Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Estudio de Redes Neuronales y Aplicaciones Prácticas

Trabajo de graduación previo a la obtención del título de

Ingeniero de Sistemas

Autores: Mauricio Córdova Q.
Pablo Mora M.

Director: Ing. Pablo Pintado

Cuenca, Ecuador

2007

DEDICATORIA

Mauricio Córdova Q.

Quiero dedicar esta monografía a mis padres, Jorge y Vilma, y a mi hermana Ximena, por ser mi ejemplo de humildad, lucha y perseverancia, por enseñarme a salir adelante a pesar de todas las cosas y por todo el apoyo incondicional que me han dado durante mi vida universitaria.

Pablo Mora M.

Quiero dedicar esta monografía a mis padres Rigoberto y Griselda. Quienes han sido una fuente de inspiración y humildad, que con su esfuerzo, consejos y apoyo incondicional que me han ayudado para salir adelante. Gracias.

AGRADECIEMIENTOS

Queremos agradecer al Ing. Pablo Pintado por todo el apoyo y la colaboración prestada en todo momento durante la realización de esta monografía, y a todos los profesores que sin titubeo compartieron todo su conocimiento dentro y fuera de las aulas de clase.

Por último queremos agradecer a la Universidad del Azuay por darnos la oportunidad de crecer y desarrollarnos dentro del plano universitario, social y personal.

Índice de Contenidos

RESUMEN

Las Redes de Neuronas Artificiales son un tipo de aprendizaje y de procesamiento automático inspirado en el funcionamiento del sistema nervioso. Éstas simulan las propiedades de los sistemas neuronales biológicos a través de modelos matemáticos recreados mediante mecanismos artificiales, tratando de modelar esquemáticamente la estructura hardware del cerebro. Estos sistemas de procesamiento de información, paralelos, distribuidos y adaptativos, son capaces de aprender de la experiencia a partir de datos del entorno y empleando algoritmos numéricos.

Conforme avanzamos hacia una era de máquinas inteligentes y automatización del razonamiento, resulta evidente la necesidad de una metodología que refleje la capacidad del ser humano de tomar decisiones en un entorno de imprecisión e incertidumbre, cuyo fin es diseñar y construir máquinas con un coeficiente de inteligencia elevado.

Es por todo esto que hemos decidido investigar este tema en donde analizamos algunos de los modelos de redes neuronales que consideramos importantes con sus posibles aplicaciones prácticas. Como resultado de este estudio obtendremos un documento explicativo que esperamos contribuya al entendimiento y desarrollo de este tema dentro de nuestro medio.

ABSTRACT

Artificial Neural Networks are a type of automatic learning and processing inspired by the operation of the nervous system. They simulate the properties of the biological neuron systems through mathematical models recreated by means of artificial mechanisms, trying to schematically model the brain hardware structure. These parallel, distributed, and adaptable information processing systems are able to learn from the experience starting out from data of the environment and using numerical algorithms.

As we advance toward an era of intelligent machines and automation of reasoning, the need of a methodology that reflects the capacity of the human being to take decisions in an imprecision and uncertain environment becomes evident. Its goal would be to design and build machines with a high intelligence quotient.

For all the above mentioned, we have decided to investigate this topic where we analyze some neuron network models that we consider important, as well as their possible practical applications. As a result of this study, we will obtain an explanatory document which, we hope, will contribute to the understanding and development of this subject in our society**.**

INTRODUCCIÓN

Las redes neuronales son construidas para imitar el proceso de aprendizaje biológico humano inteligente, auto-modificación, y aprendizaje por creación de inferencias, proveen un aprendizaje por experiencia dinámica.

Las redes neuronales no son más que un modelo artificial y simplificado del cerebro humano, que es el ejemplo perfecto que disponemos para un sistema que es capaz de adquirir conocimiento a través de la experiencia. Una red neuronal es un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona.

El desarrollo de las redes neuronales se inspira en las soluciones que la naturaleza ha encontrado a lo largo de millones de años de evolución para el tratamiento del tipo de información masiva y distorsionada procedente del entorno natural, soluciones que copiadas en sistemas artificiales se espera que contribuyan a resolver importantes problemas tecnológicos (visión, habla, inteligencia artificial, etc.).

Con la utilización de las redes neuronales artificiales se puede incorporar un cierto tipo de inteligencia en un sistema de procesamiento y control. Cada día está más extendida la opinión de que sistemas como los neuronales van a desempeñar un papel importante en la construcción de máquinas que emulen la capacidad humana de toma de decisiones en entornos imprecisos y con ruido. En la práctica, una red neuronal artificial puede simularse mediante un programa, o bien realizarse en circuitos electrónicos específicos. En este estudio nos centraremos especialmente en las redes neuronales que actualmente está causando mayor impacto, debido a su aplicabilidad práctica.

CAPITULO 1
FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES

A lo largo de este capítulo expondremos los fundamentos básicos de las redes neuronales artificiales o ANS (Artificial Neural Systems). Describiremos los aspectos principales de los ANS, relacionados con la estructura de la neurona artificial y de la arquitectura de la red. Posteriormente trataremos algunas de las aplicaciones prácticas de las redes neuronales.

1.1 LA NEURONA BIOLOGICA

Una neurona biológica es una célula especializada en procesar información. Vistas bajo el microscopio, las neuronas pueden presentarse de diferentes formas, aunque muchas de ellas presentan un aspecto similar al de la figura 1.1, con un cuerpo celular o soma (de entre 10 y 80 micras de longitud), del que surge un denso árbol de ramificaciones compuesto por las dendritas, y del cual parte la fibra tubular denominada axón, que también se ramifica en su extremo final para conectar con otras neuronas. Desde un punto de vista funcional, las neuronas constituyen procesadores de información sencillos. Como todo sistema de este tipo, poseen un canal de entrada de información, las dendritas, un órgano de cómputo, el soma, y un canal de salida, el axón.

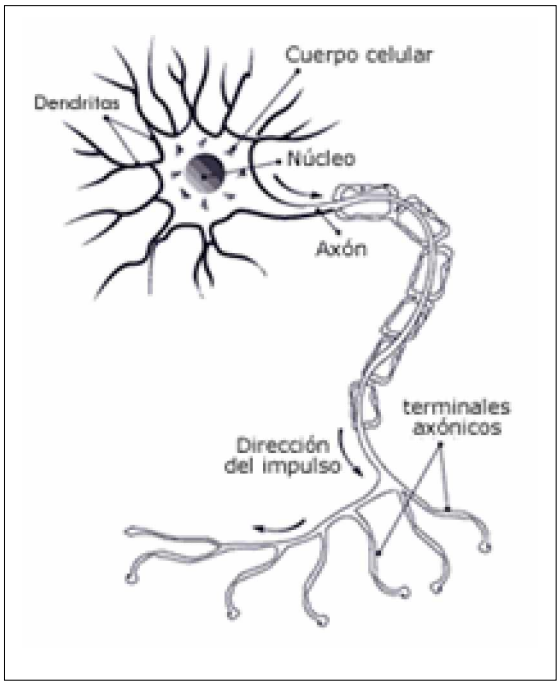


Figura 1.1 Estructura de una neurona biológica típica

La unión entre neuronas se denomina sinapsis. Aquí se habla de neuronas presinápticas (que envía las señales) y la neuronas postsinápticas (reciben la información) representadas en la figura 1.2. Las sinapsis son direccionales, es decir, la información fluye siempre en un único sentido. La forma de comunicación más habitual entre dos neuronas es de tipo químico. La neurona presináptica libera sustancias químicas complejas denominadas neurotransmisores, que atraviesan el vacío sináptico. Si la neurona postsináptica posee en las dendritas o en el soma canales sensibles a los neurotransmisores liberados, los fijarán, y como consecuencia de ello permitirán el paso de determinados iones a través de la membrana. Las corrientes iónicas que de esta manera se crean pequeños potenciales postsinápticos, este es el origen de la existencia de sinapsis excitatorias y de sinapsis inhibitorias.

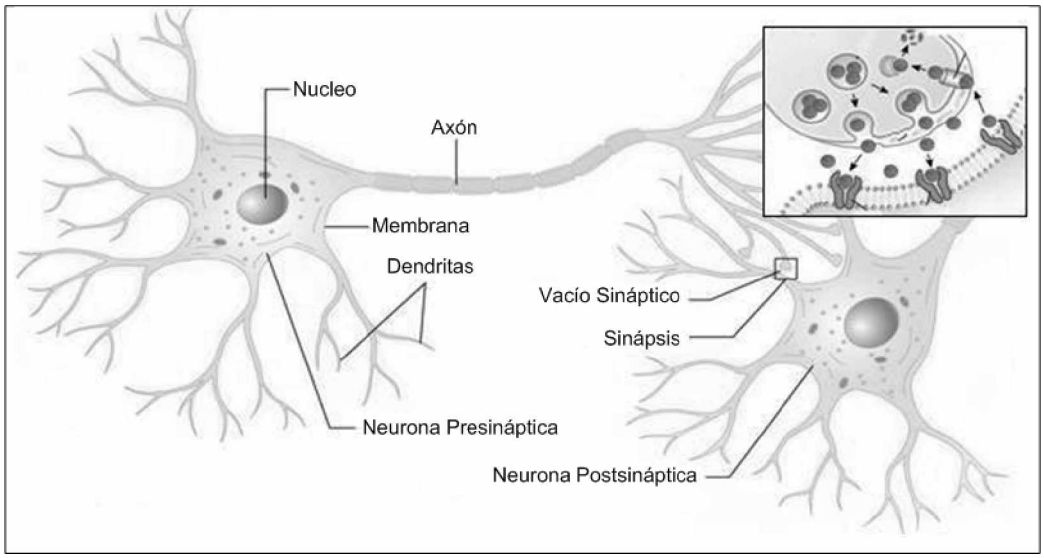


Figura 1.2 Neuronas presináptica y postsináptica

1.2 ESTRUCTURA DE UN SISTEMA NEURONAL ARTIFICIAL

Cerebro y computador

Los ANS imitan la estructura hardware del sistema nervioso, con la intención de construir sistemas de procesamiento de la información paralelos, distribuidos y adaptativos, que puedan presentar un cierto comportamiento "inteligente". Como sabemos, un computador convencional es, en esencia, una máquina de von Neumann, construida en torno a un único procesador, que ejecuta de un modo secuencial un programa almacenado en memoria. Por el contrario, el cerebro no está compuesto por un único procesador, sino por millones de ellos (neuronas).

Sistemas paralelos, distribuidos y adaptativos

Los tres conceptos clave de los sistemas nerviosos, que se pretende emular en los sistemas artificiales, son: paralelismo de cálculo, memoria distribuida y adaptabilidad al

entorno. De esta manera, podemos hablar de las redes neuronales como sistemas paralelos, distribuidos y adaptativos.

Procesamiento paralelo: un ordenador convencional, que trabaja secuencialmente, emplearía varios minutos en realizar sobre una imagen compuesta por 256x256 pixeles, una sencilla tarea de tratamiento en bajo nivel (acentuar contrastes, extraer contornos, etc.), mucho más simple que la que llevaba a cabo el sistema visual para reconocer una imagen. El cerebro tarda aproximadamente 20 ms en preprocesar una imagen compuesta por millones de píxeles, extraer sus rasgos característicos, analizarla, e interpretarla. Ningún sistema creado por el hombre es capaz de realizar algo semejante. La clave reside en que en este último caso los miles de millones de neuronas que intervienen en el proceso de visión están operando en paralelo sobre la totalidad de la imagen.

Memoria distribuida: mientras que en un computador la información ocupa posiciones de memoria bien definidas, en los sistemas neuronales se encuentra distribuida por las sinapsis de la red, de modo que si una sinapsis resulta dañada, no perdemos más que una parte muy pequeña de la información.

Adaptabilidad: los ANS se adaptan fácilmente al entorno modificando sus sinapsis y además aprenden de la experiencia, pudiendo generalizar conceptos a partir de casos particulares.

Estructura de un sistema neuronal artificial

Los elementos básicos de un sistema neuronal biológico son las neuronas, que se agrupan en conjuntos compuestos por millones de ellas organizadas en capas. Un conjunto de estos subsistemas da lugar a un sistema global (el sistema nervioso. En la

realización de un sistema neuronal artificial puede establecerse una estructura jerárquica similar. El elemento esencial es la neurona artificial, que se organizará en capas; varias capas forman una red neuronal; y, por último, una red neuronal, junto con las interfaces de entrada y salida, más los módulos convencionales adicionales necesarios, constituirán el sistema global de proceso.

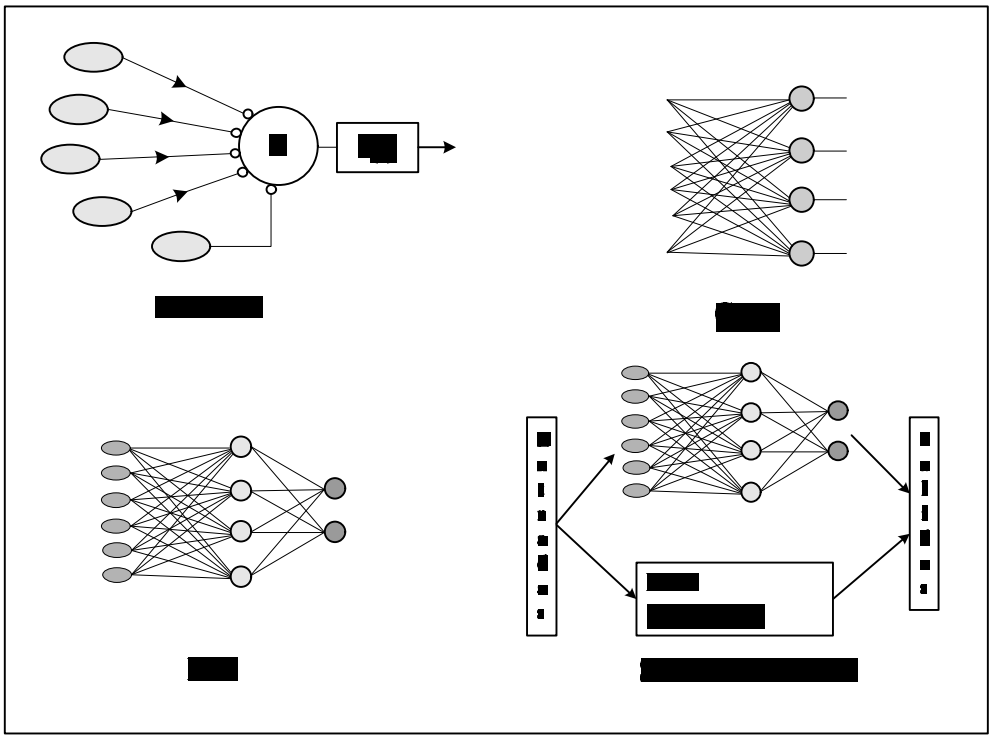


Figura 1.3 Estructura jerárquica de un Sistema Neuronal Artificial.

Según los autores Rumelhart y McClelland, un sistema neuronal, está compuesto por los siguientes elementos:

- Un conjunto de procesadores elementales o neuronas artificiales.
- Un patrón de conectividad o arquitectura.
- Una dinámica de activaciones.

- Una regla o dinámica de aprendizaje.

- El entorno donde opera.

1.3 MODELO DE UNA NEURONA ARTIFICIAL

1.3.1 Modelo general de neurona artificial

A continuación describiremos la estructura genérica de neurona artificial:

Se denomina procesador elemental o neurona a un dispositivo simple de cálculo que, a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta o salida. Los elementos que constituyen la neurona de etiqueta i son los siguientes:

- Conjunto de entradas, $x_j(t)$.

- Pesos sinápticos de la neurona i, w_{ij} que representa la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i.

- Regla de propagación $\sigma(w_{ij}, x_j(t))$, que proporciona el valor del potencial postsináptico $h_i(t) = \sigma(w_{ij}, x(t))$ de la neurona i en función de sus pesos y entradas.

- Función de activación $f_i(a_i(t-1), h_i(t))$, que proporciona el estado de activación actual $a_i(t) = f_i(a_i(t-1), h_i(t))$ de la neurona *i*, en función de su estado anterior $a_i(t-1)$ y de su potencial postsináptico actual.

- Función de salida $F_i(a_i(t))$, que proporciona la salida actual $y_i(t) = F_i(a_i(t))$ de la neurona *i* en función de su estado de activación.

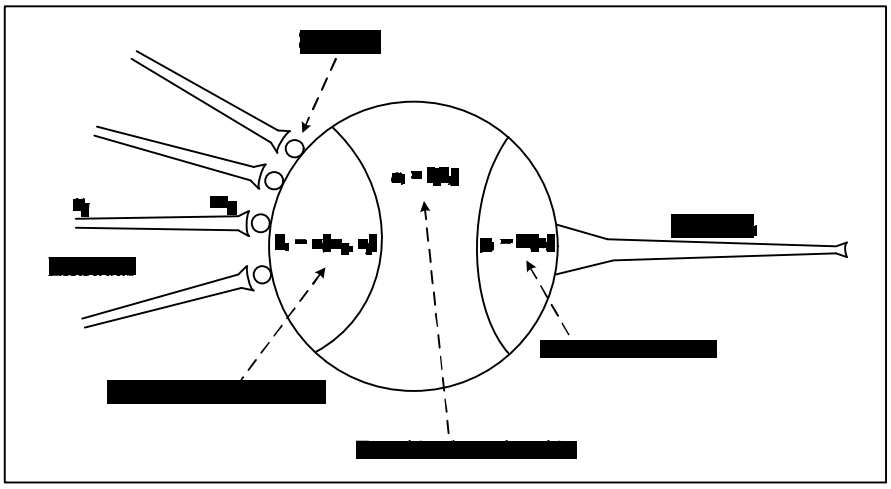


Figura 1.4 Modelo general de una neurona artificial

De este modo, la operación de la neurona *i* puede expresarse como

$$y_i(t) = F_i(f_i[a_i(t-1), \sigma_i(w_{ij}, x_j(t))])$$

Fórmula 1.1

Entradas y salidas

Las variables de entrada y salida pueden ser binarias (digitales) o continuas (analógicas), dependiendo del modelo y aplicación. Dependiendo del tipo de salida, las neuronas suelen recibir nombres específicos. Las neuronas estándar cuya salida sólo puede tomar los valores 0 o 1 se suelen denominar genéricamente neuronas de tipo McCulloch-Pitts,

mientras que aquellas que únicamente pueden tener por salidas -1 o +1 se suelen denominar neuronas tipo Ising. Si puede adoptar diversos valores discretos en la salida (por ejemplo, -2, -1, 0, +1, +2), se dice que se trata de una neurona de tipo Potts.

Regla de propagación

La regla de propagación permite obtener, a partir de las entradas y los pesos, el valor del potencial postsináptico h_i de la neurona.

$$h_i(t) = \sigma_i(w_{ij}, x_j(t))$$

Fórmula 1.2

La función más habitual es de tipo lineal, y se basa en la suma ponderada de las entradas con los pesos sinápticos.

$$h_i(t) = \sum_j w_{ij} x_j$$

Fórmula 1.3

que formalmente también puede interpretarse como el producto escalar de los vectores de entrada y pesos.

$$h_i(t) = \sum_j w_{ij} x_j = w_i^T . x$$

Fórmula 1.4

El peso sináptico w_{ij} define en este caso la intensidad de interacción entre la neurona presináptica *j* y la postsináptica *i*. Dada una entrada positiva (procedente de un sensor o simplemente la salida de otra neurona), si el peso es positivo tenderá a excitar a la neurona postsináptica, si el peso es negativo tenderá a inhibirla. Así se habla de sinapsis excitadoras (de peso positivo) e inhibidoras (de peso negativo). Una regla de tipo no lineal, de uso más limitado, es la siguiente:

$$h_i(t) = \sum_{j1\,j2\ldots\,jp} w_{ij1\,j2\ldots\,jp}\, x_{j1} x_{j2} \ldots x_{jp}$$

Fórmula 1.5

que implica una interacción de tipo multiplicativo entre las entradas de la neurona. El uso de esta última regla de propagación determina que una neurona se denomine de orden superior o neurona sigma-pi, e implica una mayor complejidad, tanto en el estudio de la dinámica de la red neuronal, como en la realización de su hardware.

Función de activación o función de transferencia

La función de activación o de transferencia proporciona el estado de activación actual $a_i(t)$ a partir del potencial postsináptico $h_i(t)$ y del propio estado de activación anterior $a_i(t-1)$.

$$a_i(t) = f_i(a_i(t-1), h_i(t))$$

Fórmula 1.6

La función de activación se suele considerar determinista, y en la mayor parte de los modelos es monótona creciente y continua. La forma $y = f(x)$ de las funciones de activación más empleadas en los ANS se muestra en la tabla 1.1. Para abreviar, en ella designamos con x al potencial postsináptico, y con y al estado de activación. La más simple de todas es la función identidad, empleada por la Adaline. Otro caso también muy simple es la función escalón, empleada en el Perceptrón Simple y en la red de Hopfield discreta, así como en la neurona clásica de McCulloch-Pitts. La función lineal a tramos se puede considerar como una lineal saturada en sus extremos, es de gran sencillez computacional y resulta más meritorio desde un punto de vista biológico pues, como se ha explicado, las neuronas se activan más a mayor excitación, hasta saturarse a la máxima respuesta que pueden proporcionar.

