A_1	A_2	Clase
T	T	+
T	T	+
T	F	-
F	F	+
F	T	-
F	T	-

a. ¿Cuál es la entropía de este conjunto con respecto a la clase?

b. ¿Cuál es la ganancia de información de A2 en relación a estos ejemplos de entrenamiento.

14. Compare los métodos de evaluación de resultados de Aprendizaje Automático estudiados.

15. Implemente el ID3, el PRISM y el 1R con ejemplos de la vida real y compare los resultados obtenidos sobre un mismo conjunto de datos.

K52/13

2. Redes neuronales artificiales

2.1. Fundamentos de las Redes Neuronales

2.1.1. El modelo biológico

Una neurona es una célula viva y, como tal, contiene los mismos elementos que forman parte de todas las células biológicas. Además, contienen elementos característicos que

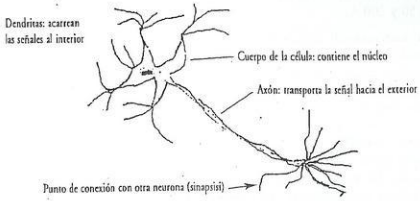


Figura 2.1. Estructura de una neurona biológica

las diferencian [Hilera José R., Martínez Victor J., 1994]. Una neurona consta de un cuerpo celular más o menos esférico de 5 a 10 micras de diámetro del que se desprende una rama principal o axón y varias ramas más cortas llamadas dendritas [Figura 2.1]. A su vez el axón presenta ramas en torno a su punto de arranque, y con frecuencia se ramifica extensamente cerca de su extremo.

Una de las características que diferencian a las neuronas del resto de las células vivas, es la capacidad que tienen éstas de comunicarse. En términos generales, las dendritas y el cuerpo celular reciben señales de entrada, el cuerpo celular las combina e integra y emite señales de salida. El axón transporta estas señales a los terminales axónicos, que se encargan de distribuir información a un nuevo conjunto de neuronas. Por lo general, una neurona recibe información de miles de otras neuronas y envía información a miles de otras más. Se calcula que en el cerebro humano existen del orden de 10^{15} conexiones.

Las señales a las que nos estamos refiriendo son de dos tipos de naturaleza: eléctrica y química. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, mientras que las señales que se transmiten entre los terminales axónicos de una neurona y las dendritas de otra es de origen químico; concretamente, se realiza mediante moléculas de sustancias transmisoras (neurotransmisores) que fluyen a través de unos contactos especiales, llamados sinapsis, que tienen la función de receptor y están localizados entre los terminales axónicos y las dendritas de las neuronas siguientes (espacio sináptico entre 50 y 200 Å).

La generación de las señales eléctricas está íntimamente relacionada con la composición de la membrana celular. El proceso de generación de las señales eléctricas de una neurona se puede simplificar del siguiente modo[Cardinali Daniel P., 1995]: la neurona, como todas las células, es capaz de mantener en su interior un líquido cuya composición difiere marcadamente de la composición del líquido del exterior. La diferencia más notable se da con relación a la concentración de iones sodio y potasio. El medio externo es una 10 veces más rico en iones sodio que el interno, mientras que el interno tiene una concentración de iones potasio 10 veces mayor que el externo. Esta diferencia de concentraciones de iones sodio y potasio genera una diferencia de potencial entre el interior y el exterior de la membrana celular del orden de los 70 mV (negativa en el interior de la membrana). Esto es lo que se conoce como el potencial de reposo de la célula nerviosa.

La llegada de señales procedentes de otras neuronas a través de las dendritas (recepción de neurotransmisores) actúa acumulativamente, bajando ligeramente el valor del potencial de reposo. Dicho potencial modifica la permeabilidad de la membrana de manera de que cuando llega a cierto valor crítico comienza una entrada masiva de iones sodio que invierten la polaridad de la membrana.

La inversión del voltaje de la cara interior de la membrana cierra el paso a los iones potasio hasta que se restablece el equilibrio en reposo. La inversión del voltaje, conocida como potencial de acción, se propaga a lo largo del axón y, a su vez, provoca la emisión de los neurotransmisores en los terminales axónicos.

Después de un período refractario, puede seguir un segundo impulso. El resultado de todo esto es la emisión por parte de la neurona, de trenes de impulsos cuya frecuencia varía en función (entre otros factores) de la cantidad de neurotransmisores recibidos.

