado el número de prototipos y si se han balanceado o no los prototipos iniciales. En cuanto a los errores, se incluyen tres diferentes: los que, siendo de la clase NO, han sido clasificados como de la clase SÍ (columna NO); los que, siendo de la clase SÍ, han sido clasificados como de la clase NO (columna SÍ), y el total de ejemplos mal clasificados.

Parámetros	Error entrenamiento			Error validación			
	Prot.	Balanc.	NO	SÍ	Total	NO	SÍ
4	NO	84.13 %	40.91 %	69.27 %	87.70 %	45.71 %	72.40 %
4	SÍ	92.06 %	19.19 %	67.01 %	92.62 %	15.71 %	64.58 %
8	NO	80.69 %	57.07 %	72.57 %	75.41 %	58.57 %	69.27 %
8	SÍ	79.10 %	60.61 %	72.74 %	75.41 %	55.71 %	68.23 %
16	NO	81.75 %	61.11 %	74.65 %	76.23 %	55.71 %	68.75 %
16	SÍ	87.30 %	49.49 %	74.31 %	84.43 %	44.29 %	69.79 %
2	NO	85.71 %	35.35 %	68.40 %	78.69 %	27.14 %	59.90 %
2	SÍ	87.30 %	33.33 %	68.75 %	81.97 %	22.86 %	60.42 %

Cuadro 9.1: Resultados del problema de diabetes con una red LVQ de cuatro prototipos

El cuadro aporta diferente información 9.3. En primer lugar la tasa de acierto se sitúa en torno al 70 %. Es un buen valor de reconocimiento, comparable con el que producen los mejores resultados publicados para este problema<sup>2</sup>. Sin embargo, lo que se aprecia a primera vista es que los aciertos en las dos categorías difieren significativamente. Es mucho más fácil decidir que alguien no tiene la enfermedad, que decidir lo contrario. Esto puede ser debido a que hay muchos más ejemplos de personas no enfermas que de personas enfermas (378 frente a 198). El balanceo de los prototipos no soluciona nada, pues se ve que en todos los casos la tasa de acierto de la clase minoritaria disminuye. Lo que sí parece mejorar el acierto de la clase SÍ, es el aumento del número de prototipos hasta al menos ocho, aunque se resiente el error total. Por último, se ve también que

<sup>2</sup>Los mejores resultados publicados producen una tasa de acierto de alrededor del 74 %.

el utilizar el mismo número de prototipos que de clases da los peores resultados. Se confirma lo que se había adelantado previamente, y se producen aciertos por debajo del 60 %, llegando a bajar del 23 % para la clase SÍ.

En cuanto al error de entrenamiento, se aprecia que para pocos prototipos –2, 4– el error de entrenamiento se aproxima bastante al de predicción; esto es debido a que es más difícil hacer una sobreestimación del conjunto de aprendizaje con pocos prototipos que con muchos. Con sólo cuatro prototipos, su colocación no puede ajustarse a pequeñas acumulaciones de puntos de entrenamiento, ya que las grandes acumulaciones tirarán más de ellos. En cambio, con muchos prototipos sí puede reservarse uno para estas pequeñas acumulaciones, que no son generalizables al conjunto de validación, y hacen aumentar el error de este último conjunto. En cualquier caso, al no haberse realizado experimentos con un número de prototipos excesivamente elevado, en el resultado final no se aprecian sensiblemente las consecuencias del sobreaprendizaje, es relativamente ligero.

## Bibliografía

- [Amari, 1971] Amari, S. (1971). Characteristics of randomly connected threshold-element networks and network systems. *Proceeding of the IEEE*, 59(1):35–47.
- [Amari, 1972] Amari, S. (1972). Characteristics of random nets of analog neuron-like elements. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-2:643–657.
- [Amari, 1974] Amari, S. (1974). A method of statistical neurodynamics. *Kybernetik*, 14:201–215.
- [Amari, 1977] Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–87.
- [Amari, 1982] Amari, S. (1982). *A mathematical theory of self-organizing nerve systems*. Biomathematics: Current Status and Perspectives. North-Holland, Amsterdam.
- [Amari, 1983] Amari, S. (1983). Field theory of self-organizing neural nets. *IEEE Transaction on Systems, Man and Cybernetics*, SMC-13:741–748.
- [Amari and Takeuchi, 1978] Amari, S. and Takeuchi, M. (1978). Mathematical theory on formation of category detecting nerve cells. *Biological Cybernetics*, 29:127–136.
- [Anastasio, 1991] Anastasio, T. (1991). A recurrent neural network model of velocity storage in the vestibulo-ocular reflex. In *Advances in Neural Information Processing Systems*, pages 32–38. Morgan Kaufmann.
- [Anderson, 1968] Anderson, J. (1968). A memory storage model utilizing spatial correlation functions. *Kybernetik*, 5:113–119.
- [Anderson, 1970] Anderson, J. (1970). Two models for memory organization using interacting traces. *Mathematical Biosciences*, 8:137–160.
- [Anderson, 1973] Anderson, J. (1973). A theory for the recognition of items from short memorized lists. *Psychological Review*, 80:417–438.

[Zurada et al., 1997] Zurada, J., Malinowski, A., and Usui, S. (1997). Perturbation method for deleting redundant inputs for perceptron networks. *Neurocomputing*, 14:177–193.