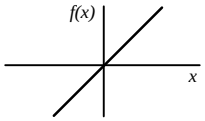
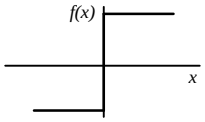
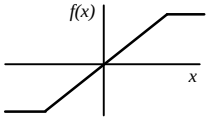
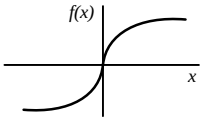
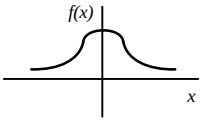
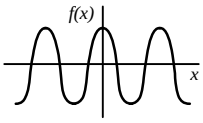
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty,+\infty]$	
Escalón	$y = signo(x)$ $y = H(x)$	$\{-1,+1\}$ $\{0,+1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si \quad x < -1 \\ x, & si \quad +1 \le x \le -1 \\ +1, & si \quad x > +1 \end{cases}$	$[-1,+1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$\{0,+1\}$ $\{-1,+1\}$	
Gaussiana	$y = A.e^{-Bx^2}$	$\{0,+1\}$	
Sinusoidal	$y = A.sen(\omega x + \varphi)$	$[-1,+1]$	

Tabla 1.1 Funciones de activación habituales.

Función de salida

Esta función proporciona la salida global de la neurona $y_i(t)$ en función de su estado de activación actual $a_i(t)$. Muy frecuentemente la función de salida es simplemente la identidad $F(x)=x$, de modo que el estado de activación de la neurona se considera como la propia salida.

$$y_i(t) = F_i(a_i(t)) = a_i(t)$$

Fórmula 1.7

1.3.2 Modelo estándar de neurona artificial

El modelo de neurona expuesto en la sección anterior resulta muy general. En la práctica suele utilizarse uno más simple, que denominaremos neurona estándar, considerando que la regla de propagación es la suma ponderada y que la función de salida es la identidad. De esta forma, la neurona estándar consiste en:

- Un conjunto de **entradas** $x_j(t)$ y pesos sinápticos w_{ij}.
- Una **regla de propagación** $h_i(t) = \sigma(w_{ij}, x_j(t))$; $h_i(t) = \sum w_{ij}x_j$ es la más común.
- Una **función de activación** $y_i(t) = f_i(h_i(t))$, que representa simultáneamente la salida de la neurona y su estado de activación.

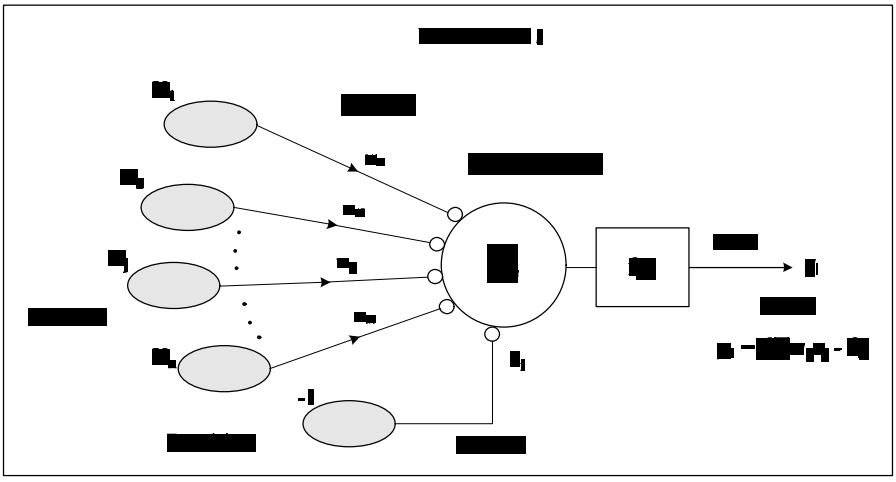


Figura 1.5 Modelo de neurona estándar

Con frecuencia se añade al conjunto de pesos de la neurona un parámetro adicional θ_i, que denominaremos umbral, que se resta del potencial postsináptico, por lo que el argumento de la función de activación queda:

$$\sum_j w_{ij} x_j - \theta_i$$

Fórmula 1.8

En conclusión, el modelo de neurona que denominaremos estándar queda:

$$y_i(t) = f_i(\sum_j w_{ij} x_j - \theta_i)$$

Fórmula 1.9

Ahora bien, si hacemos que los índices i y j comiencen en 0, podemos definir $w_{i0} \equiv \theta_i$ y $x_0 \equiv -1$ (constante), con lo que el potencial postsináptico se obtiene realizando la suma desde j=0.

$$y_i(t) = f_i(\sum_{j=0}^{n} w_{ij} x_j)$$

Fórmula 1.10

1.4 ARQUITECTURAS DE REDES NEURONALES

Definiciones básicas. Tipos de arquitecturas

En un ANS los nodos se conectan por medio de sinapsis, que determinan el comportamiento de la red. A esta estructura se conoce también como conexiones sinápticas que son direccionales, es decir, la información solamente puede propagarse en un único sentido (desde la neurona presináptica a la postsináptica).

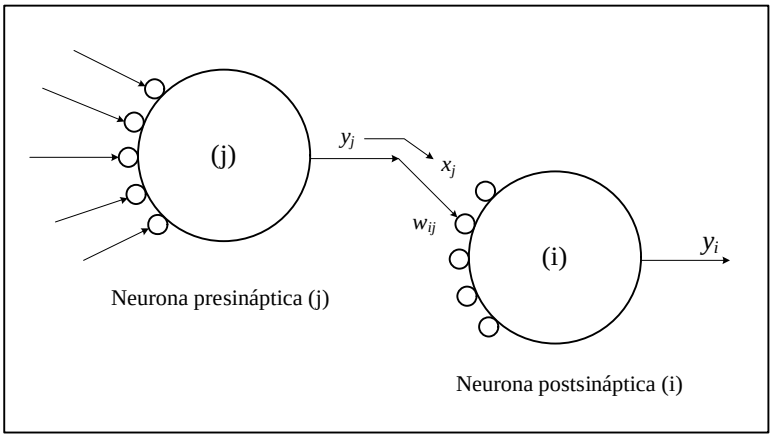


Figura 1.6 Interacción entre una neurona presináptica y otra postsináptica.

Las neuronas se suelen agrupar en unidades estructurales denominadas capas. Las neuronas de una capa pueden agruparse, formando grupos neuronales o clúster. Dentro de un grupo, o de una capa si no existe este tipo de agrupación, las neuronas suelen ser del mismo tipo. Finalmente, el conjunto de una o más capas constituye la red neuronal. Se distinguen tres tipos de capas: de entrada, de salida y ocultas.

- **Capa de entrada:** o sensorial, está compuesta por neuronas que reciben datos o señales procedentes del entorno (por ejemplo, proporcionados por sensores).

- **Capa de salida:** es aquella cuyas neuronas proporcionan la respuesta de la red neuronal.

- **Capa oculta:** es aquella que no tiene una conexión directa con el entorno, es decir, que no se conecta directamente ni a órganos sensores ni a efectores. Este tipo de capa proporciona a la red neuronal grados de libertad adicionales, gracias a los cuales puede encontrar representaciones internas correspondientes a determinados rasgos del entorno, proporcionando una mayor riqueza computacional.

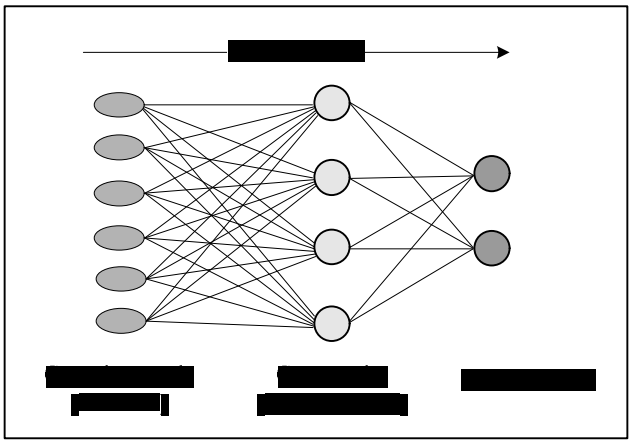


Figura 1.7 Arquitectura unidireccional de tres capas.

Las conexiones entre las neuronas pueden ser excitatorias o inhibitorias; un peso sináptico negativo define una conexión inhibitoria, mientras que uno positivo determina una conexión excitatoria. Tenemos también conexiones intra-capa e inter-capa. Las conexiones intra-capa o laterales, tienen lugar entre las neuronas pertenecientes a una misma capa, mientras que las conexiones inter-capa se producen entre las neuronas de diferentes capas. Existen además conexiones realimentadas, que tienen un sentido contrario al de entrada-salida. En algunos casos puede existir realimentación incluso de una neurona consigo misma.

Existen diferentes tipos de arquitecturas neuronales. En función a su estructura en capas, podemos hablar de:

- **Redes monocapa:** son aquellas compuestas por una única capa de neuronas.

- **Redes multicapa:** son aquellas cuyas neuronas se organizan en varias capas.

  En base al flujo de datos en la red neuronal, podemos hablar de:

- **Redes unidireccionales:** son aquellas en las que la información circula en un único sentido, desde las neuronas de entrada hacia las de salida.

- **Redes recurrentes o realimentadas:** la información puede circular entre las capas en cualquier sentido, incluido el de salida-entrada.

Por último, tenemos también las redes autoasociativas y heteroasociativas. Con frecuencia se interpreta la operación de una red neuronal como la de una memoria asociativa, que ante un determinado patrón de entradas responde con un cierto patrón de salida. Si una red se entrena para que ante la presentación de un patrón A responda con otro diferente B, se dice que la red es heteroasociativa. Si una red es entrenada para que asocie un patrón A consigo mismo, se dice que es autoasociativa (el interés de este tipo de redes, como es el caso de la de Hopfield, reside en que ante la presentación del patrón A^1 = A+ ruido, su respuesta sea el patrón original A, eliminando así el ruido presente en la señal de entrada).

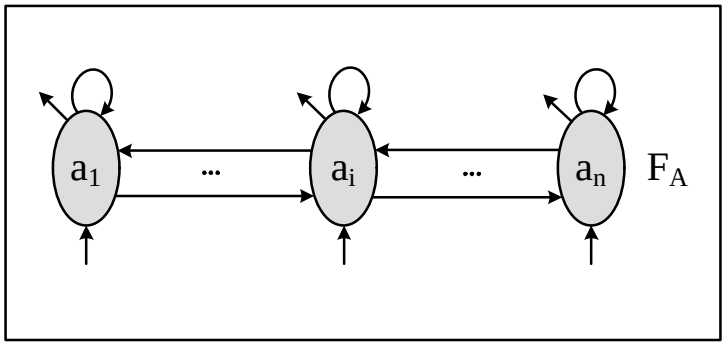


Figura 1.8 Arquitectura Monocapa y realimentada

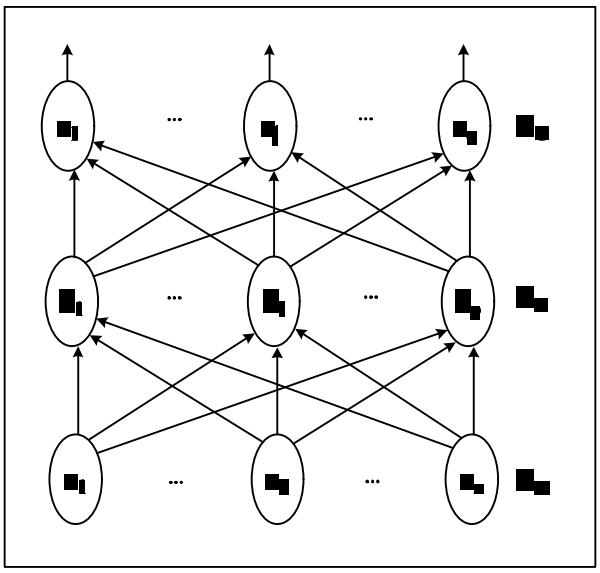


Figura 1.9 Arquitectura Multicapa y unidireccional

Dinámica de la actualización del estado de las neuronas

Dinámica síncrona: los estados se actualizan en función de un cierto reloj común. El proceso se realiza por capas, lo que significa que todas las neuronas pertenecientes a una misma capa se actualizan a la vez, comenzando desde la capa de entrada y continuando hasta la de salida. Ésta es la dinámica más habitual.

Dinámica asíncrona: no existe un reloj común, de manera que cada neurona actualiza su estado sin atender a cuándo lo hacen las demás. Es éste el tipo de dinámica presente en los sistemas neuronales biológicos.

1.5 MODOS DE OPERACIÓN: RECUERDO Y APRENDIZAJE

Se distinguen dos modos de operación en los sistemas neuronales: el modo recuerdo o ejecución, y el modo aprendizaje o entrenamiento. Este último es de particular interés,

pues una característica fundamental de los ANS es que se trata de sistemas entrenables, capaces de realizar un determinado tipo de procesamiento o cómputo aprendiéndolo a partir de un conjunto de patrones de aprendizaje o ejemplos.

Fase de aprendizaje. Convergencia

En el contexto de las redes neuronales puede definirse el aprendizaje como el proceso por el que se produce el ajuste de los parámetros libres de la red a partir de un proceso de estimulación por el entorno que rodea la red. El tipo de aprendizaje vendrá determinado por la forma en la que dichos parámetros son adaptados. En la mayor parte de las ocasiones el aprendizaje consiste simplemente en determinar un conjunto de pesos sinápticos que permita a la red realizar correctamente el tipo de procesamiento deseado.
Cuando se construye un sistema neuronal, se parte de un cierto modelo de neurona y de una determinada arquitectura de red, estableciéndose los pesos sinápticos iniciales como nulos o aleatorios. Para que la red resulte operativa es necesario entrenarla, lo que constituye el modo aprendizaje. El entrenamiento o aprendizaje se puede llevar a cabo a dos niveles. El más convencional es el de modelado de las sinapsis, que consiste en modificar los pesos sinápticos siguiendo una cierta regla de aprendizaje, construida normalmente a partir de la optimización de una función de error o coste, que mide la eficacia actual de la operación la red. Si denominamos $w_{ij}(t)$ al peso que conecta la neurona presináptica *j* con la postsináptica *i* en la iteración *t*, el algoritmo de aprendizaje, en función de las señales que en el instante *t* llegan procedentes del entorno, proporcionará el valor $\Delta w_{ij}(t)$ que da la modificación que se debe incorporar en dicho peso, el cual quedará actualizado de la forma:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

El proceso de aprendizaje es usualmente iterativo, actualizándose los pesos de la manera anterior, una y otra vez, hasta que la red neuronal alcanza el rendimiento deseado.

Algunos modelos neuronales incluyen otro nivel en el aprendizaje, la creación o destrucción de neuronas, en el cual se modifica la propia arquitectura de la red.

Tipos de aprendizaje

Los dos tipos básicos de aprendizaje son el supervisado y el no supervisado, cuya distinción proviene en origen del campo del reconocimiento de patrones. Además de las dos formas básicas anteriores pueden distinguirse el aprendizaje híbrido y el reforzado.

a) **Aprendizaje supervisado: s**e presenta a la red un conjunto de patrones, junto con la salida deseada, e iterativamente ésta ajusta sus pesos hasta que su salida tiende a ser la deseada, utilizando para ello información detallada del error que comete en cada paso. De este modo, la red es capaz de estimar relaciones entrada/salida sin necesidad de proponer una cierta forma funcional de partida. Las reglas de aprendizaje supervisadas suelen ser computacionalmente más complejas, pero también más exactos sus resultados.

b) **Aprendizaje no supervisado o autoorganizado:** se puede describir genéricamente como la estimación de la función densidad de probabilidad *p(x)* que describe la distribución de patrones *x* pertenecientes al espacio de entrada κ^n a partir de muestras. En este tipo de aprendizaje se presentan a la red multitud de patrones sin adjuntar la respuesta que deseamos. La red, por medio de la regla de aprendizaje, estima *p(x)*, a partir de lo cual pueden reconocerse regularidades en el conjunto de entradas, extraer rasgos, o agrupar patrones según su similitud.

c) **Aprendizaje híbrido:** en este caso, coexisten en la red los dos tipos básicos de aprendizaje, el supervisado y el no supervisado, los cuales tienen lugar normalmente en distintas capas de neuronas.

d) **Aprendizaje reforzado**: se sitúa a medio camino entre el supervisado y el autoorganizado. Como en el primero de los citados, se emplea información sobre el error cometido, pero en este caso existe una única señal de error, que representa un índice global del rendimiento de la red. Como en el caso del no supervisado, no se suministra explícitamente la salida deseada. En ocasiones se denomina aprendizaje por premio-castigo.

Muchos de los algoritmos de aprendizaje (aunque no todos) se basan en métodos numéricos iterativos que tratan de minimizar una función coste, lo que puede dar lugar en ocasiones a problemas en la convergencia del algoritmo. Estos aspectos no pueden abordarse de un modo general, sino que deben ser estudiados para cada algoritmo concreto. La convergencia es una manera de comprobar si una determinada arquitectura, junto a su regla de aprendizaje, es capaz de resolver un problema, pues el grado de error que se mide durante el proceso de aprendizaje describe la precisión del ajuste del mapping.

En el proceso de entrenamiento es importante distinguir entre el nivel de error alcanzado al final de la fase de aprendizaje para el conjunto de datos de entrenamiento, y el error que la red ya entrenada comete ante patrones no utilizados en el aprendizaje, lo cual mide la capacidad de generalización de la red. Interesa más una buena generalización que un error muy pequeño en el entrenamiento, pues ello indicará que la red ha capturado correctamente el mapping subyacente en los datos.

Fase de recuerdo o ejecución. Estabilidad

Generalmente, una vez que el sistema ha sido entrenado, los pesos y la estructura quedan fijos, estando la red neuronal dispuesta para procesar datos. Este modo de operación se denomina modo recuerdo o de ejecución.

1.6 COMPUTABILIDAD NEURONAL

Al igual que los computadores digitales convencionales, las redes neuronales son capaces de resolver cualquier problema computacional ya que una red neuronal es capaz de implementar cualquier función booleana. Así, cualquier circuito digital puede ser realizado con una red neuronal, sólo basta con escribirlo en función de puertas NAND, para a continuación sustituir cada puerta por una neurona como la anterior. Por otra parte, toda función booleana puede realizarse con puertas NAND, y toda puerta NAND tiene su neurona equivalente. La conclusión es que toda computación puede ser realizada por una red de neuronas.

1.7 REALIZACIÓN Y APLICACIONES DE LOS ANS

Realización de redes neuronales

El modo más habitual de realizar una red neuronal consiste en simularla en un ordenador convencional, como un PC o una estación de trabajo, haciendo uso de programas escritos en lenguajes de alto nivel, como C o Pascal. Aunque de esta manera se pierde su capacidad de cálculo en paralelo, las prestaciones que ofrecen los ordenadores actuales resultan suficientes para resolver numerosos problemas prácticos, permitiendo la simulación de redes de tamaño considerable a una velocidad razonable. Esta constituye la manera más barata y directa de realizar una red neuronal.

En el resto de las maneras de realizar un ANS se trata de aprovechar, en mayor o menor medida, su estructura de cálculo paralelo. Un paso adelante en este sentido consiste en simular la red sobre computadores con capacidad de cálculo paralelo (sistemas multiprocesador, máquinas vectoriales, masivamente paralelas).

Una orientación diferente consiste en llevar a cabo la emulación hardware de la red neuronal, mediante el empleo de sistemas de cálculo expresamente diseñados para realizar ANS basados o bien en microprocesadores de altas prestaciones o en procesadores especialmente diseñados para el trabajo con redes neuronales. Estas estructuras se suelen denominar placas aceleradoras, neuroemuladores o neurocomputadores de propósito general.

El aprovechamiento a fondo de la capacidad de cálculo masivamente paralelo de los ANS conduce a la realización hardware de la estructura de la red neuronal, en forma de circuitos específicos que reflejan con cierta fidelidad la arquitectura de la red. La tecnología más habitualmente empleada para ello es la microelectrónica VLSI, denominándose chips neuronales a los circuitos integrados así construidos. La realización hardware de la red neuronal es la manera de resolver problemas que involucran un gran número de datos y precisan respuestas en tiempo real (por ejemplo, un sistema de detección e interceptación de misiles enemigos).

Aplicaciones de las redes neuronales

Las redes neuronales se utilizan en la resolución de problemas prácticos concretos, que normalmente no han sido bien resueltos mediante sistemas tradicionales, como pueda ser el caso del reconocimiento de vehículos en los peajes de las autopistas o la previsión de consumo eléctrico. De entre los numerosos campos de aplicación de los ANS los más habituales son los relacionados con clasificación, estimación funcional y optimización; en general, el de reconocimiento de patrones suele considerarse como un denominador común. Se pueden señalar, entre otras, las siguientes áreas de aplicación de los sistemas neuronales: reconocimiento del habla, reconocimiento de caracteres, visión, robótica, control, procesamiento de señal, predicción, economía, defensa, bioingeniería, etc.