Existen dos tipos de sinapsis: a) las sinapsis excitadoras, cuyos neurotransmisores provocan disminuciones de potencial en la membrana de la célula postsináptica, facilitando la generación de impulsos a mayor velocidad, y b) las sinapsis inhibidoras, cuyos neurotransmisores tienden a estabilizar el potencial de la membrana, dificultando la emisión de impulsos. Casi todas las neuronas reciben entradas procedentes de sinapsis excitadoras o inhibidoras. En cada instante algunas de ellas estarán activas y otras se hallarán en reposo; la suma de los efectos excitadores e inhibidores determina si la célula será o no estimulada: es decir, si emitirá o no un tren de impulsos y a qué velocidad.

2.1.2. Elementos de una red neuronal artificial (RNA)

Las redes neuronales son modelos que intentan reproducir el comportamiento del cerebro. Como tal modelo, realiza una simplificación, averiguando cuáles son los elementos relevantes del sistema. Una elección adecuada de sus características, más una estructura conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea.

Cualquier modelo de red neuronal consta de dispositivos elementales de proceso: las neuronas. A partir de ellas, se pueden generar representaciones específicas, de forma tal que un estado conjunto de ellas puede significar una letra, un número o cualquier otro objeto. Generalmente se pueden encontrar tres tipos de neuronas:

1) Aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de entrada.

2) Dicha información se transmite a ciertos elementos internos que se ocupan de su procesado. Es en las sinapsis y neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. Puesto que no tienen relación directa con la información de entrada ni con la de salida, estos elementos se denominan unidades ocultas.

3) Una vez finalizado el período de procesado, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema.

La neurona artificial [Figura 2.2] pretende mimetizar las características más importantes de las neuronas biológicas. Cada neurona i-ésima está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación, una función de activación o de transferencia que transforma el estado actual de activación en una señal de salida y_i. Dicha señal de salida es enviada a través de canales de comunicación unidireccionales a otras unidades de la red; en estos canales, la señal se modifica de acuerdo con la sinapsis (el peso w_{ij}) asociada a cada uno de ellos según una determinada regla. Las señales moduladas que han llegado a la unidad j-ésima se combinan entre ellas, generando así la entrada total Net_j.

$$Net_j = \sum_i y_i \, w_{ij}$$

La dinámica que rige la actualización de los estados de las unidades (evolución de la red neuronal) puede ser de dos tipos: modo asincrónico y modo sincrónico. En el primer caso, las neuronas evalúan su estado continuamente según les va llegando información y lo hacen de forma independiente. En el caso sincrónico la información también llega de forma continua pero los cambios se realizan simultáneamente, como si existiera un reloj interno que decidiera cuándo deben cambiar su estado. Los sistemas biológicos quedan probablemente entre ambas posibilidades.

KS2/13

2.1.2.1. *La neurona artificial.*

Si se tienen N unidades (neuronas), podemos ordenarlas arbitrariamente y designar a la j-ésima unidad como U_j. Su trabajo es simple y único y consiste en recibir las entradas de las células vecinas y calcular un valor de salida el cual es enviado todas las células conectadas con U_j.

En cualquier sistema que se esté modelando es útil caracterizar tres tipos de unidades: entradas, salidas y ocultas. Las unidades de entrada reciben señales desde el entorno; estas entradas (que son a la vez entradas a la red) pueden ser señales provenientes de sensores o de otros sectores del sistema (salidas de la red); estas señales pueden controlar directamente otros sistemas. Las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema; es decir, no tienen contacto con el exterior.

Se conoce como **capa** o **nivel** a un conjunto de neuronas cuyas entradas provienen de la misma fuente (que puede ser otra capa de neuronas). El estudio más detallado de la estructura de las redes neuronales será tratado más adelante, ahora se describirá el modelado de la neurona artificial.

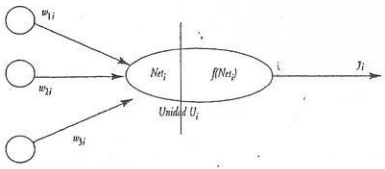


Figura 2.2 Estructura de una neurona artificial.

2.1.2.2. *Estado de activación*

Adicionalmente al conjunto de unidades, la representación necesita los estados del sistema en un tiempo t. Esto se especifica por un vector de números rea-

les $A(t)$, que representa el estado de activación del conjunto de unidades de procesamiento [HileraJosé R., Martínez Victor J., 1994]. Cada elemento del vector representa la activación de una unidad U_i en el tiempo t. La activación de una unidad en el tiempo t se designa por $a_i(t)$, es decir:

$$A_i(t) = (a_1(t), a_2(t), \ldots, a_N(t))$$

El procesamiento que realiza la red se ve como la evolución de un patrón de activación en el conjunto de unidades que lo componen a través del tiempo.