## Índice alfabético

- activación global, 94
- activación local, 94
- Adaline, 47–49
- agrupamiento, 138, 213
- algoritmo de aprendizaje recurrente en tiempo real, 131
- algoritmo de retropropagación, 64, 66
- algoritmo de retropropagación a través del tiempo, 128
- análisis de componentes principales, 138
- aprendizaje, 15, 18, 20
- aprendizaje competitivo, 145, 146, 168
- aprendizaje en tiempo real, 119
- aprendizaje hebbiano, 29, 142, 149, 153
- aprendizaje no supervisado, 22, 137
- aprendizaje por épocas, 118
- aprendizaje por refuerzo, 23
- aprendizaje supervisado, 22
- aproximación global, 90, 115
- aproximación local, 90, 94, 115
- arquitectura, 17
- asincronía, 147, 172
- axón, 13, 14
- célula ganadora, 147, 149, 152, 153
- células de McCulloch-Pitts, 28, 35, 36, 38
- calidad de un clasificador, 213
- capa de clasificación, 146
- capa de competición, 146, 150, 168
- capa de entrada, 146, 150, 168
- categoría, 146
- categorización, 139, 146
- clasificación, 213
- clasificador por vecindad, 217
- codificación, 138
- conexiones bottom-up, 168
- conexión inhibitoria, 174
- conexion excitatoria, 173
- conexion inhibitoria, 173
- conexiones bottom-up, 175
- conexiones recurrentes, 117, 123, 128
- conexiones top-down, 168, 176
- conjunto de entrenamiento, 24
- conjunto de validación, 24
- control, 199, 201
- control inverso especializado, 203
- control inverso generalizado, 202
- control neuronal, 201
- control predictivo, 205
- criterio de convergencia, 20, 21
- dendritas, 14
- descenso del gradiente, 48, 65, 66, 104, 123
- dilema estabilidad-plasticidad, 166, 167
- discriminante de Fisher, 215
- discriminantes lineales, 215
- distancia de vecindario, 157
- distancia euclídea, 151, 152
- esquema de aprendizaje, 155
- estabilidad, 166
- estimación de densidad, 217
- extracción de características, 138
- factor de vigilancia, 169, 171
- falso negativo, 214

falso positivo, 214  
 familiaridad de patrones, 137  
 filtro adaptativo, 173  
 función de activación, 16, 18, 19  
 función de base radial, 92  
 función de olvido, 154  
 función de salida, 39, 40  
 función distancia, 151  
 función energía, 122  
 función escalón, 39  
 función lógica AND, 37, 42  
 función lógica NOT, 36  
 función lógica OR, 37  
 función lógica OR exclusivo, 52, 54  
 función sigmoidal, 61, 70  
 función sombrero mexicano, 152  
 función tangente hiperbólica, 61, 70  
 función umbral, 47  
 horizonte de predicción, 187, 193, 206  
 inestabilidad, 168  
 interacción lateral, 142, 143, 145, 146, 152  
 jacobiano del proceso, 204  
 K medias, 97, 220  
 linealmente separable, 54, 55  
 máquina de Boltzmann, 31  
 método híbrido, 97, 103, 109  
 método totalmente supervisado, 103, 108, 109  
 mínimo local, 81  
 mínimos cuadrados, 100  
 mapas autoorganizativos de Kohonen, 150  
 matriz seudoinversa, 102  
 memoria a corto plazo, 167  
 memoria a largo plazo, 167  
 memoria autoajustable, 177  
 modelización, 199, 200  
 modelo del proceso, 199, 207  
 modelo del proseo, 200  
 modelos de predicción, 188

modelos NAR, 187, 194  
 modelos NARMA, 200, 207  
 momento, 74  
 neurona artificial, 15, 16  
 neuronas de contexto, 124–126  
 nivel de activación, 15  
 np-completo, 163  
 onda de reset, 172  
 parálisis, 81  
 patrón de conectividad, 17  
 Perceptron, 38, 41, 45, 49  
 Perceptron multicapa, 58, 63  
 plasticidad, 141, 166  
 predicción en múltiples pasos de tiempo, 187, 190  
 predicción en un paso de tiempo, 187, 189  
 problema del viajante, 163  
 proceso de convergencia, 139  
 proceso de divergencia, 139  
 proceso dinámico, 199  
 propiedad topológica, 142, 143, 165  
 prototipado, 138  
 prototipo, 146  
 punto estable, 118, 122, 123  
 razón de aprendizaje, 65, 74, 85, 112  
 red de Elman, 126, 127  
 red de Hopfield, 120  
 red de Jordan, 125, 127  
 redes de base radial, 90  
 redes de cuantización vectorial, 219  
 redes parcialmente recurrentes, 119, 123  
 redes recurrentes, 117, 119, 127  
 regla de Hebb, 141, 142, 153  
 regla de los dos tercios, 170  
 regla del vecino más cercano, 219  
 regla Delta, 47, 49, 50  
 regla Delta generalizada, 56, 66, 72  
 relación de características, 138  
 resonancia, 174  
 ruido, 168

señal inhibitoria, 169  
 serie temporal, 185  
 sinapsis, 13, 14, 24, 30  
 sistema de control, 199, 201  
 sistemas autoorganizados, 23  
 sobreaprendizaje, 78, 79  
 subsistema atencional, 167  
 subsistema orientador, 167, 171  
 término de control, 167  
 tarea de clasificación, 211, 213  
 tarea de predicción, 213  
 tasa de aprendizaje, 153, 158  
 Teoría de la resonancia adaptativa, 145, 149, 166  
 umbral, 39, 42  
 unidades autoescalables, 177  
 valor predictivo, 173  
 vecindario, 156, 158, 160  
 vecindario bidimensional, 156  
 vecindario en estrella, 156  
 vecindario tridimensional, 156  
 vecindario unidimensional, 156, 163  
 winer takes all, 148, 149, 153, 156