A continuación presentamos algunos casos de aplicación de ANS a problemas reales.

- Reconocimiento de caracteres: es uno de los campos donde mayores éxitos han cosechado estos sistemas; se estima que aproximadamente el 50% de los sistemas de OCR (Óptical Carácter Recognition) se basa en redes neuronales. Por ejemplo, Synaptics, empresa del Silicon Valley, ha desarrollado un chip neuronal para el reconocimiento de direcciones escritas en los sobres de las cartas.

- Control de procesos industriales: Citroën emplea redes neuronales en la determinación de la calidad del material utilizado en la confección de los asientos de los vehículos. Ford en reducción de contaminantes y Renault para detectar averías en el encendido de los automóviles.

- Finalmente, en el avión de combate F-15 se ha ensayado un sistema neuronal para ayudar al piloto en caso de ser alcanzado por fuego enemigo (aprovechando que una red neuronal electrónica puede aprender a desenvolverse en las nuevas circunstancias que influyen en el pilotaje del avión miles de veces más rápidamente que el ser humano), y recientemente una red neuronal ha conseguido aterrizar un avión Jumbo sin intervención humana.

1.8 CONCLUSION

Los ANS, como los ordenadores convencionales, son máquinas universales, por lo que para resolver un determinado problema, cualquiera de las dos aproximaciones sería perfectamente válida, procesamiento neuronal o convencional. La cuestión que entonces surge es, dado un problema, cuál de las dos alternativas resulta más eficiente en su resolución. Un ordenador digital resulta más eficiente en la ejecución de tareas aritméticas y lógicas, mientras que un ANS resolverá mejor problemas que deban tratar

con grandes bases de datos que almacenen enormes cantidades de información, y en los que existan muchos casos particulares, como sucede en los problemas de reconocimiento de patrones en ambiente natural. De esta manera podemos concluir que un estilo de computación no es mejor que el otro, simplemente para cada problema particular se deberá elegir el método más adecuado, y en el caso de problemas muy complejos, éstos deberían ser separados en partes, para resolver cada una mediante el método más idóneo.

CAPITULO 2
REDES NEURONALES SUPERVISADAS

En este capítulo estudiaremos las redes unidireccionales organizadas en capas con aprendizaje supervisado, que son utilizadas como clasificadores de patrones y estimadores de funciones. Dentro de este grupo de redes, trataremos el perceptrón simple, la adaline y el perceptrón multicapa. El popular algoritmo de aprendizaje denominado back-propagation o BP que se aplica a este último modelo.

2.1 REDES UNIDIRECCIONALES

Muchos problemas del mundo real pueden interpretarse desde el punto de vista de la estimación o aproximación funcional, en el sentido de tratar de encontrar la función que a partir de un conjunto de entradas proporciona la salida deseada. Por ejemplo, si queremos desarrollar un reconocedor de caracteres manuscritos el objetivo será encontrar un sistema que implemente la función que asocia la imagen de una determinada letra o carácter escrito con la clase a la que pertenece.

2.2 EL ASOCIADOR LINEAL: APRENDIZAJE HEBBIANO

El asociador lineal consta únicamente de una capa de neuronas lineales, cuyas entradas las denotamos por x y sus salidas por y, vector que constituye además la respuesta de la red neuronal. Asimismo, denotaremos por $W = \{w_{ij}\}$ a la matriz de pesos sinápticos; cada fila de W contiene los pesos de una neurona w_i. La operación del asociador lineal es:

$$y_i = \sum_{j=1}^{n} w_{ij} x_j$$

Por lo tanto, cada neurona *i* del asociador lineal lleva a cabo la suma ponderada de las entradas con sus pesos sinápticos. Es decir, esta neurona calcula el potencial postsináptico por medio de la convencional suma ponderada, cantidad a la que aplica finalmente una función de activación de tipo identidad.

El asociador lineal debe aprender a asociar *p* pares entrada-salida, $\{(x^{\mu}, t^{\mu}) / 1 \leq \mu \leq p\}$ ajustando sus pesos *W* de modo que ante un cierto patrón de entrada x^{μ} responda con t^{μ}, y que ante entradas similares, $(x^{\mu} + \varepsilon)$, responda con salidas también próximas $(t^{\mu} + \delta)$ (con ε y δ cantidades pequeñas). El problema se centra en encontrar matriz de pesos *W* óptima en el sentido descrito. Para ello, en el campo de las redes neuronales normalmente se hace uso de una regla de aprendizaje, que a partir de las entradas y de las salidas deseadas, proporcione el conjunto óptimo de pesos *W*.

Regla de aprendizaje de Hebb

Es uno de los modelos clásicos de aprendizaje en redes neuronales. Donald Hebb en 1949 postuló un mecanismo de aprendizaje para la neurona biológica, cuya idea básica consiste en que cuando un axón presináptico causa la activación de cierta neurona postsináptica, la eficacia de la sinapsis que las relaciona se refuerza.

De una manera general, se denomina aprendizaje hebbiano a aquellas formas de aprendizaje que involucran una modificación en los pesos Δw_{ij} proporcional al producto de una entrada *j* por la salida *i* de la neurona.

$$\Delta w_{ij} = \varepsilon y_i x_j$$

siendo ε un parámetro denominado ritmo de aprendizaje, que suele ser una cantidad entre 0 y 1. Esta expresión puede considerarse la representación matemática del modelo de aprendizaje descrito por Hebb.

Consideremos nuestro asociador lineal. La regla de Hebb se expresa en este caso particular así:

$$\Delta w_{ij}^{\mu} = t_i^{\mu} x_j^{\mu}$$

Fórmula 2.1

Y, por lo tanto:

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}^{\mu}$$

Fórmula 2.2

Si los pesos de partida son nulos, el valor final de W para las p asociaciones será:

$$W = t^1 x^{1T} + t^2 x^{2T} + \ldots + t^p x^{pT}$$

Fórmula 2.3

Empleando la regla de Hebb para el entrenamiento del asociador lineal, si los vectores de entrada $\{x^1, x^2, \ldots, x^p\}$ son ortonormales (ortogonales y de longitud unidad) se cumple:

$$Wx^{\mu} = (t^1 x^{1T} + \ldots + t^p x^{pT}) x^{\mu} = t^1 (x^{1T} . x^{\mu}) + \ldots + t^p (x^{pT} . x^{\mu}) = t^{\mu}$$

Fórmula 2.4

y por tanto, ante la entrada x^{μ} se reproduce la respuesta aprendida t^{μ}, es decir, la regla de Hebb ha conseguido en este caso que la red aprenda a realizar las asociaciones deseadas.

El problema reside en que las condiciones son muy restrictivas, pues para que las asociaciones sean correctas los patrones de entrada deben ser ortonormales. Por ello, si la dimensión del espacio de entrada es *n*, solamente podrá aprender hasta *n* asociaciones. Para almacenar más pares entrada-salida será preciso utilizar otras estrategias.

Eliminando la condición de ortogonalidad, se tiene (manteniendo el requisito de vectores de longitud 1):

$$Wx^{\mu} = t^{1}(x^{1T}.x^{\mu}) + t^{2}(x^{2T}.x^{\mu}) + \ldots + t^{p}(x^{pT}.x^{\mu}) = t^{\mu} + \sum_{\nu \neq \mu} t^{\nu}(x^{\nu T}.x^{\mu}) = t^{\mu} + \delta$$

Fórmula 2.5

expresión denominada expansión señal-ruido, pues proporciona la salida deseada más un término adicional, que interpretamos como el ruido superpuesto a la señal. Empleando reglas algo más sofisticadas que la de Hebb, como la de la pseudoinversa o la de Widrow-Hoff, se obtendrá una matriz de pesos que logrará además que el ruido δ sea pequeño comparado con la señal.

Regla de la pseudoinversa

El aprendizaje usualmente se planteará como un procedimiento para alcanzar el conjunto de pesos óptimo que resuelva un problema dado. Para ello hay que proponer un criterio que mida el rendimiento de la red neuronal para encontrar una regla de actualización de pesos que lo optimice.

Así como con la regla de Hebb se podían almacenar hasta *n* vectores ortonormales, con la pseudoinversa se pueden almacenar hasta *n* vectores linealmente independientes. Si pretendemos almacenar más pares entrada-salida, surgirán errores, pero el mapping

lineal implementado seguirá siendo óptimo en el sentido del error cuadrático medio (se alcanzará el menor error posible).

Habitualmente para el cálculo de la pseudoinversa se utiliza el teorema de Greville, aunque presenta el inconveniente de que para aprender un nuevo patrón se debe recalcular toda la matriz de pesos, lo que no resulta común dentro de la filosofía de los ANS, en la que tiende a que todos los modelos sean locales y operen incrementalmente. Se ha mostrado que en la práctica la siguiente forma aproximada del teorema de Greville, local e iterativa, suele proporcionar resultados correctos:

$$w_i^{new} = w_i^{old} + \varepsilon (t_i^{\mu} - (w_i^{old})^T x^{\mu}) x^{\mu}$$

Fórmula 2.6

siendo ε el ritmo de aprendizaje, ($0 < \varepsilon < 1$). En este esquema iterativo, los patrones deben ser presentados a la red neuronal repetidamente, obteniéndose de esta manera una aproximación a la matriz pseudoinversa mediante cálculos simples y locales.

2.3 EL PERCEPTRON SIMPLE

Este modelo neuronal fue introducido por Rosenblatt a finales de los años cincuenta. La estructura del perceptrón se inspira en las primeras etapas de procesamiento de los sistemas sensoriales de los animales (por ejemplo el de visión), en los cuales la información va atravesando sucesivas capas de neuronas, que realizan un procesamiento progresivamente de más alto nivel.

El perceptrón simple es un modelo unidireccional, compuesto por dos capas de neuronas, una sensorial o de entradas, y otra de salida. La operación de una red de este tipo, con *n* neuronas de entrada y *m* de salida, se puede expresar como:

$$y_i(t) = f(\sum_{j=1}^{n} w_{ij} x_j - \theta_i), \forall i \ , \ 1 \leq i \leq m$$

Fórmula 2.7

Las neuronas de entrada no realizan ningún cómputo, únicamente envían la información a las neuronas de salida. La función de activación de las neuronas de la capa de salida es de tipo escalón. Así, la operación de un perceptrón simple puede escribirse:

$$y_i = H(\sum_{j=1}^{n} w_{ij} x_j - \theta_i), \forall i \ , \ 1 \leq i \leq m$$

Fórmula 2.8

con *H(.)* la función de Heaviside o escalón. El perceptrón puede utilizarse tanto como clasificador, como para la representación de funciones booleanas, pues su neurona es esencialmente de tipo MacCulloch-Pitts, de salida binaria.

Mostraremos a continuación que un perceptrón permite realizar tareas de clasificación. Cada neurona del perceptrón representa una determinada clase, de modo que dado un vector de entrada, una cierta neurona responde con 0 si no pertenece a la clase que representa, y con un 1 si pertenece. Es fácil ver que una neurona tipo perceptrón solamente permite discriminar entre dos clases linealmente separables (es decir, cuyas regiones de decisión pueden ser separadas mediante una única condición lineal o hiperplano). Sea una neurona tipo perceptrón de dos entradas, x_1 y x_2 con salida *y*, cuya operación se define por lo tanto:

$$y = H(w_1 x_1 + w_2 x_2 - \theta)$$

O bien

$$y = \begin{cases} 1, si \;\; w_1 x_1 + w_2 x_2 \geq \theta \\ 0, si \;\; w_1 x_1 + w_2 x_2 < \theta \end{cases}$$

Si consideramos x_1 y x_2 situadas sobre los ejes de abscisas y ordenadas en el plano, la condición:

$$w_1 x_1 + w_2 x_2 - \theta = 0 \Rightarrow x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$

representa una recta (hiperplano, si trabajamos con *n* entradas) que divide el plano en dos regiones, aquellas para las que la neurona proporciona una salida '0' o '1', respectivamente. Luego, efectivamente, una neurona tipo perceptrón representa un discriminador lineal, al implementar una condición lineal que separa dos regiones en el espacio, que representan dos diferentes clases de patrones.

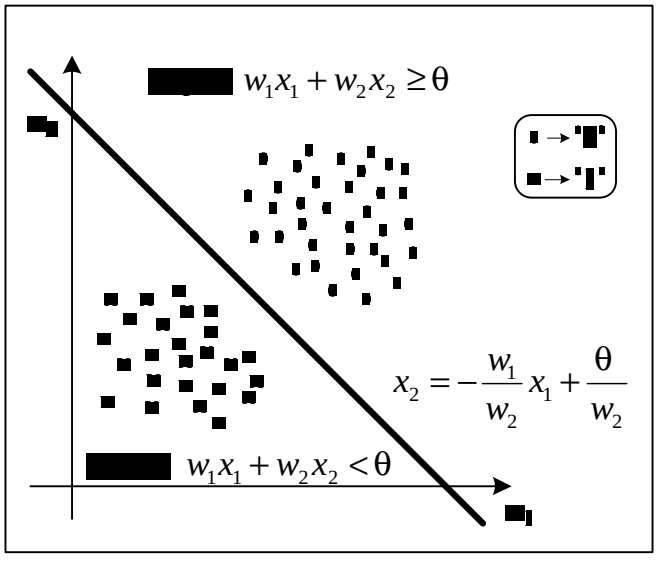


Figura 2.1 Regiones de decisión en el plano

Consideremos la función lógica NAND. En este caso pueden encontrarse unos parámetros w_1, w_2 y θ que determinen una recta que separa perfectamente las regiones correspondientes a los valores lógicos 0 y 1. Por ello, la función lógica NAND se dice

separable linealmente, puesto que hemos podido encontrar una única condición lineal que divida ambas regiones. Por ejemplo, un perceptrón con los siguientes parámetros implementa la función NAND; $w_1 = w_2 = -2$, y $\theta = -3$.

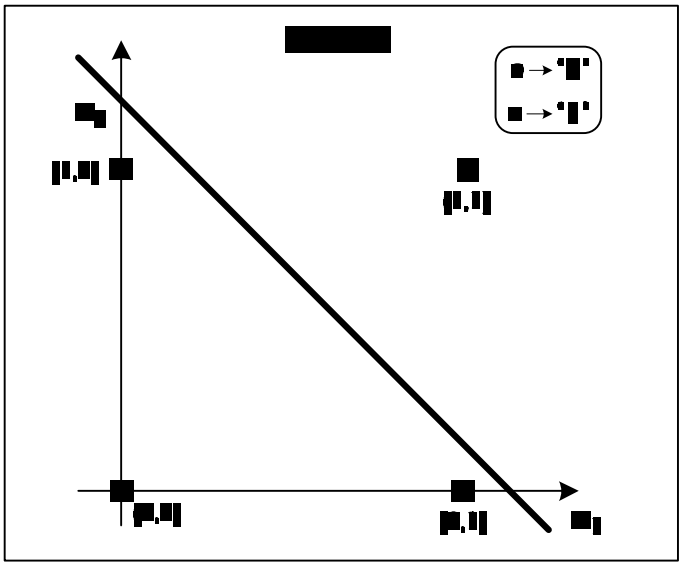


Figura 2.2 Función lógica NAND

Sin embargo, consideremos la función lógica or-exclusivo o XOR (su salida es el 0 lógico si las variables de entrada son iguales y 1 si son diferentes), y representémosla también en el plano. En este caso podemos apreciar que no se puede encontrar una única condición lineal que separe las regiones correspondientes a los valores de salida 0 y 1, por lo que se dice que la XOR no es separable linealmente. Como la neurona del perceptrón representa en el fondo un discriminador lineal, esta neurona por sí sola no puede implementar la función XOR.

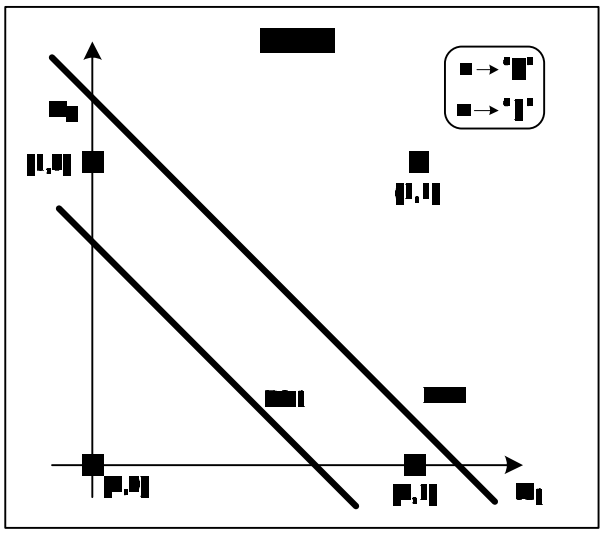


Figura 2.3 Función lógica XOR

Por lo tanto, pese a su gran interés, el perceptrón presenta serias limitaciones, pues solamente puede representar funciones linealmente separables. Así, aunque pueda aprender automáticamente a representar complejas funciones booleanas o resolver con éxito muchos problemas de clasificación, en otras ocasiones fallará estrepitosamente. Una solución a las limitaciones del perceptrón simple puede consistir en incluir más capas en la arquitectura, con lo que tendremos un perceptrón multicapa.

Los tipos de regiones de decisión que pueden formarse mediante estructuras simples y multicapa se muestran a continuación:

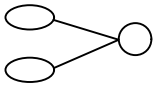
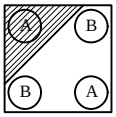
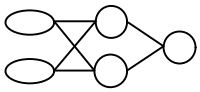
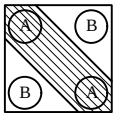
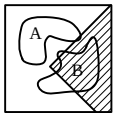
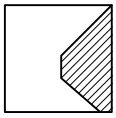
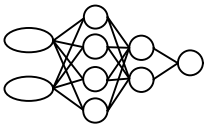
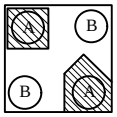
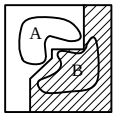
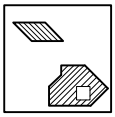
Arquitectura	Región de decisión	Ejemplo 1: XOR	Ejemplo 2: Clasificación	Regiones más generales
Sin capa oculta 	Hiperplano (dos regiones)			
Una capa oculta 	Regiones polinomiales convexas			
Dos capas ocultas 	Regiones arbitrarias			

Tabla 2.1 Tipos de regiones de decisión en el perceptrón.

El problema a resolver en este caso, es la discriminación entre dos clases de patrones, la clase A y la B. Los contornos que se forman con estructuras multicapa son polinomiales, puesto que consisten en la composición de discriminaciones lineales correspondientes a diferentes neuronas de tipo umbral. Si consideramos neuronas de respuesta continua, por ejemplo de tipo sigmoideo, los contornos serían similares, aunque sin esquinas. De la tabla 2.1 podemos apreciar que mediante una estructura de dos capas, sin capa oculta, la región de decisión es un hiperplano que separa en dos el espacio de las variables. Haciendo uso de tres capas, con una oculta, se pueden discriminar regiones convexas, sean cerradas o abiertas. Con una estructura de cuatro capas, dos de ellas ocultas, se puede discriminar regiones de forma arbitraria, cuyo único límite viene impuesto por el número de nodos empleados.

A finales de los sesenta ya se apuntaba como solución a las limitaciones del perceptrón introducir capas ocultas, pero el problema residía en que si bien se disponía de un algoritmo de aprendizaje para el perceptrón simple, no se disponía de ningún procedimiento que permitiese obtener automáticamente los pesos en uno multicapa, con neuronas ocultas. Este problema fue resuelto por Paúl Werbos, pero fue preciso esperar

hasta mediados de los años ochenta para que se redescubriera un algoritmo similar, que denominaron back-propagation o BP, y diera a conocer a la comunidad internacional su gran potencial para la resolución de problemas prácticos.

2.3.1 Algoritmo de aprendizaje del perceptrón

La importancia del perceptrón radica en su carácter de dispositivo entrenable, pues el algoritmo de aprendizaje introducido por Rosenblatt permite que el perceptrón determine automáticamente los pesos sinápticos que clasifican un determinado conjunto de patrones etiquetados.

El algoritmo de aprendizaje del perceptrón es de los denominados por corrección de errores. Los algoritmos de este tipo ajustan los pesos en proporción a la diferencia existente entre la salida actual de la red y la salida deseada, con el objetivo de minimizar el error actual de la red.