Todas las neuronas que componen la red se hallan en cierto estado. Los valores que puede tomar un estado pueden ser continuos o discretos. Además pueden ser limitados o ilimitados. Si son discretos, suelen tomar un conjunto pequeño de valores o bien valores binarios.

Finalmente es necesario saber qué criterios o reglas siguen las neuronas para alcanzar tales estados de activación. En principio esto va a depender de dos factores: Por un lado, es necesario tener idea del mecanismo de interacción entre las neuronas. El estado de activación estará fuertemente influenciado por tales interacciones, ya que el efecto que producirá una neurona sobre otra será proporcional a la fuerza, peso o magnitud de la conexión entre ambas. Por otro lado, la señal que envía cada neurona a sus vecinas dependerá de su propio estado de activación.

2.1.2.3. Conexiones entre neuronas

Las conexiones que unen a las neuronas que forman una RNA tienen asociado un peso, que es el que hace que la red adquiera conocimiento. Considérese como el valor de salida de una neurona i en un instante dado. Una neurona recibe un conjunto de señales que le dan información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión (sinapsis) entre la neurona i y la j está ponderada por un peso w_{ij}. Normalmente, como simplificación, se considera que el efecto de cada señal es aditivo, de tal forma que la entrada neta que recibe una neurona (potencial postsináptico) Net_j es la suma del producto de cada señal individual por el valor de la sinapsis que conecta ambas neuronas: $Net_j = \sum_{i=1}^{N} w_{ij}\, y_i$.

Esta regla muestra el procedimiento a seguir para combinar los valores de entrada a una unidad con los pesos de las conexiones que llegan a esa unidad y es conocida como regla de propagación.

Suele utilizarse una matriz W con todos los pesos w_{ij} que reflejan la influencia que sobre la neurona j tiene la neurona i. W es un conjunto de elementos positivo, negativos o nulos. Si w_{ij} es positivo indica que la interacción entre las neuronas i y j es excitadora; es decir, siempre que la neurona i esté activada, la neurona j recibirá una señal de i que tenderá a activarla. Si w_{ij} es negativo la sinapsis será inhibitoria. En este caso si i está activada, enviará una señal j que tenderá a desactivar a esta. Finalmente, si w_{ij} es 0 se supone que no hay conexión entre ambas.

2.1.2.4. Función de transferencia o activación

Entre las unidades o neuronas que forman una red neuronal artificial existe un conjunto de conexiones que unen unas con otras. Cada unidad transmite señales a aquellas que están conectadas con su salida. Asociada con cada unidad U_i hay una función de activación o de transferencia $f(Net_i)$ que transforma la entrada neta de la neurona en una señal de salida $y_i(t) = f(Net_i(t))$. El vector que contiene las salidas de todas las neuronas en un instante t es $Y(t) = (f_1(Net_1(t)), f_2(Net_2(t)), \ldots, f_N(Net_N(t)))$.

Además, normalmente la función de activación no está centrada en el origen del eje de abscisas, sino que existe cierto desplazamiento debido a las características internas de la neurona y que no es igual en todas ellas. Este valor se denota como θ y representa el umbral de activación de la neurona i.

$$y_i(t+1) = f(Net_i - \Theta_i) = f(\sum_{j=1}^{N} w_{ij}\, y_i\,(t) - \Theta_i)$$

Existen cuatro funciones de activación típicas que determinan distintos tipos de neuronas [Hilers José R., Martínez Victor J., 1994]:

- Función escalón
- Función lineal y mixta
- Sigmoidal
- Función Gaussiana.

La función escalón o umbral solo se utiliza cuando las salidas de la red son binarias (dos valores posibles). La salida de una neurona se activa sólo cuando la entrada neta es mayor o igual que cierto valor umbral (la función puede estar desplazada sobre los ejes). Las funciones mixta y sigmoidal son las más apropiadas cuando queremos como salida la información analógica. Veamos con más detalle las distintas funciones:

Neurona de función escalón

La forma más fácil de definir la activación de una neurona es considerar que ésta es binaria. La función de trasferencia escalón se asocia a neuronas binarias en las cuales cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1; si es menor, la activación es 0 o -1. Las redes formadas por este tipo de neuronas son fáciles de implementar en hardware, pero a menudo sus capacidades están limitadas.

Neurona de función lineal y mixta.

La función lineal o identidad responde a la expresión $f(x) = x$. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la activación se define como 0 o -1. Si dicha suma es mayor que un límite superior, entonces la activación es 1. Si la suma de entradas está entre ambos límites entonces la activación se define como una función lineal de la suma de las entradas.

Neurona de función continua (sigmoidal).

Cualquier función definida simplemente en un intervalo de posibles valores de entrada, con un incremento monotónico y que tenga ambos límites superiores e inferiores (por ejemplo las funciones sigmoidal o arcotangente), podrá realizar la función de activación de forma satisfactoria.

Con la función sigmoidal, para la mayoría de los valores del estímulo de entrada (variable independiente), el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que en la mayoría de los casos, el valor de salida este comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal (o cualquier otra función similar) es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando x es 0. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para las funciones escalón, con la ventaja de que la derivada está definida en todo el intervalo. La función escalón

K52/13

no tiene derivada definida en el punto de transición y esto no ayuda a los métodos de aprendizaje en los que se usan derivadas.

Neurona con función de transferencia gaussiana.

Los centros y anchura de estas funciones pueden ser adaptados lo cual las hace más adaptativas que las funciones sigmoidales. Mapeos que suelen requerir dos niveles ocultos utilizando neuronas con funciones de transferencia sigmoidales, algunas veces se pueden realizar con un solo nivel en redes con funciones gaussianas.

Para simplificar la expresión de la salida de una neurona i, es habitual considerar la existencia de una neurona ficticia con valor de salida unidad asociada a la entrada de cada neurona i mediante una conexión con peso de valor $-\Theta_i$. De esta forma la expresión de salida quedará:

$$y_i(t+1) = f(Net_i - \Theta_i) = f(\sum_{j=1}^{N} w_{ij}\, y_j(t) - \Theta_i) = f(\sum_{j=0}^{N} w_{ij}\, y_i(t))$$

2.1.2.5. Regla de aprendizaje

Existen muchas definiciones de concepto general de aprendizaje, una de ellas podría ser: *La modificación del comportamiento inducido por la interacción con el entorno y cómo resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos.* Esta definición fue enunciada muchos años antes de que surgieran las redes neuronales, sin embargo puede ser aplicada también a los procesos de aprendizaje de estos sistemas. Biológicamente, se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas: es decir, el conocimiento se encuentra en las sinapsis. En el caso de las redes neuronales artificiales, se puede considerar que el conocimiento se encuentra representado en los pesos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad, puede decirse que se aprende modificando los valores de los pesos de la red.

Al igual que el funcionamiento de una red depende del número de neuronas de las que disponga y de cómo estén conectadas entre sí, cada modelo dispone de su o sus propias técnicas de aprendizaje.

2.1.2.6. *Representación vectorial*

En ciertos modelos de redes neuronales, se utiliza la forma vectorial como herramienta de representación de algunas magnitudes. Si consideramos una red formadas por varias capas de neuronas idénticas, podemos considerar las salidas de cierta capa de n unidades como un vector n-dimensional $Y = (y_1, y_2, \ldots, y_n)$. La entrada neta de la j-ésima unidad se puede escribir en forma de producto escalar del vector de entradas por el vector de pesos. Cuando los vectores tienen igual dimensión, este producto se define como la suma de los productos de los componentes de ambos vectores.

$$net_j = \sum_{i=1}^{N} w_{ij}\, y_i$$

En donde N representa el número de conexiones de la j-ésima unidad. La ventaja de la notación vectorial es que la ecuación anterior se puede escribir de la forma:

$$net_j = WY$$

2.1.3. Estructura de una RNA

2.1.3.1. *Niveles o capas de neuronas*

La distribución de neuronas dentro de la red se realza formando niveles o capas de un número determinado de neuronas cada una. A partir de su situación dentro de la red, se pueden distinguir tres tipos de capas:

- De entrada: Es la capa que recibe directamente la información proveniente de las fuentes externas de la red.
- Ocultas: Son internas a la red y no tiene contacto directo con el exterior. El número de niveles ocultos puede estar entre 0 y un número elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto a su número, las distintas topologías de redes neuronales.
- De salida: Transfieren información de la red hacia el exterior.

Se dice que una red es totalmente conectada si todas las salidas desde un nivel llegan a todos y cada uno de los nodos del nivel siguiente.

2.1.3.2. *Formas de conexión entre neuronas*

La conectividad entre los nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas [Figura 2.3]. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo (conexión autorrecurrente).

Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles

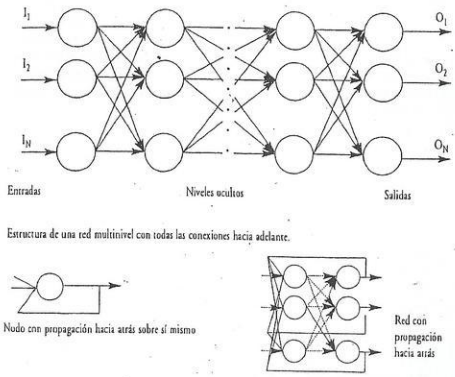


Figura 2.3. Distintas conexiones entre neuronas

precedentes, la red se escribe como de propagación hacia delante, cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de propagación hacia atrás. Las redes de propagación hacia atrás que tienen lazos cerrados son sistemas recurrentes.