Sea un conjunto de p patrones x^{μ}, $\mu = 1,\ldots, p$, con sus salidas deseadas t^{μ}. Tanto las entradas como las salidas solamente pueden tomar los valores -1 o 1 (o bien, 0 o 1, según definamos los niveles lógicos). Se tiene una arquitectura de perceptrón simple, con pesos iniciales aleatorios, y se requiere que clasifique correctamente todos los patrones del conjunto de aprendizaje. Actuaremos del siguiente modo, ante la presentación del patrón *μ-ésimo*, si la respuesta que proporciona el perceptrón es correcta, no actualizaremos los pesos; si es incorrecta, los modificaremos según la regla de Hebb, con lo que se tiene:

$$\Delta w_{ij}^{\mu}(t) = \begin{cases} 2\varepsilon . t_i^{\mu} x_j^{\mu}, & si \quad y_i^{\mu} \neq t_i^{\mu} \\ 0, & si \quad y_i^{\mu} = t_i^{\mu} \end{cases}$$

que se puede reescribir del siguiente modo:

$$\Delta w_{ij}^{\mu}(t) = \varepsilon.(t_i^{\mu} - y_i^{\mu})x_i^{\mu}$$

Fórmula 2.9

que es la forma habitual de expresar la regla del perceptrón. En su utilización práctica, se debe llegar a un compromiso para el valor del ritmo de aprendizaje ε, puesto que un valor pequeño implica un aprendizaje lento, mientras que uno excesivamente grande puede conducir a oscilaciones en el entrenamiento, al introducir variaciones en los pesos excesivamente amplias. Al ser las entradas y las salidas discretas {-1,+1}, también lo será la actualización de los pesos que únicamente podrá tomar los valores 0 o $\pm 2\varepsilon$.

Es importante remarcar que el proceso de aprendizaje es iterativo: es parte de una configuración sináptica de partida y se presentan una y otra vez los patrones, para que los pesos se ajusten iterativamente, hasta que todos queden bien clasificados. El hiperplano que establece el límite entre dos clases se desplaza lentamente hasta conseguir separarlas por completo (si ello es posible). El ajuste de los pesos en la iteración *t* debido a todo el conjunto de aprendizaje será:

$$w_{ij}(t+1) = w_{ij}(t) + \sum_{\mu=1}^{p} \Delta w_{ij}^{\mu}(t)$$

Fórmula 2.10

Rosenblatt demostró que si la función a representar es linealmente separable, este algoritmo siempre converge en un tiempo finito y con independencia de los pesos de partida. Por otra parte, si la función no es linealmente separable, el proceso de entrenamiento oscilará. Por otro lado, el algoritmo del perceptrón se detiene tan pronto como consigue clasificar correctamente todos los ejemplos, por lo que con frecuencia la línea de discriminación queda muy cerca de las muestras de uno de los grupos. Para

obtener una discriminación óptima se han introducido algoritmos como el denominado Adatron.

2.4 ADALINE

Conocida también como Adaline, introducida por Widrow en 1959, cuyo nombre proviene de Adaptive Linear Neuron. Este modelo utiliza una neurona similar a la del perceptrón, pero de respuesta lineal, cuyas entradas pueden ser continuas. Por otra parte, a diferencia del nodo del asociador lineal, el de la adaline incorpora un parámetro adicional denominado umbral, aunque debe tenerse en cuenta que no se trata de un umbral de disparo como el del perceptrón, sino de un parámetro que proporciona un grado de libertad adicional. De este modo, la fórmula de la adaline queda:

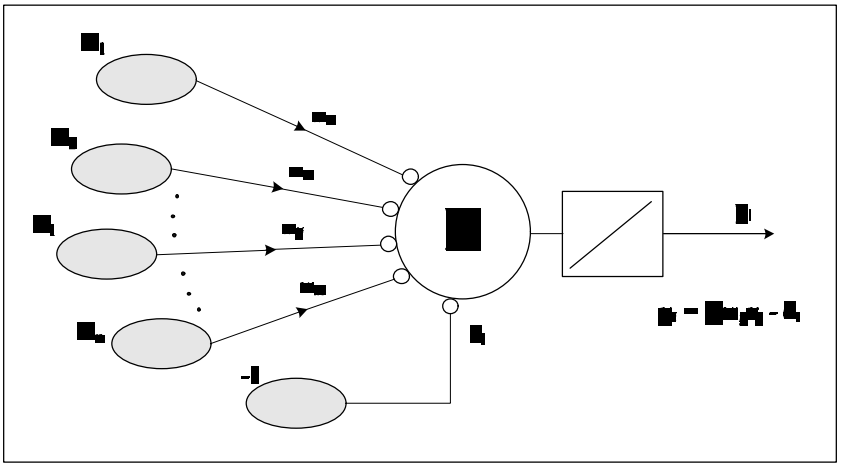


Figura 2.4 Neurona lineal de Adaline

$$y_i(t) = \sum_{j=1}^{n} w_{ij} x_j - \theta_i, \forall i \ , \ 1 \leq i \leq m$$

Fórmula 2.11

No obstante, la diferencia más importante con el perceptrón y con el asociador lineal reside en la regla de aprendizaje que implementa. En la adaline se utiliza la regla de Widrow-Hoff, también conocida como regla LMS (Least Mean Squares, mínimos cuadrados), que conduce a actualizaciones de tipo continuo, siendo la actualización de los pesos proporcional al error que la neurona comete.

Este ANS es un modelo muy conocido y ampliamente utilizado, por ejemplo, para cancelar el ruido en la transmisión de señales. De este modo, y desde hace años, millones de módems en todo el mundo incluyen una adaline. Su utilidad se ve limitada por tratarse de un sistema lineal. Así, solamente podrá separar correctamente patrones linealmente independientes, fallando en ocasiones ante patrones linealmente separables, que el perceptrón siempre discrimina. No obstante, ante patrones no separables linealmente, los resultados que proporciona son en promedio mejores que los del perceptrón, pues la adaline siempre opera reduciendo el error cuadrático medio al mínimo posible.

2.4.1 Regla LMS

La regla de Widrow-Hoff o LMS, que en un caso particular es conocida como regla delta, constituye el algoritmo de aprendizaje asociado al adaline. La regla LMS conducirá a tener asociaciones perfectas cuando sus vectores sean linealmente independientes, proporcionando cuando no lo sean una matriz de pesos óptima desde el punto de vista de los mínimos cuadrados.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} = w_{ij}(t) + \alpha (t_i^{\mu} - y_i^{\mu}) x_i^{\mu}$$

Fórmula 2.12

Ésta es la regla LMS o de Widrow-Hoff, que puede considerarse la versión iterativa de la regla de la pseudoinversa para el caso de vectores estocásticos. Si en lugar de

considerar la función activación lineal la consideramos sigmoidea, el algoritmo de aprendizaje se denomina regla delta.

Widrow y Hoff demostraron la convergencia de la fórmula 2.12 a la configuración w^*_{ij} que minimiza el valor esperado del error, a condición de que $\alpha(t)$ cumpla:

$$\sum_{t=1}^{\infty}\alpha(t)=\infty \qquad\qquad \sum_{t=1}^{\infty}\alpha^2(t)<\infty$$

por ejemplo $\alpha=\alpha(t)=t^{-1}$ satisface ambas condiciones. En muchas ocasiones es suficiente con que α tome un valor pequeño ($0<\alpha<1$). Las condiciones anteriores garantizan que el aprendizaje no se lleve a cabo ni excesivamente rápido ni muy lentamente. El sistema que aprende debe ser suficientemente estable como para recordar los patrones antiguos (un ritmo de aprendizaje excesivamente grande los borraría, pues la presente actualización sería de gran magnitud), pero suficientemente plástico como para aprender los nuevos (un ritmo muy pequeño, provoca actualizaciones diminutas, por lo que el proceso de entrenamiento se dilataría excesivamente en el tiempo). Esto constituye el denominado dilema de la plasticidad frente a la estabilidad. La primera condición asegura que el sistema sea plástico, mientras que la segunda asegura la condición de estabilidad.

2.5 EL PERCEPTRON MULTICAPA

Un perceptrón multicapa se obtiene añadiendo capas intermedias (ocultas) a un perceptrón simple, o MLP (Multi-Layer Perceptron). Esta arquitectura suele entrenarse mediante el algoritmo denominado retropropagación de errores o BP, o bien haciendo uso de alguna de sus variantes o derivados, motivo por el que en muchas ocasiones el

conjunto arquitectura MLP + aprendizaje BP suele denominarse red de retropropagación, o BP.

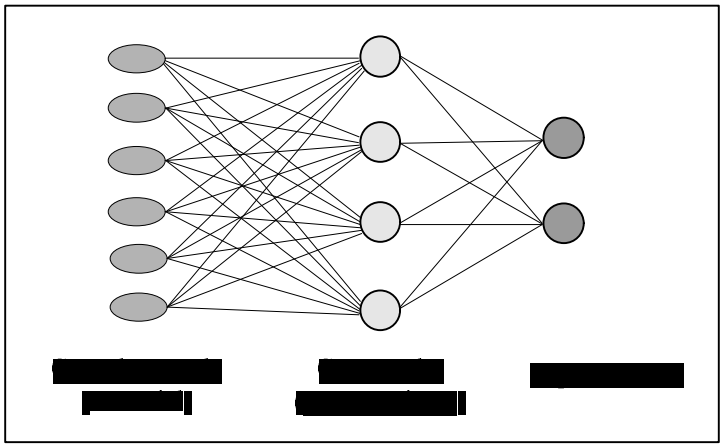


Figura 2.5 Estructura del Perceptrón multicapa

Los importantes requisitos de cómputo que el algoritmo BP precisa no podían ser satisfechos con los medios disponibles a principios de los setenta, por lo que el primer descubrimiento del BP era quizás algo prematuro. Por fin en los años ochenta los computadores eran suficientemente potentes como para permitir la aplicación del BP a problemas de interés.

Denominaremos x_i a las entradas de la red, y_i a las salidas de la capa oculta y z_k a las de la capa final; t_k serán las salidas objetivo. Por otro lado, w_{ij} son los pesos de la capa oculta y θ_j sus umbrales, w'_{kj} los pesos de la capa de salida y θ'_k sus umbrales. La operación de un MLP con una capa oculta y neuronas de salida lineal se expresa matemáticamente de la siguiente manera:

$$z_k = \sum_j w'_{kj}\, y_j - \theta'_i = \sum_j w'_{kj}\, f\left(\sum_i w_{ji} x_i - \theta_j\right) - \theta'_i$$

Fórmula 2.13

2.5.1 Aprendizaje por retropropagación de errores (BP)

Una solución al problema de entrenar los nodos de las capas oculta pertenecientes a arquitecturas multicapa la proporciona el algoritmo de retropropagación de errores o BP (backpropagation). El procedimiento para entrenar mediante BP una arquitectura MLP dada es el siguiente:

1. Establecer aleatoriamente los pesos y umbrales iniciales (*t:=0*).
2. Para cada patrón μ del conjunto de aprendizaje:
   2.1 Llevar a cabo una fase de ejecución para obtener la respuesta de la red ante el patrón *μ-ésimo.*
   2.2 Calcular las señales de error asociadas Δ'^{μ}_{k} y Δ^{μ}_{j}.
   2.3 Calcular el incremento parcial de los pesos y umbrales debidos a cada patrón μ.
3. Calcular el incremento total (para todos los patrones) actual de los pesos $\delta w'_{kj}$ y δw_{ji}. Hacer lo mismo para los umbrales.
4. Actualizar pesos y umbrales.
5. Calcular el error actual, *t:=t+1*, y volver a 2. si todavía no es satisfactorio.

Se debe comenzar siempre con pesos iniciales aleatorios (normalmente números pequeños, positivos y negativos), ya que si se parte de pesos y umbrales iniciales nulos el aprendizaje no progresará (puesto que las salidas de las neuronas y el incremento en los pesos serán siempre nulos).

En el esquema presentado, se lleva a cabo una fase de ejecución para todos y cada uno de los patrones del conjunto de entrenamiento, se calcula la variación en los pesos debida a cada patrón, se acumulan, y solamente entonces se procede a la actualización de los pesos. Este esquema se suele denominar aprendizaje por lotes.

Una variación común al algoritmo consiste en actualizar los pesos sinápticos tras la presentación de cada patrón (en vez de presentarlos todos y luego actualizar), esquema denominado aprendizaje en serie (on-line). Aunque el verdadero BP es el que se ejecuta por lotes, el aprendizaje en serie es habitualmente empleado en aquellos problemas en los que se dispone de un muy numeroso conjunto de patrones de entrenamiento, en el que habrá mucha redundancia en los datos. Si se emplease en este caso el modo por lotes, el tener que procesar todos los patrones antes de actualizar los pesos demoraría considerablemente el entrenamiento. Por ejemplo, con un conjunto de entrenamiento compuesto por 10.000 patrones, en el que cada patrón aparece repetido cien veces, entrenando por lotes el aprendizaje duraría cien veces más que en modo serie.

EL BP, sea en versión por lotes o en serie, constituye un método de gran generalidad, lo que presenta ventajas e inconvenientes. Su ventaja principal es que se puede aplicar a multitud de problemas diferentes, proporcionando con frecuencia buenas soluciones con no demasiado tiempo de desarrollo.

Como desventaja se encuentra, entre otras, su lentitud de convergencia, uno de los precios que hay que pagar por disponer de un método general de ajuste funcional que no requiere en principio información razonable. Otro problema del BP es que puede incurrir en el denominado sobreaprendizaje, fenómeno directamente relacionado con la capacidad de generalización de la red a partir de los ejemplos presentados. Por otra parte, debe tenerse en cuenta que el algoritmo BP no garantiza alcanzar el mínimo global de la función error, tan solo un mínimo local, por lo que el proceso de aprendizaje puede quedarse estancado en uno de estos mínimos locales.

2.6 CAPACIDAD DE GENERALIZACIÓN DE LA RED

Uno de los aspectos fundamentales de los ANS es su capacidad de generalizar a partir de ejemplos, lo que constituye el problema de la memorización frente a generalización. Por generalización se entiende la capacidad de la red de dar una respuesta correcta ante patrones que no han sido empleados en su entrenamiento. Una red neuronal correctamente entrenada generalizará, lo que significa que ha aprendido adecuadamente el mapping no sólo los ejemplos concretos presentados, por lo que responderá correctamente ante patrones nunca vistos con anterioridad.

2.6.1 Validación cruzada (cross-validation)

En un proceso de entrenamiento se debe considerar, por una parte, un error de aprendizaje, que se suele calcular como el error cuadrático medio de los resultados proporcionados por la red para el conjunto de patrones de aprendizaje. Con una red suficientemente grande, puede reducirse tanto como se quiera sólo con llevar a cabo más iteraciones. Por otra parte, existe un error de generalización, que se puede medir empleando un conjunto representativo de patrones diferentes a los utilizados en el entrenamiento. De esta manera, podemos entrenar una red neuronal haciendo uso de un conjunto de aprendizaje, y comprobar su eficiencia real, o error de generalización, mediante un conjunto de test.

Un hecho experimental, fácilmente observable con cualquier simulador, es que si se entrena una red hasta alcanzar un muy pequeño error en aprendizaje (por ejemplo, inferior a un 1%), la eficacia real del sistema o generalización (medido como error en test) se degrada. Si representamos a la vez el error en aprendizaje y el error test durante el transcurso del aprendizaje, se obtiene la siguiente figura:

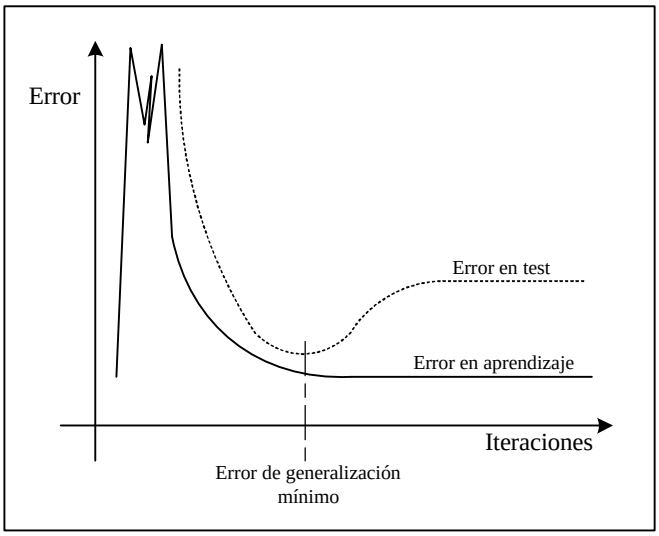


Figura 2.6 Evolución del error de aprendizaje y generalización (situación ideal)

Tras una fase inicial, en la que pueden aparecer oscilaciones en el valor del error, el de aprendizaje tiende a disminuir monótonamente, mientras que el error de generalización a partir de cierto punto comienza a incrementarse, lo que indica una degradación progresiva del aprendizaje.

La explicación de este fenómeno es la siguiente. Al principio la red se adapta progresivamente al conjunto de aprendizaje, acomodándose al problema y mejorando la generalización. Sin embargo, en un momento dado el sistema se ajusta demasiado a las particularidades de los patrones empleados en el entrenamiento, aprendiendo incluso el ruido en ellos presente, por lo que crece el error que comete ante patrones diferentes a los empleados en el entrenamiento (error de generalización). En este momento la red no ajusta correctamente el mapping, sino que simplemente está memorizando los patrones del conjunto de aprendizaje, lo que técnicamente se denomina sobreaprendizaje o sobreajuste, pues la red está aprendiendo demasiado. Idealmente, dada una arquitectura de red neuronal, ésta debería entrenarse hasta un punto óptimo en el que el error de generalización es mínimo. El procedimiento consiste en entrenar y validar a la vez para detenerse en el punto óptimo, a lo cual se denomina validación cruzada, y es

ampliamente utilizado en la fase de desarrollo de una red neuronal supervisada. No obstante, la situación descrita ha sido en cierta medida idealizada; una situación más realista sería la presentada en la figura 2.7, en donde pueden presentarse varios mínimos para el conjunto de test, debiendo detener el aprendizaje en el punto óptimo de mínimo error de generalización, y no quedarnos en el primer mínimo en test que aparezca. Existen algunas técnicas de parada temprana, aunque muchas veces basta con dejar que el aprendizaje discurra hasta una cota razonable de error (0.5%, 0.1%, 0.01%, etc., depende del problema), guardando periódicamente las distintas configuraciones intermedias de pesos, para luego quedarnos con la que proporcionó un error en test mínimo.

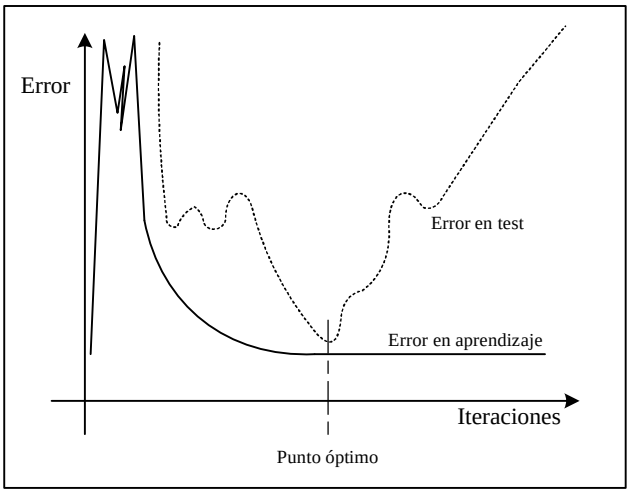


Figura 2.7 Evolución del error de aprendizaje y generalización (situación real)

La clave de este asunto está en que las redes neuronales son estimadores lineales poderosos, capaces de modelar situaciones muy complejas. En herramientas lineales la complejidad del modelo viene dada simplemente por el número de parámetros libres a ajustar, mientras que en los sistemas no lineales (como son las redes neuronales), la complejidad del modelo depende tanto del numero de parámetros como de su valor actual. Los modelos que implementan las redes neuronales son de elevada complejidad, por lo que pueden aprender casi cualquier cosa, motivo por el cual incurren fácilmente en sobreaprendizaje.

Hablando en términos generales, si el problema es sencillo, bastarán pocos parámetros para su ajuste, luego deberá utilizarse una red pequeña. Si el problema es complejo se necesitarán más parámetros de ajuste, luego se necesitará una red de mayor tamaño. Por lo tanto, debe ajustarse el tamaño de la red a la complejidad del problema que se está tratando, debiéndose limitar en lo posible su tamaño, dificultando así la aparición de sobreentrenamiento.

Cuando se entrena una red unidireccional supervisada debe tenerse muy en cuenta todo esto, y la técnica de validación cruzada suele ser un buen remedio; usualmente, de todo el conjunto de entrenamiento se emplea aproximadamente un 80% de los patrones para entrenar, reservándose un 20% como conjunto de test.

Número de ejemplos de entrenamiento

La capacidad de generalización de la red la determinan en buena medida las siguientes tres circunstancias:

1) La arquitectura de la red.

2) El número de ejemplos de entrenamiento

3) La complejidad del problema.

Los tres puntos están muy relacionados; en términos generales, cuanto más complejo sea el problema a modelar, más grande deberá ser la red y, por lo tanto, más ejemplos se necesitarán para entrenarla contemplando todas las situaciones posibles.