2.2. *Características de las redes neuronales.*

Existen 4 aspectos que caracterizan una red neuronal: su topología, el mecanismo de aprendizaje, el tipo de asociación realizada entre la información de entrada y de salida y la forma de representación de estas informaciones[Hilera José R., Martínez Victor J., 1994].

2.2.1. Topología de las redes neuronales.

La topología o arquitectura de las redes neuronales consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red. En este sentido, los parámetros fundamentales de red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas.

Cuando se realiza una clasificación de las redes en términos topológicos, se suele distinguir entre las redes con una sola capa o nivel de neuronas y las redes con múltiples capas (2, 3, etc.). A continuación se analizarán ambos tipos de redes.

2.2.1.1. Redes Monocapa (1 capa)

En las redes monocapa [Tabla 2.1.], como la red de Hopfield y la red Brain State In-a-Box, se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red. También pueden existir conexiones autorrecurrentes (salida de una neurona conectada a la entrada), aunque en algún modelo, como el Hopfiled, esta recurrencia no se utiliza.

Una topología equivalente a las de las redes de una capa es la denominada topología crossbar (barras cruzadas). Una red de este tipo (por ejemplo, la red Learning Matrix) consiste en una matriz de terminales (de entrada y salida) o barras que se cruzan en unos puntos a los que se les asocia un peso. Esta representación crossbar suele utilizarse como etapa de transición cuando se pretende implementar físicamente una red monocapa, puesto que es relativamente sencillo desarrollar como hardware una estructura como la indicada (por ejemplo, las barras cruzadas serían cables y los puntos de conexión resistencias cuyos valores representarían los pesos de la red).

Finalmente hay que indicar que las redes monocapa se utilizan típicamente en tareas relacionadas con lo que se conoce como autoasociación; por ejemplo, para regenerar informaciones de entrada que se presentan a la red incompletas o distorsionadas.

Tipos de conexiones		Modelo de red
Conexiones laterales explícitas	Conexiones autorrecurrentes	Brain State-In-aBox
		Additive Grossberg (AG)
		Shunting Grossberg (SG)
		Optimal Linear Associative Memory
	No autorrecurrentes	Hopfield
		Boltzman Machine
		Cauchy Machine
Crossbar		Learning Matrix (LM)

Tabla 2.1. Redes Monocapa

2.2.1.2. Redes Multicapa

Las redes multicapa [Tabla 2.2] son aquellas que disponen de conjuntos de neuronas agrupadas en varios (2, 3, etc...) niveles o capas. En estos casos, una forma para distinguir la capa a la que pertenece una neurona, consistiría en fi-

jarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida. Normalmente, todas las neuronas de una capa reciben señales de entrada de otra capa anterior, más cercana a las entradas de la red y envían las señales de salida a una capa posterior, más cercana a las salidas de la red. A estas conexiones se las denomina conexiones hacia adelante o feedforward.

Sin embargo, en un gran número de estas redes también existe la posibilidad de conectar las salidas de las neuronas de capas posteriores a las entradas de las capas anteriores, a estas conexiones se las denomina conexiones hacia atrás o feedback.

Estas dos posibilidades permiten distinguir entre dos tipos de redes con múltiples capas: las redes con conexiones hacia adelante o redes feedforward y las redes que disponen de conexiones tanto hacia adelante como hacia atrás o redes feedforward/feedback.

2.2.1.2.1. Redes con conexiones hacia adelante (feedforward)

En las redes feedforward todas las señales neuronales se propagan hacia adelante a través de las capas de la red. No existen conexiones hacia atrás (ninguna salida de neuronas de la capa i se aplica a la entrada de neuronas de capas i-1,i-2,...), y normalmente tampoco autorrecurrentes (salida de una neurona conectada a su propia entrada), ni laterales (salida de una neurona conectada a entradas de neuronas de la misma capa), excepto en el caso de los modelos de red propuestos por Kohonen denominados Lerning vector quantizer (LVQ) y Topology Preserving Map (TPM), en las que existen unas conexiones implícitas muy particulares entre las neuronas de la capa de salida.

Las redes feedforward más conocidas son: Perceptrón, Adaline, Madaline, Linear Adaptive Memory (LAM), Drive Reinforcment, Backpropagation. Todas ellas son especialmente útiles en aplicaciones de reconocimiento o clasificación de patrones.

2.2.1.2.2. Redes con conexiones hacia adelante y hacia atrás (feedforward/feedback)

En este tipo de redes circula información tanto hacia adelante (forward) como hacia atrás (backward) durante el funcionamiento de la red. Para que esto sea posible existen conexiones feedforward y conexiones feedback entre las neuronas.

En general, excepto el caso particular de las redes Cognitron y Neocognitron, suelen ser bicapa existiendo por lo tanto dos conjuntos de pesos: los correspondientes a las conexiones feedforward de la primera capa (capa de entrada) hacia la segunda (capa de salida) y los de las conexiones feedback de la segunda a la primera. Los valores de los pesos de estos tipos de conexiones no tienen porqué coincidir, siendo diferentes en la mayor parte de los casos.

Este tipo de estructura (bicapa) es particularmente adecuada para realizar una asociación de una información o patrón de entrada (en la primera capa) con otra información o patrón de salida en la segunda capa (heteroasociación), aunque también pueden ser utilizadas para la clasificación de patrones.

Algunas redes de ese tipo tienen un funcionamiento basado en lo que se conoce como resonancia, de tal forma que las informaciones en la primera y segunda capas interactúan entre sí hasta que alcanzan un estado estable. Este funcionamiento permite un mejor acceso a las informaciones almacenadas en la red.

Los dos modelos de red feedforward/feedback de dos capas más conocidos son la red ART (adpative resonance theory) y la red BAM (Bidirectional Asociative Memory).

También en este grupo de redes existen algunas que tienen conexiones laterales entre neuronas de la misma capa. Estas conexiones se diseñan como conexiones excitadoras (con peso positivo), permitiendo la cooperación entre neuronas, o como inhibidoras (peso negativo), estableciéndose una competición entre las neuronas correspondientes. Una red de este tipo que, además, dispone de conexiones autorrecurrentes es la denominada CABAM.

Finalmente, hay que comentar la existencia de un tipo de red feedforward-/feedback multicapa muy particular denominada Neocognitron, en la que las neuronas se disponen en planos superpuestos (capas bidimensionales), lo cual permite que puedan eliminarse las variaciones geométricas (tamaños, giros, desplazamientos) o distorsiones que presenten las informaciones o patrones de entrada a la red.

Nº de capas	Tipos de Conexiones		Modelo de Red
2 Capas	Conexiones hacia adelante		Adaline/Madaline
			Perceptrón
			Linear Assoc. Rew. Penalty
			Lin assoc. Mem.
			Optimal LAM
			Drive Reinforcement (DR)
		Conexiones laterales	LVQ
		Implícitas y autorrecurrentes	TPM
	Conex. Adelante/atrás	Sin laterales	BAM
			Adaptive BAM
			Temporal Assoc.Mem
			Fuzzy Assoc. Mem.
		Con laterales y autorrec.	Competitive Adpative Bam
			ART
3	Conex. hacia adelante	Sin laterales	Adpaptive Heur. Critic
			Boltzmann/Cuachy Machine
		Con laterales	Cunterpropagation
			Bol/Cau. Machine
	Conex. adelante atrás y laterales		Bol/Cau. Machine
N	Conex. hacia adelante		Backpropagation
	Adelante/atrás (capas bidimensionales)		Cognitron/Neocognitron

Tabla 2.2. Redes neuronales multicapa

2.2.2. Mecanismo de Aprendizaje

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los sistemas biológicos existe una continua creación y destrucción de conexiones. En los modelos de redes neuronales artificiales la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de 0. De la misma forma, una conexión se destruye cuando su peso pasa a ser 0.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por tanto se puede afirmar que este proceso ha terminado (la red ha aprendido) cuando los valores de los pesos permanecen estables dwi/dt=0.

Un aspecto importante respecto al aprendizaje en las redes neuronales es el conocer cómo se modifican los valores de los pesos; es decir, cuáles son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando se pretende que la red aprenda una nueva información.

Estos criterios determinan lo que se conoce como la **regla de aprendizaje de la red**. De forma general se suelen considerar dos tipos de reglas: las que responden al aprendizaje supervisado, y las correspondientes al aprendizaje no supervisado.

La diferencia fundamental entre ambos tipos estriba en la existencia o no de una agente externo (supervisor) que controle el proceso de aprendizaje de la red.

Otro criterio que se puede utilizar para diferenciar las reglas de aprendizaje se basa en considerar si la red puede aprender durante su funcionamiento habitual o si el aprendizaje supone la desconexión de la red; es decir, su deshabilitación hasta que el proceso finalice. En el primer caso se trataría de un aprendizaje on line, mientras que el segundo se conoce como aprendizaje off line.