A menudo el número de patrones-ejemplo disponibles es limitado, y en proporción el número de parámetros efectivos de la red elegida (grados de libertad) suele ser muy grande. Así queremos que la red alcance un error de generalización de, por ejemplo, $\varepsilon = 0.1$ (un 10%), el número de patrones de aprendizaje necesarios p será del orden de $p = 10.w$, expresión que se suele dar como ejemplo del número aproximado de patrones que serían necesarios para entrenar adecuadamente una red neuronal de w pesos. Por ejemplo, para una red 10-5-1 (10 neuronas de entrada, 5 ocultas y 1 de salida.), que dispone de 61 parámetros, entre pesos y umbrales, el número de patrones necesarios para alcanzar un error del 10% será de unos 610, lo que representa una cifra de patrones muy alta, no disponible en muchas aplicaciones prácticas. Ello ilustra de nuevo la facilidad de incurrir en sobreaprendizaje al entrenar una red neuronal.

Reducción del tamaño de la arquitectura de red

Hay que tener presente la llamada maldición de la dimensionalidad, que consiste en que el número de datos necesarios para especificar un mapping, en general crece exponencialmente con la dimensión del espacio de entrada, lo que agrava en los problemas de dimensión de entrada elevada el disponer de un número de patrones para el aprendizaje escaso. Disminuyendo el número de parámetros de la red (tamaño) se tendrá una relación w/ε más favorable.

Se pueden utilizar algunas técnicas para la reducción del número de parámetros de la red; algunas bien conocidas son las de compartir pesos, podado de la red o decaimiento de pesos. En la primera de las citadas, diversas neuronas comparten sus pesos, de modo que el número total disminuye. En el proceso de podado la red es entrenada hasta un cierto nivel, para luego eliminar aquellos pesos que no aportan prácticamente nada a su operación. El decaimiento es un caso especial del podado; durante el aprendizaje se deja a los pesos tender poco a poco a cero, para que aquellos que no sean actualizados periódicamente, se anulen y desaparezcan.

2.7 APLICACIÓN PRÁCTICA

A continuación presentamos la interfaz de la aplicación que muestra el funcionamiento de la Adalina.

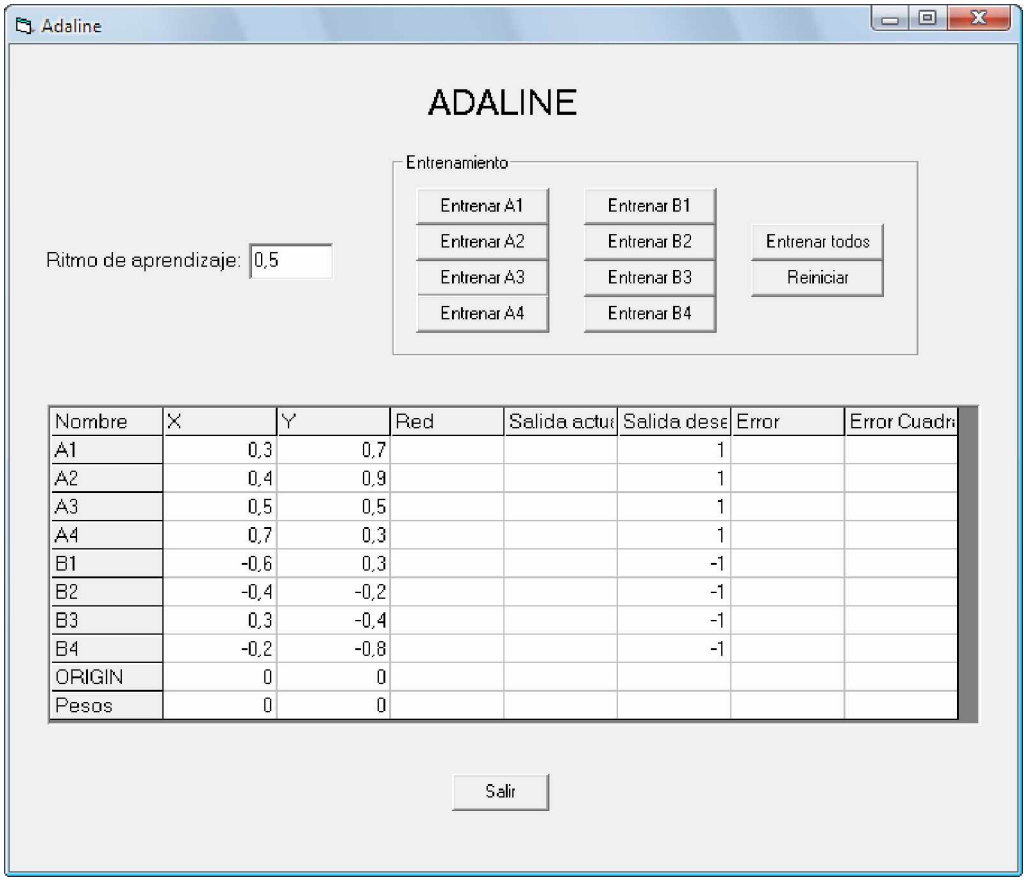


Figura 2.8 Interfaz de la aplicación práctica que demuestra el funcionamiento de la Adalina

En la figura anterior tenemos los siguientes elementos:

- Ritmo de aprendizaje que debe ser un valor entre 0 y 1.

- Una sección en la cual se encuentran los diferentes botones para el entrenamiento de la neurona respectiva.

- Dentro de la tabla que se encuentra en la pantalla, tenemos la entradas representadas por *X* y *Y*, el valor de la red, la salida actual (luego del entrenamiento), la salida deseada, el error cometido y por último el error cuadrado. Al lado izquierdo de la tabla tenemos las neuronas con los pesos que se manejarán en el entrenamiento.

2.8 CONCLUSION

A lo largo de este capítulo hemos analizado las redes unidireccionales dentro de las cuales estudiamos el perceptrón simple que permite realizar tareas de clasificación; el perceptrón multicapa que adhiere capas ocultas dentro de su estructura para solucionar las limitaciones del perceptrón simple; por último se analizó la Adaline que posee un modelo similar al perceptrón pero con respuesta lineal.

CAPITULO 3
REDES AUTOORGANIZADAS

Estudiadas algunas de las redes supervisadas más importantes, en este capítulo trataremos el otro gran grupo de los modelos neuronales, los no supervisados o autoorganizados. Éstos se caracterizan porque en su entrenamiento no se presentan las salidas objetivo que se desean asociar a cada patrón de entrada. La red, a partir de un proceso de autoorganización, proporcionará cierto resultado, el cual será reflejo de las relaciones de similitud existentes entre dichos patrones de entrada. La principal aplicación de estos modelos será la realización de agrupamiento de patrones (clustering), análisis exploratorio, y visualización y minería de datos (data mining).

3.1 MODELOS NEURONALES NO SUPERVISADOS

A diferencia de lo que sucede en el aprendizaje supervisado tratado hasta el momento, en el no supervisado (autoorganizado) no existe ningún maestro externo que indique si la red neuronal está operando correcta o incorrectamente, pues no se dispone de ninguna salida objetivo hacia la cual la red neuronal deba tender. Así, durante el proceso de aprendizaje la red autoorganizada debe descubrir por sí misma rasgos comunes, regularidades, correlaciones o categorías en los datos de entrada, e incorporarlos a su estructura interna de conexiones (pesos). Se dice en este caso que las neuronas deben autoorganizarse en función de los estímulos (señales o datos) procedentes del exterior. Para obtener resultados de calidad la red requiere un cierto nivel de redundancia en las entradas procedentes del espacio sensorial, en definitiva, un número de patrones de aprendizaje suficientemente amplio.

El tipo de procesamiento que una red neuronal no supervisada puede realizar depende en buena medida de su arquitectura. A continuación algunos de ellos:

a) **Análisis de similitud entre patrones.** Este tipo de procesamiento aparece cuando existe una única neurona cuya salida es continua, indicando el grado de similitud o parecido entre el patrón de entrada actual y el promedio de los presentados en el pasado. Dicho promedio queda representado durante el entrenamiento en el vector de pesos sinápticos de la neurona, de modo que la neurona aprende lo que es típico dentro de un conjunto de patrones.

b) **Análisis de componentes principales.** Si se extiende en el caso anterior a varias neuronas de salida continua, el proceso de aprendizaje supone encontrar una cierta base del espacio de entrada, representada por el conjunto de los vectores de pesos sinápticos de todas las neuronas, que se corresponderán con los rasgos más sobresalientes del espacio sensorial.

c) **Agrupamiento/clasificación (clustering).** La red realiza tareas de agrupamiento o clasificación cuando se compone de neuronas de salida discreta {0, 1}, donde cada una representa una categoría, y solamente una de ellas puede permanecer activada a la vez. Ante un patrón de entrada, la neurona que se activa indica a qué categoría o grupo (cluster) pertenece. Durante el aprendizaje la red deduce las categorías presentes en el espacio de entrada a partir de la medida de distancias (euclídea o de otro tipo) entre los patrones presentados.

d) **Memoria asociativa.** Representa una generalización del caso anterior de redes para agrupamiento; en este caso la salida proporcionada por la neurona activada no es solamente un 0 o un 1, sino el vector prototipo de la clase en cuestión.

e) **Codificación.** Es análogo al anterior, sólo que en este caso la neurona no proporciona como salida el vector prototipo de la clase, sino una versión

codificada (por ejemplo, una etiqueta), habitualmente empleando menos bits y manteniendo la información relevante (compresión de datos).

f) **Mapas de rasgos.** Se trata de modelos no supervisados en los que las neuronas se ordenan geométricamente (por ejemplo, en forma de matriz bidimensional o mapa), llevando a cabo una proyección del espacio sensorial de entrada sobre la red (mapa), proyección que preserva en lo posible la topología del espacio original, pero reduciendo sus dimensiones.

Los casos citados no son necesariamente diferentes ni excluyentes; un mismo modelo de red no supervisada puede estar capacitado para efectuar varios de estos procesamientos. Por ejemplo, la codificación puede realizarse mediante un análisis de componentes principales o mediante clustering. Este último caso recibe el nombre de cuantificación vectorial (vector quantization). Aunque los modelos autoorganizados tienen un campo de aplicación propio, relacionado en general con el análisis exploratorio de datos, en ocasiones puede sustituir a los modelos supervisados. Esta circunstancia puede darse, por ejemplo, cuando la aplicación del aprendizaje BP a un problema que requiere aprendizaje supervisado es demasiado lenta, en cuyo caso puede resultar de utilidad el empleo de un modelo multicapa híbrido (con una capa autoorganizada y otra supervisada), pues al romper el entrenamiento en dos fases se gana en velocidad.

Tipos de redes no supervisadas

Los modelos no supervisados suelen ser simples, monocapa y con algoritmos sencillos y rápidos, más próximos a la biología. En general, se pueden clasificar en dos grandes grupos. El primero lo denominaremos redes no supervisadas hebbianas, pues su aprendizaje es de tipo Hebb. Como característica general de dichas redes puede señalarse que en ellas en número elevado de neuronas de salida puede activarse simultáneamente. Algunos modelos utilizan reglas de aprendizaje directamente basadas

en la regla de Hebb, como las redes PCA (que realizan análisis de componentes principales o Principal Component Analysis). Otros se derivan de determinadas propiedades a optimizar, como pueda ser la maximización del contenido de información. Existe otra amplia clase de modelos autoorganizados, denominados redes no supervisadas competitivas, en las que solamente una neurona (o grupo de vecinas) puede quedar finalmente activadas. La base de la operación de estos modelos es la competición entre las neuronas, materializada en forma de inhibiciones laterales, a través de las cuales cada una trata de inhibir a las demás. En este proceso de competición la neurona más activa conseguirá inhibir a todas las demás, por lo que será la única que permanezca activada, motivo por el cual estas redes también se denominan redes WTA (winner-take-all).

En los modelos competitivos, durante la fase de aprendizaje las neuronas vencedoras obtienen como premio el refuerzo de sus conexiones sinápticas. La competición es un comportamiento básico en muchos de los modelos neuronales autoorganizados más conocidos, como el ART, Neocognitrón o los mapas autoorganizados. La idea común a todos ellos es el agrupamiento o categorización de patrones, pues esencialmente desarrollan clusters que agrupan patrones. No obstante, también pueden resultar útiles para muchas otras tareas, como cuantificación vectorial, aproximación funcional, procesamiento de imagen, análisis estadístico y optimización combinatorial.

3.2 MODELO DE MAPAS AUTOORGANIZADOS

3.2.1 Introducción a los mapas autoorganizados

Se observa que en muchas regiones del córtex de los animales superiores aparecen zonas donde las neuronas detectoras de rasgos (o características) se distribuyen topológicamente ordenadas. Por ejemplo, en el área somatosensorial (Figura 3.1), las

neuronas que reciben señales de sensores que se encuentran próximos en la piel se sitúan también próximas en el córtex, de manera que reproducen (aunque distorsionadamente) el mapa de la superficie de la piel en una zona de la corteza cerebral; lo mismo sucede, por ejemplo, en el córtex motor. Por lo que respecta al sentido del oído, existen en el cerebro áreas que representan mapas tonotópicos, donde los detectores de determinados rasgos relacionados con el tono de un sonido se encuentran ordenados en dos dimensiones. En resumen, la representación de la información en la corteza cerebral aparece con frecuencia organizada espacialmente, circunstancia que el modelo neuronal de mapas de rasgos autoorganizados o SOFM (Self-Organizing Feature Maps) trata de reproducir.

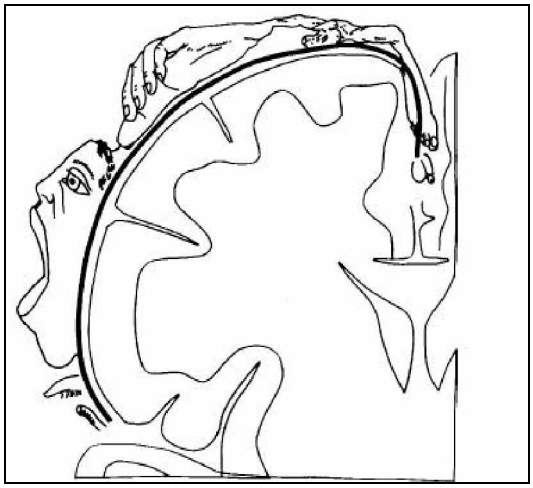


Figura 3.1 Córtex somatosensorial y motor

Los SOFM poseen un gran potencial de aplicabilidad práctica. De entre las clases de problemas del mundo real en los que han demostrado su eficacia cabe citar: clasificación de patrones, cuantificación vectorial, reducción de dimensiones, extracción de rasgos, monitorización de procesos, análisis exploratorio, visualización y minería de datos. Por ejemplo, los SOFM han sido empleados en reconocimiento de habla, control robot, optimización combinatorial, clasificación de proteínas y de restos arqueológicos, monitorización de procesos industriales, ayuda al diseño de circuitos integrados, reconocimiento de patrones financieros y minería de grandes bases de datos en Internet.

En este modelo, las neuronas se organizan en una arquitectura unidireccional de dos capas (Figura 3.2). La primera es la capa de entrada o sensorial, que consiste en *m* neuronas, una por cada variable de entrada, que se comportan como buffers, distribuyendo la información procedente del espacio de entrada a las neuronas de la segunda capa. Las entradas son muestras estadísticas $x(t) \in \Re^m$ del espacio sensorial.

El procesamiento se realiza en la segunda capa, que forma el mapa de rasgos, consistente habitualmente en una estructura rectangular de $nx \times ny$ neuronas (Figura 3.2), que operan en paralelo. Aunque la arquitectura rectangular es la más corriente, a veces también se utilizan capas de una sola dimensión (cadena lineal de neuronas) o de tres dimensiones (paralelepípedo). Etiquetaremos las *m* neuronas de entrada con el índice $k(1 \le k \le m)$, y las $nx \times ny$ neuronas del mapa con un par de índices $i \equiv (i, j)$ $(1 \le i \le nx, \quad 1 \le j \le ny)$ que determinan su localización espacial. Cada neurona de entrada *k* está conectada a todas las neuronas (*i, j*) del mapa mediante un peso sináptico w_{ijk} .

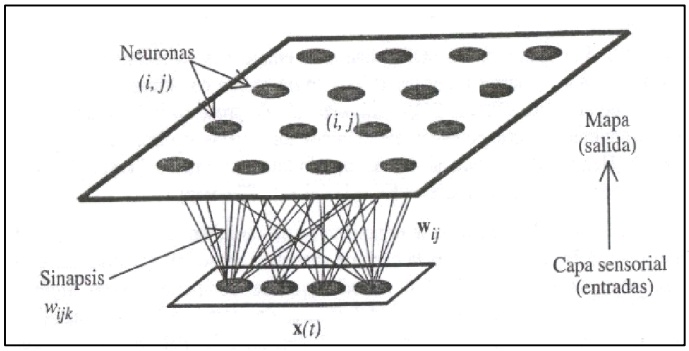


Figura 3.2 Arquitectura del SOFM

En resumen, el mapa puede describirse como un matriz de procesadores elementales *(i, j)* ordenados en dos dimensiones (Figura 3.3), que almacenan un vector de pesos

sinápticos o vector de referencia (codebook) $w_{ij}(t)$, con $\{w_{ij}(t); w_{ij} \in \Re^m, 1 \le i \le nx, \quad 1 \le j \le ny\}$.

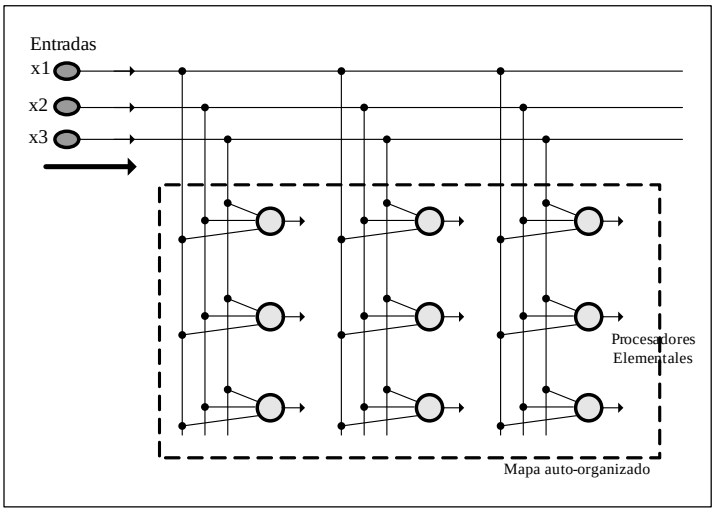


Figura 3.3 Disposición de los procesadores elementales en un SOFM bidimensional

En fase de ejecución (operación normal de la red), los pesos permanecen fijos. En primer lugar, cada neurona (*i,j*) calcula la similitud entre el vector de entradas *x*, $\{x_k \mid 1 \le k \le m\}$ y su propio vector de pesos sinápticos w_{ij}, según una cierta medida de distancia o criterio de similitud establecido. A continuación, se declara vencedora la neurona *g=(g1, g2)*, cuyo vector de pesos W_g es más similar al de entradas. De esta manera, cada neurona actúa como un detector de rasgos específicos, y la neurona ganadora nos indica el tipo de rasgo o patrón detectado en el vector de entradas.

$$d(w_g, x) = \min_{ij} \{d(w_{ij}, x)\}$$

Fórmula 3.1

En la fase de aprendizaje cada neurona del mapa sintoniza con diferentes rasgos del espacio de entrada. El proceso es el siguiente. Tras la presentación y procesamiento de

un vector de entradas $x(t)$, la neurona vencedora modifica sus pesos de manera que se parezcan un poco más a $x(t)$. De este modo, ante el mismo patrón de entrada, dicha neurona responderá en el futuro todavía con más intensidad. El proceso se repite para numerosos patrones de entrada, de forma que al final los diferentes vectores de referencia sintonizan con dominios específicos de las variables de entrada, y tienden a representar la función densidad de probabilidad $p(x)$ (o función de distribución) del espacio sensorial. Si dicho espacio está dividido en grupos, cada neurona se especializará en uno de ellos, y la operación esencial de la red se podrá interpretar entonces como un análisis cluster.

El modelo de mapa de Kohonen aporta una importante novedad, pues incorpora a este esquema, relaciones entre las neuronas próximas en el mapa. Para ello introduce una función de vecindad, que define un entorno alrededor de la neurona ganadora actual (vecindad); su efecto es que durante el aprendizaje se actualizan tanto los pesos de la vencedora como los de las neuronas pertenecientes a su entorno. De esta manera, en el modelo de SOFM se logra que las neuronas próximas sintonicen con patrones similares, quedando de esta manera reflejada sobre el mapa una cierta imagen del orden topológico presente en el espacio de entrada (ordenación de los detectores de rasgos).

En esencia, por medio del proceso descrito los SOFM realizan la proyección no lineal de un espacio multidimencional de entrada $\Re^m$ sobre un espacio discreto de salida, representado por la capa de neuronas. El mapa representa una imagen del espacio sensorial, pero de menor número de dimensiones, reflejando con mayor fidelidad aquellas dimensiones del espacio de entrada de mayor varianza (que suelen coincidir con los rasgos más importantes de las entradas). La proyección se realiza de manera óptima, en el sentido de que la topología del espacio de entrada se preserva en lo posible sobre la superficie de la red (entradas *x* similares se corresponden con neuronas próximas). Así, la distribución de las neuronas sobre el mapa resulta ser un reflejo de la función

densidad de probabilidad $p(x)$: regiones en el espacio sensorial cuyos representantes *x* aparecen con más frecuencia ($p(x)$ mayor) serán proyectadas sobre un número mayor de neuronas en el mapa.