Cuando el aprendizaje es off line, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y otro de test o prueba que serán utilizados en la fase correspondiente. En las redes con aprendizaje off line, los pesos de las conexiones permanecen fijos después de que termina la etapa de entrenamiento de la red. Debido precisamente a su carácter estático, estos sistemas no presentan problemas de estabilidad en su funcionamiento.

En las redes con aprendizaje on line no se distingue entre fase de entrenamiento y de operación, de tal forma que los pesos varían dinámicamente siempre que se presente una nueva información al sistema. En este tipo de redes, debido al carácter dinámico de las mismas, el estudio de la estabilidad suele ser una aspecto fundamental.

2.2.2.1. Redes con aprendizaje supervisado

El aprendizaje supervisado [Tabla 2.3.] se caracteriza por un entrenamiento controlado por una agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor comprueba la salida de la red y en el caso de que ésta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada.

En este tipo de aprendizaje se suelen considerar, a su vez, tres formas de llevarlo a cabo que dan lugar a los siguientes aprendizajes supervisados:

Aprendizaje por corrección de error.

Aprendizaje por refuerzo.

Aprendizaje estocástico.

2.2.2.1.1. Aprendizaje por corrección de error

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red; es decir, en función del error cometido en la salida [Figura 2.4].

Una regla o algoritmo simple de aprendizaje por corrección de error podría ser el siguiente:

$$\Delta w_{ij} = \alpha\, y_i\, (d_j - y_j)$$

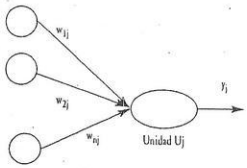


Figura 2.4. Aprendizaje por corrección de error

Siendo:

Δwij: Variación en el peso de la conexión entre las neuronas i y j ($\Delta wij = wij^{actual} - wij^{anterior}$).

yi: Valor de salida de la neurona i.

dj: Valor de salida deseado para la neurona j.

yj: Valor de salida obtenido en la neurona j.

a: Factor de aprendizaje ($0<\alpha<=1$) que regula la velocidad del aprendizaje.

Un ejemplo de este tipo de algoritmo lo constituye la regla de aprendizaje del Perceptrón, utilizada en el entrenamiento de la red del mismo nombre que desarrolló Rosenblatt en 1958. Sin embargo, existen otros algoritmos más evolucionados que éste, que presenta limitaciones, como el no considerar la magnitud del error global cometido durante el proceso completo de aprendizaje de la red, considerando únicamente los errores individuales (locales) correspondientes al aprendizaje de cada información por separado.

Un algoritmo muy conocido que mejora el del perceptrón y permite un aprendizaje más rápido y un campo de aplicación más amplio es el propuesto por Widrow y Hoff en 1960 denominado regla delta o regla del mínimo error cuadrado (LMS Error:Least Mean Squared Error), que se aplicó en las redes desarrolladas por los mismos autores, conocidas como Adaline (Adpative Linear Element), con una única neurona de salida y Madaline con varias neuronas de salida.

Widrow y Hoff definieron una función que permitía cuantificar el error global cometido en cualquier momento durante el proceso de entrenamiento de la red.

Este error medio se expresa de la siguiente forma:

$$\text{Error}_{\text{global}} = \frac{1}{2P} \sum_{k=1}^{P} \sum_{j=1}^{N} (y_j^{(k)} - d_j^{(k)})^2$$

Siendo:

N: Número de neuronas de salida (en el caso de Adaline N=1).

P: Número de informaciones que debe aprender la red.

$\frac{1}{2} \sum_{j=1}^{N} (y_j^{(k)} - d_j^{(k)})^2$: Error cometido en el aprendizaje de la información k-ésima.

Por tanto, de lo que se trata es de encontrar unos pesos para las conexiones de la red que minimicen esta función de error. Para ello, el ajuste de los pesos de las conexiones de la red se puede hacer de forma proporcional a la variación relativa del error que se obtiene al variar el peso correspondiente.

$$\Delta w_{ij} = k \frac{\partial Error_{global}}{\partial w_{ij}}$$

Mediante este procedimiento, se llegan a obtener un conjunto de pesos con los que se consigue minimizar el error medio.

Otro algoritmo de aprendizaje por corrección de error lo constituye el denominado regla delta generalizada o algoritmo de retropropagación del error (error backpropagation) también conocida como regla LMS (Least mean square error) multicapa. Se trata de una generalización de la regla delta para poder aplicarla a redes con conexiones hacia adelante con capas ocultas.