La función vecindad representa matemáticamente de una forma sencilla el efecto global de las interacciones laterales existente entre las neuronas en el cerebro, pues en ves de considerar en detalle que una neurona trata de activar a sus vecinas y de inhibir a las alejadas (como sucede en el córtex), modelamos esta situación mediante una sencilla función que define el tamaño de la vecindad en torno a la vencedora, dentro de la cual todas las neuronas son premiadas actualizando sus pesos, y fuera de ella son castigadas al no actualizar sus pesos o al hacerlo en sentido contrario. La utilización de la función vecindad en el modelo de mapas autoorganizados aporta respecto del modelo competitivo sencillo dos ventajas adicionales: el ritmo efectivo de convergencia se mejora y el sistema es más robusto frente a variaciones en los valores iniciales de los pesos.

Hay que señalar que el tratamiento teórico del modelo de Kohonen es complejo, entre otras razones, porque se demuestra que la dinámica de aprendizaje no puede ser descrita mediante una función de Lyapunov (lo que simplificaría su estudio). Sin embargo, si se elimina la función vecindad, cada neurona individual representa un mapping lineal, haciendo factible un análisis teórico que puede proporcionar ideas sobre la operación del modelo completo.

3.2.2 Algoritmo de aprendizaje

Como hemos visto, la principal novedad de los SOFM consiste en que la modificación de los pesos no se aplica solamente a una neurona específica (la ganadora), sino también a su vecindad. Al comienzo del entrenamiento la vecindad comprende una amplia región

del mapa, lo que permite una ordenación global de los pesos sinápticos. Con el transcurso de las iteraciones, el tamaño de la vecindad se reduce, y finalmente solamente se modifican los pesos de la neurona ganadora. Así, el proceso de aprendizaje comprende dos fases fundamentales: una ordenación global, en la que se produce el despliegue del mapa; y un ajuste fino, en el que las neuronas se especializan.

Un algoritmo de aprendizaje autoorganizado

Se debe tener en cuenta que no existe un algoritmo de aprendizaje totalmente estándar para los SOFM. Sin embargo, el resultado final es bastante independiente de los detalles de su realización concreta, como puedan ser los pesos sinápticos de partida, el esquema de la actualización del ritmo de aprendizaje, o la forma establecida para la vecindad. A continuación exponemos un algoritmo autoorganizado habitual:

1) **Inicialización de los pesos** sinápticos w_{ijk} . Se puede partir en $t = 0$ de diferentes configuraciones: pesos nulos, aleatorios de pequeño valor absoluto (lo más habitual), o con un valor de partida predeterminado.

2) En cada iteración, **presentación de un patrón** $x(t)$ tomado de acuerdo con la función de distribución $p(x)$ del espacio sensorial de entrada (en la muy habitual situación de disponer solamente de un conjunto finito de patrones de entrenamiento, basta con tomar al azar uno de ellos y presentarlo a la red).

3) Cada neurona $i \equiv (i, j)$ en paralelo del mapa calcula la similitud entre su vector de pesos sinápticos w_{ij} y el actual vector de entradas *x*. Un criterio de medida de similitud muy utilizado es la **distancia euclídea:**

$$d^2(w_{ij}, x) = \sum_{k=1}^{n} \left(w_{ijk} - x_k \right)^2$$

Fórmula 3.2

4) Determinación de la neurona ganadora $g \equiv (g1, g2)$, cuya distancia sea la menor de todas.

5) Actualización de los pesos sinápticos de la neurona ganadora $g \equiv (g1, g2)$, y los de sus neuronas vecinas. La regla más empleada es

$$w_{ijk}(t+1) = w_{ijk}(t) + \alpha(t).h(|i-g|,t).(x_k(t) - w_{ijk}(t))$$

Fórmula 3.3

donde $\alpha(t)$ es un parámetro denominado ritmo de aprendizaje. La función *h(.)* se denomina función de vecindad, puesto que establece qué neuronas son las vecinas a la actualmente ganadora. Esta función depende de la distancia entre neurona *i* y la ganadora *g*, valiendo cero cuando *i* no pertenece a la vecindad de *g* (con lo que sus pesos no son actualizados), y un número positivo cuando sí pertenece (sus pesos sí son modificados). Como veremos, la vecindad es un conjunto de neuronas centrado en la ganadora. Tanto α como el radio de la vecindad disminuyen monótonamente con *t*.

6) Si se ha alcanzado el número máximo de iteraciones establecido, entonces el proceso de aprendizaje finaliza. En caso contrario, se vuelve al paso 2.

Se puede realizar a continuación una segunda fase en el aprendizaje, en la que se produce el ajuste fino del mapa, de modo que la distribución de los pesos sinápticos se ajuste más a la de las entradas. El proceso es similar al anterior, tomando $\alpha(t)$ constante e igual a un pequeño valor (por ejemplo, 0.01), y radio de vecindad constante e igual a uno.

En el aprendizaje, el número de iteraciones debe ser lo suficientemente grande por requerimientos estadísticos, así como proporcional al número de neuronas del mapa (a más neuronas, son necesarias más iteraciones), e independiente del número de componentes de *x*. Aunque 500 iteraciones por neurona es una cifra adecuada, de 50 a 100 suelen ser suficientes para la mayor parte de los problemas. Entre 20.000 y 100.000 iteraciones representan cifras habituales en la simulación por ordenador del entrenamiento de un SOFM.

Una cuestión a tener presente es que el criterio de similitud y la regla de aprendizaje que se utilicen deben ser métricamente compatibles; así ocurre con la distancia euclídea (Fórmula 3.2) y la regla de aprendizaje (Fórmula 3.3). El empleo de diferentes métricas para la fase de recuerdo y para la actualización de los pesos puede causar problemas en el desarrollo del mapa. Una medida de similitud alternativa, más simple que la euclídea, es la correlación o producto escalar:

$$C_{ij} = \sum_{k=1}^{n} w_{ijk} x_k$$

Fórmula 3.4

que suele incorporarse al algoritmo, junto con la regla de adaptación (Fórmula 3.3). Sin embargo, dicha regla procede de la métrica euclídea, y la correlación solamente es compatible con esta métrica si se utilizan vectores normalizados de norma1 (en cuyo caso distancia euclídea y correlación coinciden). Por esta razón, en ocasiones se realiza la afirmación errónea de que el modelo de Kohonen precisa vectores normalizados. Si utilizamos la distancia euclídea (Fórmula 3.2) y la regla (Fórmula 3.3), no es necesario tratar con vectores normalizados (otra cuestión diferente es que en determinado problema dicha normalización pueda ser aconsejable para mantener las entradas dentro de determinado rango dinámico).

Interpretación del algoritmo de aprendizaje

La siguiente interpretación del proceso de aprendizaje puede resultar interesante para comprender la operación de los SOFM. El efecto de la regla de aprendizaje (Fórmula 3.3) no es otro que en cada iteración acerca en una pequeña cantidad el vector de pesos *w* de la neurona de mayor actividad (ganadora) al vector de entrada *x***,** como puede apreciarse en la Figura 3.4, donde la expresión

$$\Delta w(t) = \alpha.(x - w)$$

Fórmula 3.5

representa el incremento del vector de pesos de la ganadora $(0 < \alpha < 1)$. Así, en cada iteración el vector de pesos de la neurona vencedora rota hacia el presentado, y se aproxima a él en una cantidad que depende del tamaño del ritmo de aprendizaje α .

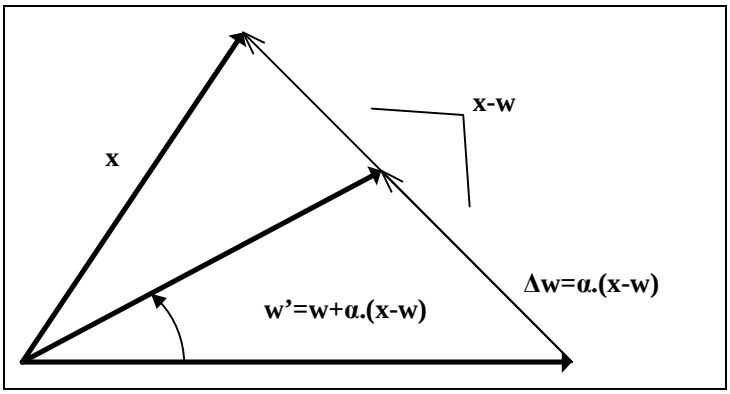


Figura 3.4 Interpretación geométrica de la regla de aprendizaje euclídea

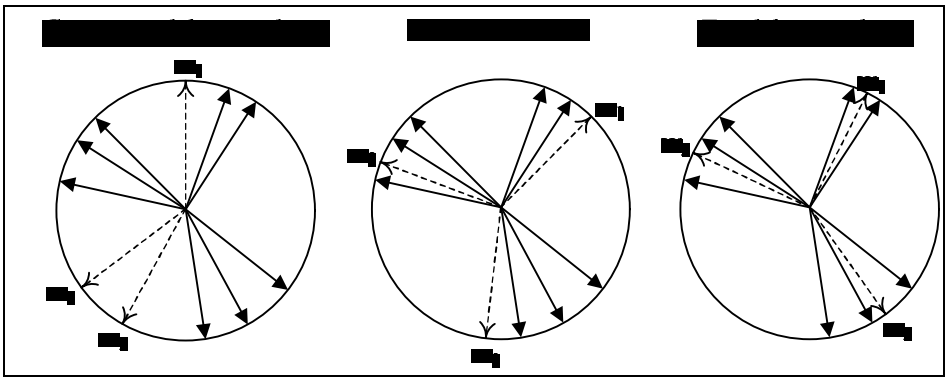


Figura 3.5 Proceso de aprendizaje en dos dimensiones

En la Figura 3.5 se muestra cómo opera la misma regla de aprendizaje para el caso de varios patrones pertenecientes al espacio de entrada, representados en la figura por vectores de trazo continuo (por simplicidad todos los vectores poseerán la misma longitud, el radio y la circunferencia). Supongamos que los vectores del espacio de entrada se agrupan en tres clusters y que el número de neuronas de la red es también tres. Al principio del entrenamiento $(t=0)$ los vectores de pesos de las tres neuronas (representados por vectores de trazo discontinuo) son aleatorios y se distribuye por la circunferencia. Conforme avanza el aprendizaje, y en virtud de la regla (Fórmula 3.3), éstos se van acercando progresivamente a las muestras procedentes del espacio de entrada, para quedar finalmente estabilizados como centroides de los tres clusters.

Consideraciones prácticas: ritmo de aprendizaje y función vecindad

El ritmo de aprendizaje $\alpha(t)$ es una función monótonamente decreciente con el tiempo, siendo habitual su actualización mediante una función lineal

$$\alpha(t)=\alpha_0+(\alpha_f-\alpha_0)\frac{t}{t_\alpha}$$

Fórmula 3.6

Con α_0 el ritmo de aprendizaje inicial (<1.0), α_f el final $(\cong 0.01)$ y t_α el máximo número de iteraciones hasta llegar a α_f. Una alternativa es utilizar una función que decrece exponencialmente.

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_f}{\alpha_0} \right)^{\frac{t}{t_\alpha}}$$

Fórmula 3.7

El empleo de una u otra función no suele influir demasiado en el resultado final.

La función vecindad $h(|i-g|,t)$ define en cada iteración *t* si una neurona *i* pertenece o no a la vecindad de la vencedora *g*. la vecindad es simétrica y centrada en *g*, de ahí que representamos como uno de sus argumentos la distancia entre la neurona genérica *i=(i,j)* y la vencedora *g=(g1,g2)*, pues

$$|i-g| = \sqrt{(i-g1)^2 + (j-g2)^2}$$

Fórmula 3.8

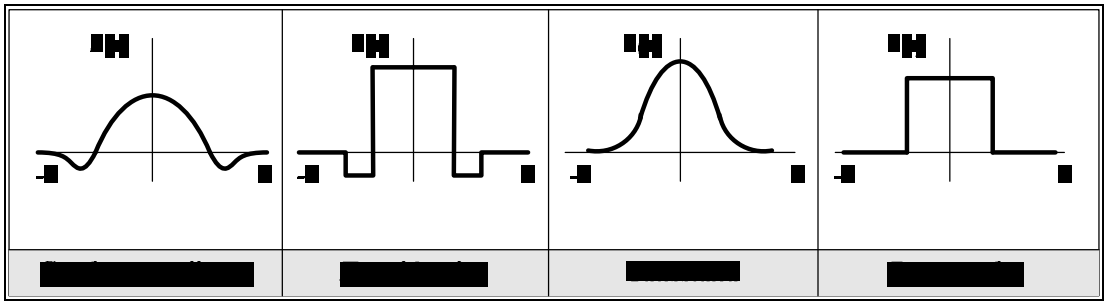


Tabla 3.1 Diferentes formas de las funciones vecindad

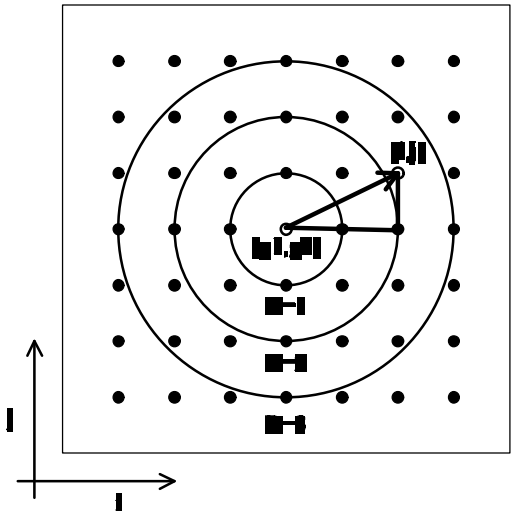


Figura 3.6 Vecindades delimitadas por la función escalón en un mapa de Kohonen para diferentes ratios

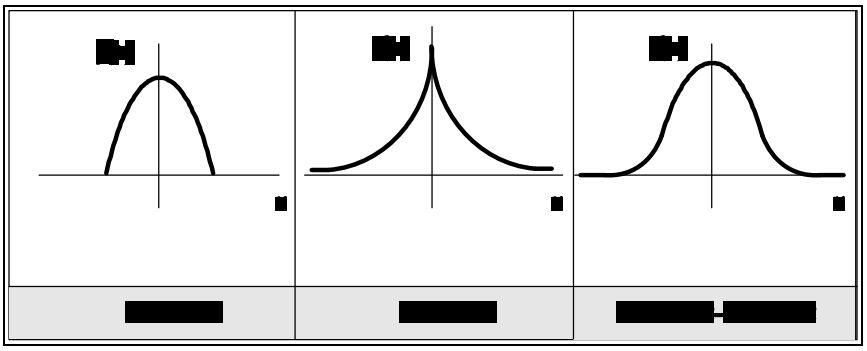


Figura 3.7. Funciones convexa (a), cóncava (b) y convexa-cóncava (c)

En general, *h(.)* decrece con la distancia a la vencedora, y depende de un parámetro denominado radio de vecindad *R(t)*, que representa el tamaño de la vecindad actual. En realidad, bajo la forma funcional de *h(.)* se encapsula el complejo sistema de iteraciones laterales existentes entre las neuronas del mapa, que sí aparecerían explícitamente en una formulación del modelo más orientada a la biología, estableciendo la forma y la intensidad de la interacción entre neuronas.

También se utiliza a veces funciones gaussianas o en forma de sombrero mejicano (tabla 3.1), continúas y derivables en todos sus puntos, que al delimitar vecindades decrecientes en el dominio espacial, establecen niveles de pertenencia en lugar de fronteras nítidas. La función vecindad posee una forma definida.

3.2.3 Algunas variantes de los SOFM

Hasta el momento nos hemos limitado a exponer el modelo de mapas autoorganizados clásico, pero es posible incorporar modificaciones que mejoren algunos aspectos de su operación. Una de las más interesantes consiste en comenzar el proceso de aprendizaje con un mapa de tamaño pequeño, para ir introduciendo progresivamente nuevas neuronas a medida que se desarrollo el entrenamiento, cuyos pesos se obtiene de la interpolación de los valores de los pesos de las neuronas vecinas, ya presentes en el mapa. Esquemas de mapas autoorganizados crecientes. Ésta es una forma de resolver el problema de cómo seleccionar de entrada las dimensiones óptimas del mapa autoorganizado.

Un aspecto no tratado hasta el momento es el efecto de los bordes del mapa. Debido a que las neuronas situadas en las orillas del mapa poseen menos vecinas, presentan una cierta asimetría respecto de las demás, puesto que se actualizan menos frecuentemente; como resultado los mapas desarrollados aparecerán algo encogidos en sus orillas. Una forma de resolver este problema consiste en establecer un ritmo de aprendizaje mayor para las neuronas de los bordes. Otra solución es cerrar la red, es decir, considerar que una neurona de un extremo del mapa es vecina de la neurona correspondiente del extremo opuesto, obteniendo así una superficie esferoidal o toroidal (dependiendo de cómo cerremos la red) donde están situadas las neuronas. De este modo se resuelve también el problema relacionado con patrones que durante el proceso de entrenamiento puedan quedar aislados al situarse en un extremo del mapa, con los posibles caminos de salida cortados por prototipos pertenecientes a otras clases.

Una de las variantes de los mapas de Kohonen más conocidas se orienta en el sentido de conseguir que todas las neuronas ganen con la misma frecuencia, de modo que no exista un conjunto de neuronas que monopolicen el proceso y no permitan un correcto desarrollo del mapa. Para lograr esta equiprobabilidad en el proceso WTA, se ha introducido una modificación al algoritmo denominada mecanismo de consciencia, cuya idea consiste en contabilizar durante el proceso de aprendizaje el número de veces que cada neurona gana, y penalizar aquellas que ganen demasiadas ocasiones.

El empleo del modelo SOFM en tareas de clustering es un error bastante extendido. Por ello Kohonen insiste a menudo en que el modelo SOFM es útil para visualizar datos, pero no debe emplearse para clasificar (clustering); para este tipo de tareas debe recurrirse al LVQ, de tipo supervisado.

3.3 EJEMPLOS DE APLICACIONES

Los mapas autoorganizados, como modelo no supervisado, pueden llevar a cabo, entre otros, los siguientes tipos de procesamiento:

- Reducción de dimensiones: ayuda a eliminar la información redundante y el "ruido" de los datos, y se lleva a cabo mediante una transformación lineal que disminuye el número de variables hasta un número deseado. Resultando importante porque hay atributos que resultan nocivos para el aprendizaje tales como: Irrelevancia (el atributo no ofrece capacidad de discriminación en el aprendizaje), Separación de la información (una información interesante puede estar recogida en varios atributos). Se trata de aumentar la densidad de la información reduciendo el espacio de entrada.

- Preprocesamiento de datos para otros sistemas: la preparación de datos para el análisis es uno de los aspectos de Data Mining que más tiempo requiere.

- Monitorización de procesos: está basada en procedimientos de adquisición y procesamiento de información. Las tareas de monitorización pueden clasificarse en una o mas, tales como: conducidos por datos (la eficiencia de los métodos basados en datos, los analíticos y los basados en conocimiento dependen de la calidad, tipo de modelos, en la calidad y cantidad de datos disponibles), analítico (Los métodos analíticos que utilizan residuos como rasgos, están referidos a los métodos analíticos redundantes).

- Cuantificación vectorial: es una técnica que permite comprimir la información, permitiendo también reducir la dispersión estadística presente en los datos. La principal motivación para realizar este proceso viene determinada por el modelo de producción de la voz.

- Modelado de funciones densidad: la función de densidad de una variable continua *X*, es la curva teórica que se deduce al imaginar la representación de las frecuencias de los resultados ocurridos tras la repetición del experimento aleatorio "infinitas" veces.

- Análisis cluster (aunque se recomienda emplear LVQ): la agregación de puntos en el espacio de entrada donde la "distancia" entre cada par de objetos es menor que la distancia de cualquiera de ellos a otro objeto que no pertenece al cluster. Siendo su objetivo descubrir la estructura en los datos de entrada, Busca agrupamientos entre los ejemplos de forma que cada grupo sea homogéneo y distinto de los demás.

Para entender su operación, a continuación comentaremos un ejemplo práctico concreto de su aplicación.

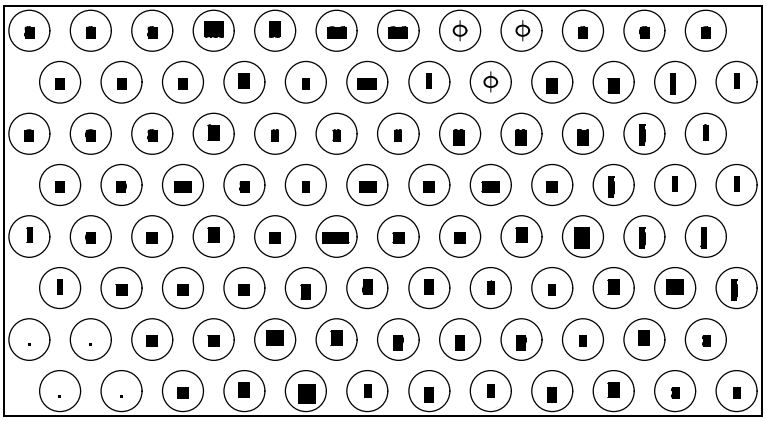


Figura 3.8 Mapa auto-organizado Fonético

El mapa fonético

Es un ejemplo muy conocido de aplicación de los SOFM, en el que un mapa auto-organizado se entrena con lo fonemas (en realidad su transformada de Fourier) de un cierto idioma (en el trabajo original se empleó el finlandés). Tras la fase de aprendizaje, los vectores de entrada se presentan a la red, quedando representados sobre la superficie del mapa (Figura 3.8), obteniéndose como resultado final un mapa fonético en el que cada neurona se ha especializado en reconocer un tipo de fonema, y en el que sonidos similares o pequeñas variaciones quedan representados sobre neuronas vecinas.

Posteriormente, se aplicó al mapa entrenado el algoritmo LVQ (Learning Vector Quantiazation), que puede utilizase también como un procedimiento supervisado de ajuste fino del mapa. El sistema así construido alcanza un rendimiento muy alto en el reconocimiento de fonemas, especialmente cuando se introduce en el modelo información contextual. Por otra parte, es de destacar el parecido del mapa fonético con los mapas fonotópicos del córtex temporal del cerebro.

3.4 MODELOS DE NEURONAS DE KOHONEN. MEDIDAS DE SIMILITUD

El modelo de neurona de Kohonen se basa en el cálculo de la similitud entre el vector de entradas y el de pesos. Así, dependiendo del criterio de distancia que se seleccione, se tendrá un modelo u otro. Expondremos a continuación algunos de los criterios más habituales.

Uno de los más comunes es la correlación o producto escalar:

$$C_{ij} = \sum_{k=1}^{n} w_{ijk} x_k$$

Fórmula 3.9

según el cual, dos vectores serán más similares cuanto mayor sea su correlación. Es interesante observar que una neurona SOFM que utilice este criterio de distancia coincide básicamente con el modelo de neurona estándar de los ANS. Sin embargo, esta medida es sensible al tamaño de los vectores; grandes diferencias en sus longitudes pueden introducir una importante distorsión en la medida de similitud.

Su interés radica en que esta medida se basa en una característica relativa a ambos vectores, como es su ángulo, independientemente de sus tamaños. Otro de los criterios de más amplio uso es la distancia euclídea

$$d^2(w_{ij}, x) = \sum_{k=1}^{n} \left(w_{ijk} - x_k \right)^2$$

Fórmula 3.10

De acuerdo con este criterio, dos vectores serán más similares cuanto menor sea su distancia. Si se utiliza una red de Kohonen para análisis cluster, la distancia euclídea es más adecuada cuando los grupos a extraer están compuestos por nubes esféricas de puntos en torno a un centro. Si no es así, el algoritmo tratará de ajustar los datos en múltiples grupos esféricos.

3.5 APLICACION PRACTICA

Para finalizar con este capítulo, presentamos la aplicación práctica que representa el funcionamiento de la red neuronal Kohonen.

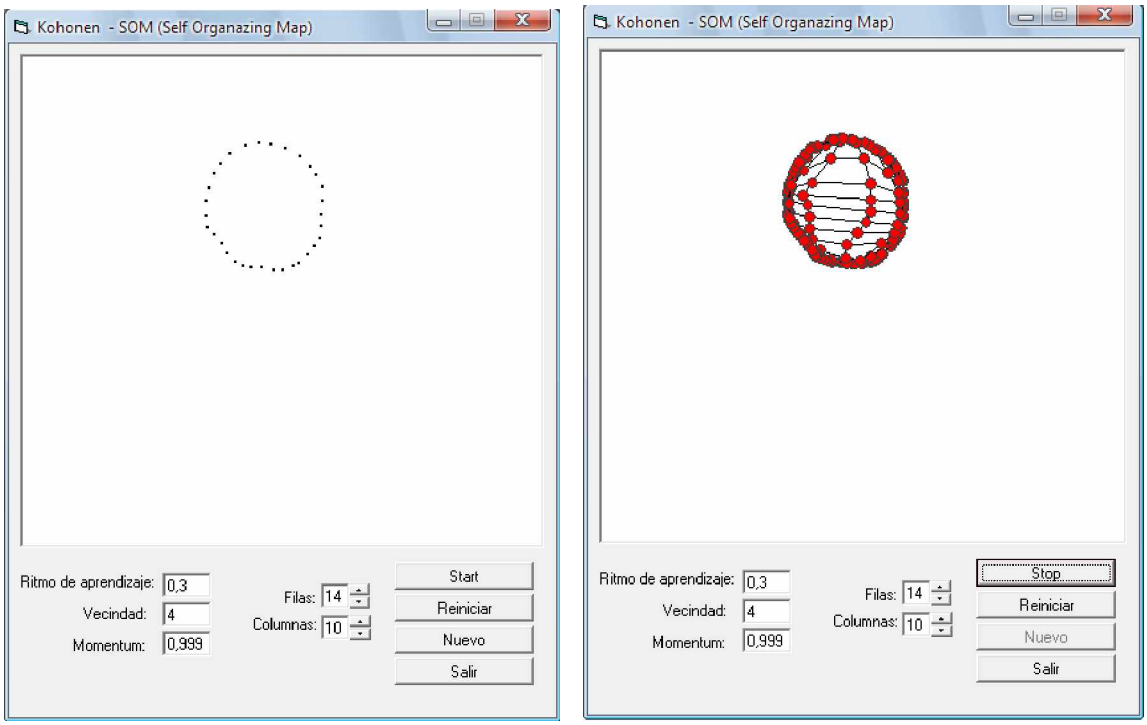


Figura 3.9 Aplicación práctica que demuestra el funcionamiento de Kohonen.

El objetivo de esta aplicación es obtener los rasgos principales del objeto dibujado en pantalla, para luego realizar cualquier análisis o búsqueda en base a dicho rasgo.

Este concepto es utilizado en un software llamado PicSOM en el cual en base a los rasgos aprendidos de un fotografía, puede buscar otras con rasgos similares.

3.6 CONCLUSION

En estas redes autoorganizadas podemos destacar algunos aspectos importantes tales como: en su entrenamiento no se presentan las salidas objetivo que se desean asociar a cada patrón de entrada, si no que la red, a partir de un proceso de autoorganización proporcionará cierto resultado, el cual si se quiere que sea de calidad requerirá de un amplio número de patrones. Podemos señalar también que la operación del este modelo es bastante robusta.

CAPITULO 4
REDES NEURONALES REALIMENTADAS

Ante un determinado patrón de entradas una red unidireccional proporciona inmediatamente una respuesta de salida, por lo que su operación es siempre estable en fase de recuerdo o ejecución. Pero si introducimos realimentaciones en este esquema, la información se propagará tanto hacia adelante como hacia atrás, comportándose por lo tanto como un sistema dinámico, de difícil análisis, en general, y en el que deberá garantizarse la estabilidad de su respuesta.

La respuesta de la red realimentada en ocasiones se estabilizará tras un determinado número de iteraciones, convergiendo a un estado estable, punto fijo. En otros casos la respuesta puede no converger, describiendo una trayectoria caótica en el espacio de las salidas. En definitiva, en el caso de las redes retroalimentadas surge el problema fundamental relacionado con la estabilidad de la respuesta de la red neuronal. La demostración de estabilidad de la respuesta suele hacer uso del método de Lyapunov

La función de Lyapunov se denomina también función energía de Lyapunov, pues constituye una generalización del concepto físico de energía. Es interesante recalcar que con este formalismo matemático simplemente se está expresando que si somos capaces de encontrar una cierta función de energía de la red que disminuya con su operación, entonces ésta será estable, de modo que disponemos de una manera relativamente simple de estudiar sus propiedades de estabilidad.

4.1 MODELO DE HOPFIELD

4.1.1 Modelo de neurona y arquitectura. Dinámicas

La arquitectura de la red de Hopfield consiste básicamente en una única capa de neuronas, donde cada una se conecta con todas las demás (Figuras 4.1 y 4.2). El modelo de neurona empleado se describe más detalladamente en la Figura 4.3. Se trata de una neurona similar a la del perceptrón, un sencillo elemento de umbral, con entradas $x_j(t)$ binarias ({0,1} o {-1, +1}, según convenga), y salidas que en principio llamaremos $y_i(t)$, también binarias. Los pesos w_{ij} son continuos, es decir, números reales. La neurona i calcula el potencial postsináptico h_i (también denominado potencial local o de membrana), como la suma de las entradas ponderada con los pesos sinápticos, menos un cierto umbral de disparo

$$h_i(t) = \sum_j w_{ij} x_j(t) - \theta_i$$

Fórmula 4.1

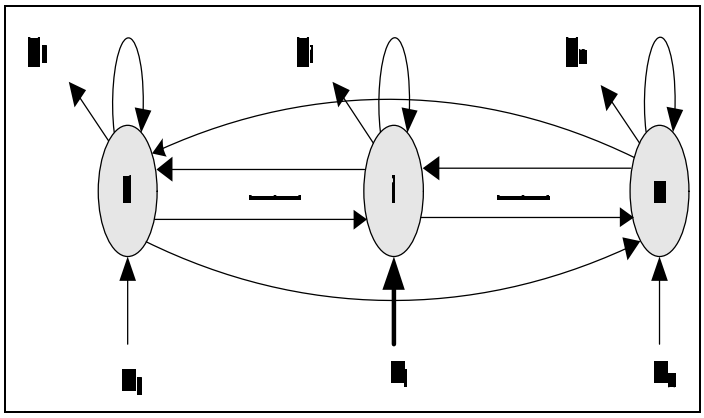


Figura 4.1 Esquema genérico de la red de Hopfield

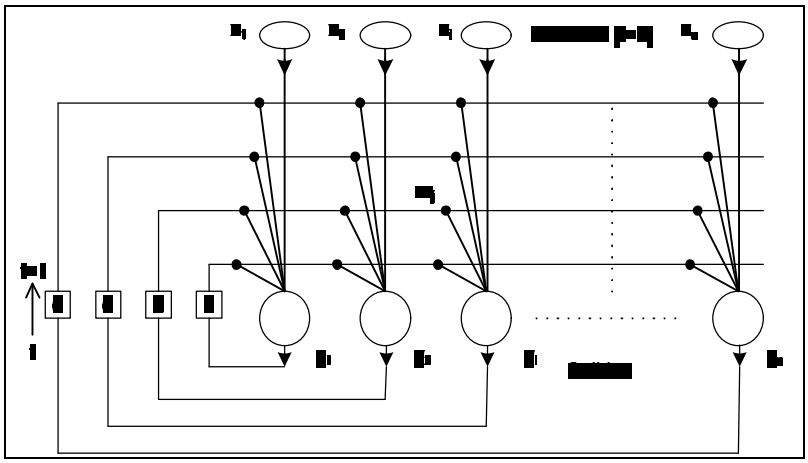


Figura 4.2 Arquitectura de la red de Hopfield. La letra 'd' representa un retraso (delay) de una iteración (una salida y en *t* se convierte en una entrada *x* en el instante posterior t+1)

Cuando las neuronas hacen uso de valores {-1, +1} se denominan de tipo Ising, siendo la función de activación que emplea la signo-de-equis, *f(x)=sign(x)*. En este caso, el estado +1 indica una neurona activada, y el -1 desactivada.

En un primer estadio *t=0*, las neuronas reciben las entradas provenientes del exterior $x_j(0)$, calculando las salidas asociadas $y_i(0)$. Debido a las realimentaciones (figura 4.2), estas salidas se convierten en las nuevas entradas de la red $x_j(1)$, proporcionando ésta una nueva salida $y_i(1)$. En general, ante las entradas $x_i(t)$, la red proporciona una salidas $y_i(t)$, que se convierten en las nuevas entradas $x_i(t+1)$, de modo que la operación de la red de Hopfield se puede expresar como:

$$x_i(t+1) = f\left(\sum_j w_{ij} x_j(t) - \theta_i\right)$$

Fórmula 4.2

regla que rige su dinámica. En resumen, al comienzo (*t=0*) cada neurona recibe del exterior un vector de entradas $x_j(0)$, de modo que cada neurona *i* queda situada inicialmente en ese estado, cuyo valor será 0 o 1. A partir de entonces, la red neuronal interrumpe su comunicación con el exterior y procesa las entradas recibidas; para tiempos *t+1* sucesivos, las entradas de cada neurona son el estado en el que se sitúan en el instante anterior *t*. habitualmente diremos que el estado *x* de la red de Hopfield en *t* viene determinado por el estado de actuación de todas y cada una de sus neuronas

$$x(t)=(x_1(t) \ldots x_i(t) \ldots x_n(t))^T$$

Fórmula 4.3

Dinámicas de la red

Dada la regla que rige la actualización del estado de una neurona individual (Fórmula 4.2), podemos plantearnos a continuación los diferentes esquemas de actualización o dinámicas de las neuronas de la red. Podemos distinguir dos tipos genéricos:

a) **Dinámica asíncrona o modo serie** de operación: en un instante t solamente una neurona de la red actualiza su estado. Ésta puede ser seleccionada aleatoriamente, o bien siguiendo un cierto orden preestablecido.

b) **Dinámica síncrona o modo paralelo** de operación, en un instante t varias neuronas actualizan su estado a la vez. En el caso en que todas las neuronas de la red lo hagan al unísono se tiene el modo completamente paralelo. No obstante, cuando hablemos de dinámica síncrona o paralela nos referiremos a la actualización de todas a la vez, por ser el caso más común.

4.1.2 Memoria asociativa

La información en nuestro cerebro, al parecer, no se esta alojada en casillas de memoria definidas, sino más bien aparece dispersa. Para recuperar información, no buscamos, por ejemplo, en la casilla de memoria número 43.537 de nuestro cerebro, sino que la recuperamos a partir de alguna pista. Por ejemplo, para recordar el color de pelo de Juan Pérez, pensamos en la etiqueta Juan Pérez, y casi instantáneamente aparece su imagen en nuestra consciencia. La operación de nuestra memoria se aproxima más a la de las llamadas memorias asociativas, también denominadas memorias direccionables por contenido, debido a que asocian ideas con ideas, ideas con imágenes, imágenes con etiquetas, etc.

Las redes neuronales operan siguiendo este mismo esquema. Por ejemplo, un MLP implementa una memoria asociativa de tipo heteroasociativo, al asociar un patrón de entrada con cierto patrón de salida diferente. Por el contrario, el modelo de Hopfield toma el papel de una memoria de tipo auto asociativo, está ideada para asociar un patrón de entradas consigo mismo.

La red de Hopfield como memoria asociativa. Estados estables

El objetivo del entrenamiento en este modelo es almacenar un determinado conjunto de patrones x^{μ} en la forma de estados estables de la red. De este modo, ante cierto patrón de entradas x se espera que la salida sea uno de los estados estables memorizados x^{μ}, por lo que la red actúa como una memoria asociativa, asociando cierto patrón de entrada a otro de salida. Por otra parte, si presentamos a la red como entradas cierto estado estable x^{μ} de los memorizados, la repuesta será obviamente el mismo estado x^{μ} por definición de estado estable, de modo que la red de Hopfield opera como una memoria asociativa. El interés de una memoria autoasociativa es el siguiente: si en vez de presentar inicialmente uno de los patrones memorizados x^{μ} presentamos uno de ellos,

pero que incorpore cierto nivel de ruido, x^{μ} +ruido, se espera que la red neuronal evolucione hasta alcanzar el estado estable más próximo, que deberá ser precisamente el estado memorizado (sin ruido) x^{μ}, recuperando el patrón original. Así, mediante esta red autoasociativa podemos eliminar el ruido en patrones. La red se dice entonces que es tolerante al ruido o que filtra el ruido presente en las señales de entrada.

4.1.3 Función energía de la red

Supongamos el modelo original de Hopfield discreto, con neuronas tipo umbral de salida {0,1} y dinámica asíncrona, por la cual en cada iteración la neurona a la que le corresponde actualizar su estado según (Fórmula 4.2) es elegida aleatoriamente. Supongamos que no existe realimentación de una neurona consigo misma, es decir $w_{ii} = 0$, o lo que es lo mismo, la matriz sináptica es de diagonal nula.

$$h_i(t).\Delta x_i(t) \geq 0 \text{, o bien, } \left(\sum_j w_{ij} x_j(t) - \theta_i \right).\Delta x_i(t) \geq 0$$

Fórmula 4.4

Definimos el incremento de energía de la red cuando la neurona *i* se actualiza como

$$\Delta E_i = -\left(\sum_{j=1}^{n} w_{ij} x_j(t) - \theta_i \right) \Delta x_i(t)$$

Fórmula 4.5

Si la matriz de pesos es de diagonal nula $w_{ij} = 0$, y además es simétrica $w_{ij} = w_{ji}$, la expresión sería:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

Fórmula 4.6

que es la energía de la red. Por construcción, la variación de esta energía es siempre menor o igual a cero $(\Delta E \leq 0)$, o lo que es lo mismo, en la operación de la red (Fórmula 4.2) su energía nunca crece. Como la energía está limitada, la red siempre llegará a un estado de mínima energía, que será un mínimo local de la energía de la red. Este mínimo local se corresponderá con un estado estable o punto fijo.

Realizando un estudio riguroso se llega a la conclusión de que para que el funcional (Fórmula 4.6) sea una verdadera función energía, bastan las siguientes dos condiciones:

$$w_{ij} = w_{ji} \qquad \text{y} \qquad w_{ii} \geq 0$$

Fórmula 4.7

Es decir, basta que la matriz de pesos sea simétrica y de diagonal no negativa.

4.2 APRENDIZAJE EN LA RED DE HOPFIELD

Un estado estable de la red es un mínimo local de la función energía, por lo tanto, los recuerdos o memorias que la red almacena son mínimos locales de la red, por lo que el aprendizaje consistiría en conseguir que la red almacene como estados estables un conjunto de recuerdos o memorias dado. Para ello, la regla de aprendizaje que se defina deberá encontrar el conjunto de pesos sinápticos *W* que hace que la función energía tenga esos patrones como mínimos locales.

4.2.1 Regla de Hebb

En la tabla 4.1 aparece un resumen de las características de este modelo que podríamos calificar como modelo de Hopfield discreto clásico.

Neurona	Tipo umbral, con salidas {0,1}
Dinámica	Asíncrona, selección aleatoria de la neurona a la que le corresponde actualizar su estado (ec. 4.2)
Arquitectura	Red monocapa, completamente interconectada y realimentada. Con $w_{ii} = 0$, y $w_{ij} = w_{ji}$
Aprendizaje	Regla de Hebb
Otras características	Función energía de la red Propiedades emergentes: memoria asociativa y tolerancia a fallos

Tabla 4.1 Resumen de las características del modelo de Hopfield discreto original

Supongamos la neurona tipo Ising {-1, +1} y umbrales nulos, y sea un conjunto de *p* patrones x^{μ}, $\mu = 1,\ldots, p$, que deseamos memorizar. La regla de Hebb en este caso se expresa

$$w_{ij} = \frac{1}{n}\sum_{\mu=1}^{p} x_i^{\mu} x_j^{\mu}$$

Fórmula 4.8

que puede observarse que cumple las dos condiciones que aseguran la estabilidad de la red

$$w_{ij} = w_{ji} \qquad w_{ii}$$

Fórmula 4.9

Obsérvese que en este caso la regla de Hebb no actúa incrementalmente, sino que de forma directa proporciona la matriz de pesos sinápticos de la red. Se han realizado multitud de estudios sobre el modelo de Hopfield con esta regla de aprendizaje.

A continuación estudiaremos el comportamiento de la red como memoria asociativa. Por medio de (Fórmula 4.8) tenemos que la red memoriza el patrón *x*. Supongamos que presentamos en *t=0* a la red entrenada un patrón *x'(0)*, cuyas primeras m componentes son diferentes a las del patrón memorizado *x*, y cuyas *n-m* componentes finales son las mismas.

De modo que el patrón presentado difiere del memorizado en m componentes; estos componentes diferentes podemos interpretarlos como ruido presente en el patrón de entrada. Calculemos la respuesta de la red ante el patrón con ruido en una primera iteración:

$$h_i(t=0)=\sum_j w_{ij}x'_j(0)=\frac{1}{n}x_i\sum_j x_j x'_j(0)=\frac{1}{n}x_i\left[-m+(n-m)\right]=\left(1-\frac{2m}{n}\right)x_i$$

Fórmula 4.10

Por lo tanto, si se cumple la condición

$$\left(1-\frac{2m}{n}\right)>0\Rightarrow m<\frac{n}{2}\Rightarrow x_i(1)=sign(h_i(0))=x_i$$

Fórmula 4.11

Recuperaremos el patrón puro memorizado. Por lo tanto, si partimos de un estado inicial que contenga menos de la mitad de las componentes diferentes a las del patrón memorizado, la evolución de la red nos lleva a recuperarlo en una sola iteración. En otras palabras, la red ha eliminado el ruido presente en el patrón de entrada y ha

recompuesto el original. El conjunto de patrones variaciones en torno al memorizado desde los cuales se puede alcanzar un cierto mínimo local se denomina cuenca de atracción del estado estable.

En cuanto a la energía de los estados memorizados y falsos, se puede demostrar que si α se mantiene pequeño, los estados memorizados, que sabemos son esencialmente estados estables, son los mínimos globales de la función energía, mientras que los estados estables falsos que aparecen son tan sólo mínimos locales, menos profundos que los memorizados.

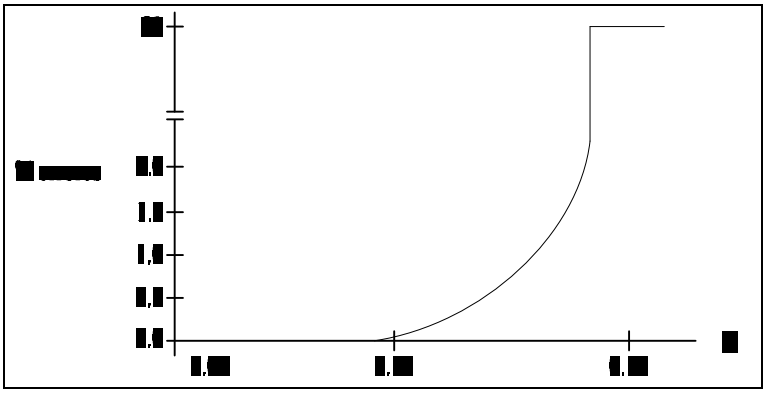


Figura 4.3 Porcentaje de errores en la red de Hopfield entrenada con la regla de Hebb en función del nivel de almacenamiento α . A partir del valor crítico α =0.14 se produce una completa pérdida de memoria.

En resumen, en el modelo clásico de Hopfield el número de patrones que la regla de Hebb permite almacenar es inferior al número de neuronas de la red (el 14% de n, aproximadamente), lo cual representa un grave problema, pues supone un gran derroche de recursos. No obstante, empleando reglas de aprendizaje más eficientes que la de Hebb se demuestra que podrían almacenarse hasta *2.n* patrones en una red de n neuronas. Es decir, el α límite teórico para el modelo de Hopfield es *2*.

La regla de aprendizaje de Hebb presenta los siguientes problemas:

a) No asegura que los patrones de entrenamiento sean mínimos de la energía, es decir, no se garantiza el almacenamiento exacto de dichos patrones.

b) Las memorias almacenadas pueden tener asociadas cuencas de atracción pequeñas.

c) La relación patrones/neuronas es pequeña (α =0.14), es decir, presenta una muy limitada capacidad de almacenamiento.

d) Presupone que los patrones son no correlacionados y aleatorios.

4.2.2 Reglas de aprendizaje óptimas

Regla de la pseudoinversa

Este algoritmo hace uso del cálculo de la pseudoinversa de una matriz, es válida para patrones aleatorios y no aleatorios, ortogonales y no ortogonales, asegura el rendimiento exacto de los patrones, y permite un nivel de almacenamiento de α =1, es decir hasta un número de patrones igual al número de neuronas de la red. Así, la de la pseudoinversa es una regla más eficiente que la de Hebb, aunque también es computacionalmente más compleja, no iterativa e involucra cálculos no locales.

A continuación pasamos a introducir esta regla de aprendizaje. Para ello, sea W la matriz de pesos sinápticos de la red de Hopfield, si x^{μ} es uno de los p vectores a memorizar, se debe cumplir:

$$signo\left[W.x^{\mu}\right] = x^{\mu}$$

Fórmula 4.12

Si trabajamos con neuronas {-1, +1}. Podemos sustituir la condición genérica anterior por otra particular, más restrictiva

$$W.x^{\mu} = x^{\mu}$$

Fórmula 4.13

Si colocamos los p patrones de aprendizaje x^{μ} como columnas de cierta matriz $\sum$, ésta tendrá por dimensiones $n \times p$, y la condición anterior se puede reescribir como

$$W\sum = \sum$$

Fórmula 4.14

El objetivo del aprendizaje será encontrar la matriz *W* que cumple esta condición. Si $\sum$ fuese una matriz cuadrada de determinante no nulo, sería invertible, con lo que la solución de (Fórmula 4.14) sería

$$W = \sum.\sum^{-1}$$

Fórmula 4.15

Pero para el caso general de una matriz no cuadrada, se demuestra que la solución de (Fórmula 4.14) es la siguiente

$$W = \sum.\sum^{+}$$

Fórmula 4.16

Siendo $\sum^{+}$ la matriz pseudoinversa de $\sum$, que representa la generalización de la inversión de una matriz cuadrada para el caso de matrices rectangulares.

Con el algoritmo de aprendizaje basado en la pseudoinversa, la matriz de pesos se obtiene del siguiente modo: en primer lugar se construye la matriz $\sum$ con los patrones como columnas, la matriz de pesos *W* de la red de Hopfield se obtiene directamente a partir de ella aplicando (Fórmula 4.16).

4.3 EJEMPLO: RECONOCIMIENTO DE CARACTERES

Un ejemplo ilustrativo de la operación de la red de Hopfield se basa en el reconocimiento de caracteres. Supongamos que disponemos de un conjunto de imágenes formado por diez prototipos correspondientes a los dígitos del 0 al 9, con cada imagen compuesta por 11 x 7 píxeles. A cada píxel, que puede estar encendido (1) o apagado (0), se le asigna el estado de una neurona (11x7=77 neuronas en total).

En una primera experiencia entrenamos una red de Hopfield discreta clásica de 77 neuronas haciendo uso de la regla de Hebb. Si tratamos de memorizar un único prototipo (por ejemplo, el correspondiente al dígito 0) podremos fácilmente observar que éste se convierte en un estado estable de la red, y que variaciones del anterior pueden aproximarse al digito 0 memorizado, presentando por lo tanto cierta tolerancia al ruido.

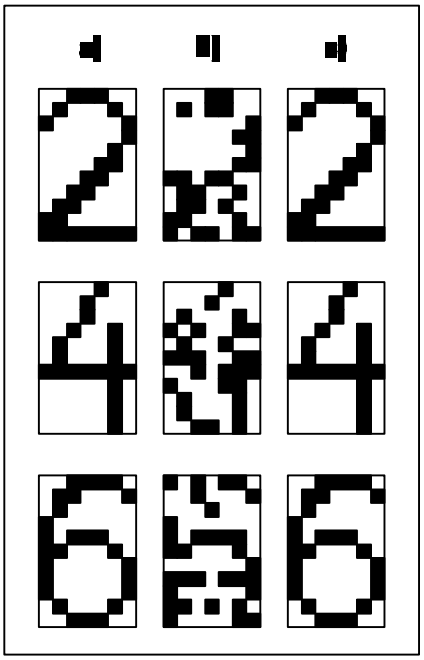


Figura 4.4 Columna a), dígitos originales; columna b), dígitos empleados como entradas a la red de Hopfield, con el 25% de los píxeles invertidos (ruido); columna c), estado final proporcionado por la red tras cierto número de iteraciones.

En una segunda experiencia entrenamos la red usando como ejemplos las imágenes correspondientes a los diez dígitos anteriores haciendo uso de la regla de la pseudoinversa, que sabemos que asegura el almacenamiento perfecto de los patrones si estos son linealmente independientes o, en todo caso, su óptimo almacenamiento en el sentido del error cuadrático medio. Se comprueba que, efectivamente, los diez prototipos se convierten en estados estables de la red. Posteriormente introducimos un cierto nivel de ruido en los patrones originales, y presentamos los nuevos prototipos ruidosos a la red, observando que si el nivel de ruido es pequeño, la red converge en todos los casos al patrón original. Si se introduce un exceso de ruido, la red puede acabar en un estado erróneo, que se identifica porque no se corresponde con ninguno de los patrones almacenados en el entrenamiento, aunque en ocasiones su imagen puede recordar la de alguno de ellos.

El nivel de ruido tolerado por la red de Hopfield puede ser relativamente alto si se hace uso de un algoritmo de aprendizaje como el de la pseudoinversa, pues puede invertirse una fracción relativamente elevada de los píxeles, de modo que la imagen del dígito original resulta prácticamente irreconocible a nuestros ojos, y no por ello la red neuronal deja de converger al estado original correcto (Figura 4.4).

Otra experiencia interesante a realizar con la red entrenada mediante la pseudoinversa consiste en tapar una zona del dígito, por ejemplo la parte inferior. En nuestro caso tapar significa poner a cero todos los píxeles correspondientes a varias líneas inferiores del dígito. Podemos apreciar diversos casos en la Figura 4.5. Obsérvese que si ocultamos demasiados píxeles, la red no es capaz de distinguir correctamente el patrón original, acabando estancada en un estado errado.

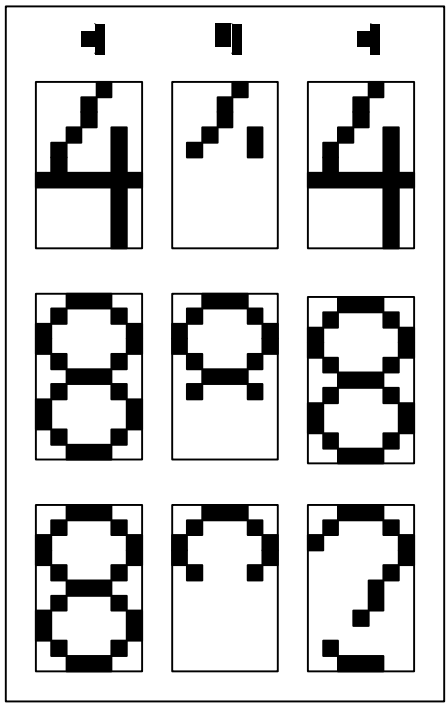


Figura 4.5 Columna a), digito original; columna b), digito con la parte inferior tapada, empleado como entrada a la red de Hopfield; columna c), estado final de la red tras varias iteraciones. En el último caso la red no tiene suficiente información y no reconoce el 8 (podría ser también un 2 o un 9)

4.4 APLICACION PRACTICA

La aplicación práctica de la red de hopfield de reconocimiento de caracteres hace uso de neuronas de tipo Ising ya que se utiliza valores (-1, +1) donde el +1 se representa con un color azul. Tenemos conjunto de imágenes formadas por las vocales, cada imagen esta compuesta por 7 x 5 píxeles, cada píxel puede estar encendido (+1) o apagado (-1)

Para iniciar con el reconocimiento de caracteres desde la ventana procedemos a dar click en la matriz para encender o apagar cada píxel (Figura 4.6).

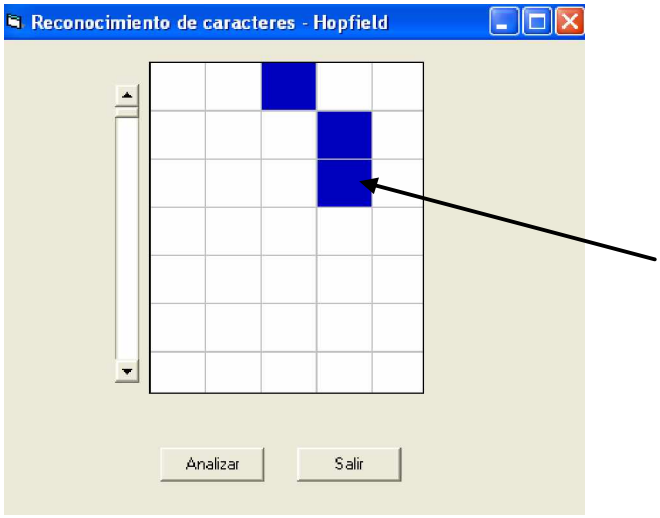


Figura 4.6 Red Hopfield, selección de píxeles

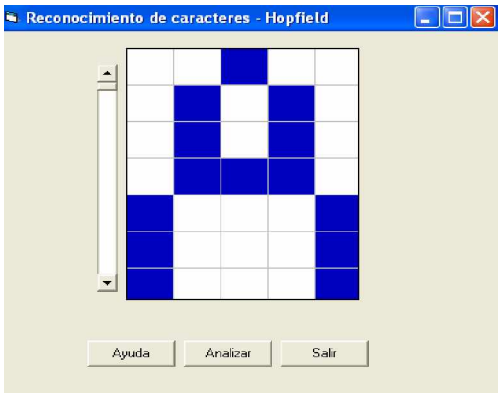


Figura 4.7 Red Hopfield, patrón original

Una vez terminado de introducir los patrones damos clic en el botón de analizar.

Al momento de dar clic en analizar empieza el entrenamiento (Figura 4.8).

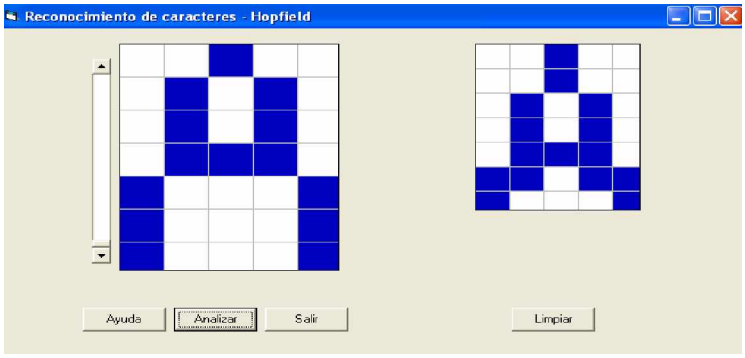


Figura 4.8 Red Hopfield, entrenamiento.

Cuando termina el proceso se presenta una ventana en la cual muestra el resultado (Figura 4.9).

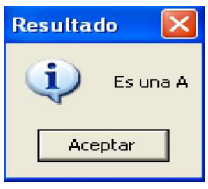


Figura 4.9 Red Hopfield, Ventana Resultado.

Ahora podemos apreciar como queda después de un cierto número de iteraciones (figura 4.10).

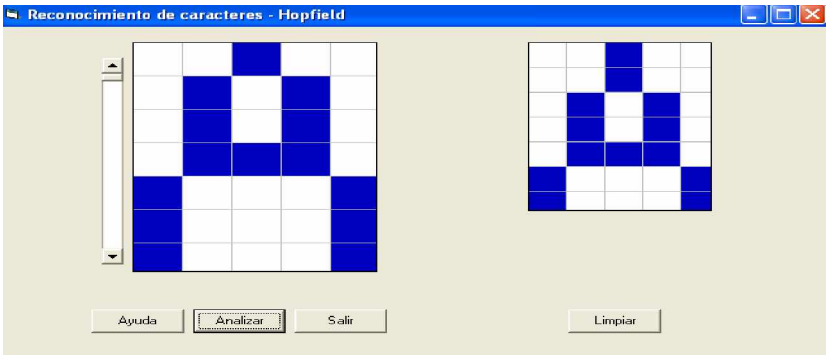


Figura 4.10 Red Hopfield, Ventana final después de entrenamiento.

Cuando entrenamos una red de hopfield y le introducimos cierto nivel de ruido la red converge a un estado estable (Figura 4.11).

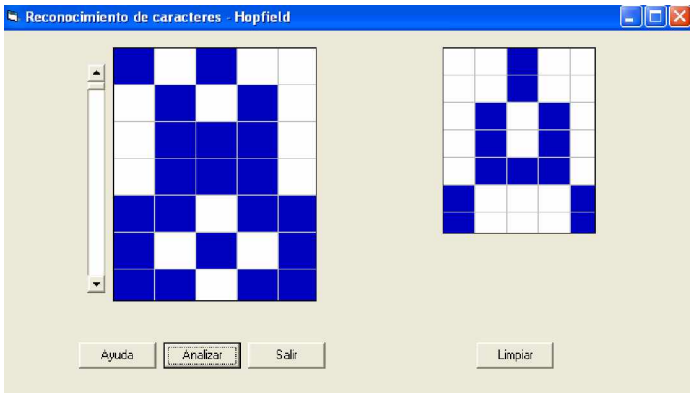


Figura 4.11 Red Hopfield, patrones con ruido.

4.5 CONCLUSION

En la actualidad siguen apareciendo modelos nuevos y obteniéndose nuevos resultados sobre los tradicionales. Los patrones presentados pueden diferir del memorizado en m componentes; estos componentes diferentes son interpretados como ruido presente en el

patrón de entrada. La red neuronal evoluciona hasta alcanzar el estado estable más próximo siendo este el estado estable memorizado sin ruido, así podemos eliminar el ruido en los patrones, es decir filtramos el ruido presente en las señales de entrada.

CONCLUSION FINAL

Como conclusión de nuestro estudio podemos decir que el mundo tiene mucho que ganar con las redes neuronales, ya que las personas que las utilizan ahora y las utilizarán en el futuro son aquellas que tienen que trabajar o analizar datos de cualquier tipo. Su habilidad para aprender mediante ejemplos las hace muy flexibles y poderosas. Además no es necesario crear un algoritmo para llevar a cabo una tarea específica. Ellas están bien adaptadas para los sistemas de tiempo real debido a su capacidad de respuesta rápida, la cual se debe a su arquitectura paralela.

Las Redes neuronales, con su notable habilidad para deducir significados de datos complicados o imprecisos pueden ser usadas para extraer patrones y detectar tendencias que son demasiado complejas para ser comprendidas por los humanos u otras técnicas computacionales.

Debe evitarse de las aplicaciones a ciegas de las redes neuronales, pues de ese modo no se alcanzara una solución tan eficaz como la que podría obtenerse con una metodología más elaborada.

Quizás el aspecto más excitante de las redes neuronales es la posibilidad de dotar a estas de conciencia. Existe un número de científicos que argumentan que la concienciación es una propiedad mecánica y que la conciencia de las redes neuronales es una posibilidad real.

Finalmente, las redes neuronales deberán emplearse donde ofrezcan ventajas; una red neuronal no tiene porque ser siempre la solución idónea. En este sentido debe tenerse en

cuenta una perspectiva global en la que se incluyan aspectos económicos, velocidad de desarrollo, robustez, fiabilidad, etc., y no solo el rendimiento del sistema. Cuando se tienen en cuenta todos estos factores, las redes neuronales suelen salir favorecidas, especialmente debido a la muy buena relación eficiencia/costo que ofrece.

Con esto hemos finalizado nuestra investigación sobre Redes Neuronales Artificiales cumpliendo a cabalidad todos los objetivos propuestos en nuestro Diseño de Monografía.

GLOSARIO

ANS: Artificial Neural Systems, Red o sistema neuronal artificial.

Centroide: o baricentro de un objeto X perteneciente a un espacio n-dimensional es la intersección de todos los hiperplanos que dividen a X en dos partes de igual n-volumen con respecto al hiperplano.

Determinista: Teoría que supone que la evolución de los fenómenos naturales está completamente determinada por las condiciones iniciales.

Emular: Imitar las acciones de otro procurando igualarlas e incluso excederlas.

Estocástico: Teoría estadística de los procesos cuya evolución en el tiempo es aleatoria, tal como la secuencia de las tiradas de un dado.

Inferencia: (inferir) Sacar una consecuencia o deducir algo de otra cosa

Inhibir: Suspender transitoriamente una función o actividad del organismo mediante la acción de un estímulo adecuado.

LVQ: Learning Vector Quantiazation, aprendizaje de vector de cuantificación.

Patrón: Modelo que sirve de muestra para sacar otra cosa igual

Pixel: Superficie homogénea más pequeña de las que componen una imagen, que se define por su brillo y color.

Sinapsis: Relación funcional de contacto entre las terminaciones de las células nerviosas.

VLSI: Very Large Scale Integration, es la integración en escala muy grande de sistemas de circuitos basados en transistores en circuitos integrados.

BIBLIOGRAFIA

SANZ MOLINA Alfredo; DEL BRIO Bonifacio Martín. Redes Neuronales y Sistemas Difusos, 2a Edición ampliada y revisada 2002. 406p.

http://www.tamps.cinvestav.mx/~castellanos/teaching/ 2007/2007RNA/2007RNA.html

http://dialnet.unirioja.es/servlet/articulo?codigo=2341596

http://ciberconta.unizar.es/LECCION/REDES/500.HTM

ftp://ftp.sas.com/pub/neural/FAQ.html

http://emsl/pnl.gov:2080/proj/neuron/neural/

http://www.kcl.ac.uk/neuronet/

http://www.cs.cmu.edu/groups

http://ewh.ieee.org/tc/nnc/

http://www.shef.ac.uk/psychology/gurney/notes/index.html

http://es.wikipedia.org/wiki/Red_neuronal_artificial

http://es.wikipedia.org/wiki/Perceptrón_multicapa

http://es.wikipedia.org/wiki/Perceptrón

http://www.electronica.com.mx/neural/

http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=45626&lngWId=1

http://www.generation5.org/content/2004/kohonenApplications.asp

http://www.generation5.org/content/1999/selforganize.asp