Estas redes multicapa pueden utilizarse en muchas más aplicaciones que las ya conocidas Perceptrón, Adaline y Madaline, pero su proceso es mucho más lento, debido a que durante el mismo se debe explorar el espacio de posibles formas de utilización de las neuronas de las capas ocultas; es decir, se debe establecer cuál va a ser su papel en el funcionamiento de la red.

K52/13

Existe también una versión recurrente del algoritmo backpropagation que se suele utilizar en redes multicapa que presentan conexiones recurrentes con el fin de que estas redes aprendan la naturaleza temporal de algunos datos.

Para concluir con los algoritmos por corrección de error, hay que mencionar que también se utilizan en algunas redes monocapa con conexiones laterales y autorrecurrentes como la red Brain State-in -a-Box (BSB) introducida por Anderson, Ritz y Jones en 1977. Aunque en una primera etapa el aprendizaje de esta red es sin supervisión, se suelen refinar los valores de los pesos de las conexiones mediante un aprendizaje por corrección de error basado en una adaptación de la regla de Widrow-Hoff.

2.2.2.1.2. Aprendizaje por refuerzo

Se trata de una aprendizaje supervisado, más lento que el anterior, que se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado; es decir, de no indicar durante el entrenamiento exactamente la salida que se desea que proporcione la red ante una determinada entrada.

En el aprendizaje por refuerzo la función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada (éxito=+1 o fracaso=-1), y en función de ello se ajustan los pesos basándose en un mecanismo de probabilidades. Se podría decir que en este tipo de aprendizaje la función del supervisor se asemeja más a la de un crítico (que opina sobre la respuesta de la red) que a la de un maestro (que indica a la red la respuesta concreta de que debe generar), como ocurriría en el caso de supervisión por corrección de error.

Un ejemplo de algoritmo por refuerzo lo constituye el denominado Linear Reward-Penalty o L_{R-P} (algoritmo lineal con recompensa y penalización) presentado por Narendra y Thathacher en 1974. Este algoritmo ha sido ampliado por Barto y Anand, quienes en 1985 desarrollaron el denominado Associative Reward Penalty que se aplica en redes con conexiones hacia delante de dos capas cuyas neuronas de salida presentan una función de activación estocástica.

Otro algoritmo por refuerzo es el conocido como Adapive Heuristic Critic, introducido por Barto, Sutton y Anderson en 1983 que se utiliza en redes feedforward de tres capas especialmente diseñadas para que una parte de la red sea capaz de generar un valor interno de refuerzo que es aplicado a las neuronas de salida de la red.

2.2.2.1.3. Aprendizaje estocástico

Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidades.

En el aprendizaje estocástico se suele hacer una analogía con la termodinámica. Se asocia a la red neuronal con un sólido físico que tiene un cierto estado energético. En el caso de la red, la energía representa el grado de estabilidad de la misma, de tal forma que el estado de mínima energía correspondería a una situación en la que los pesos de las conexiones consiguen que su funcionamiento sea el que más se ajuste al comportamiento deseado.

Según lo anterior, el aprendizaje consistiría en realizar un cambio aleatorio de los valores de los pesos y determinar la energía de la red (habitualmente la función energía es una función denominada de Lyapunov). Si la energía es menor después del cambio: es decir, si el comportamiento de la red se acerca al deseado, se acepta el cambio. Si, por el contrario, la energía no es menor, se acepta el cambio según una distribución de probabilidades.

Una red que utiliza este tipo de aprendizaje es la conocida como Boltzman Machine ideada por Hinton, Ackley y Sejnowsky en 1984 que lo combina con el aprendizaje Hebbiano o con el aprendizaje por corrección de error (como la regla delta). La Boltzman machine es una red con diferentes topologías alternativas pero siempre con unas neuronas ocultas que permiten, mediante un ajuste probabilístico, introducir un ruido que va decreciendo durante el proceso de aprendizaje para escapar de los mínimos relativos de la función de energía favoreciendo la búsqueda del mínimo global.

El procedimiento de utilizar para escapar de mínimos locales suele denominarse **simmulated annealing** y su combinación con las asignación probabilística mediante la capa oculta es lo que se conoce como **aprendizaje estocástico**. La idea de todo esto es asemejar la red con un sólido físico que inicialmente presenta una alta temperatura (ruido) y que se va enfriando gradualmente hasta alcanzar el equilibro térmico (mínima energía).

Existe otra red basada en este tipo de aprendizaje denominado Cauchy Machine desarrollada por Szu en 1986 que es un refinamiento de la anterior y que emplea un procedimiento más rápido de búsqueda del mínimo global y una función de probabilidad diferente (la distribución de probabilidad de Cauchy frente a la de Boltzman utilizada en la anterior).

Tipo de Aprendizaje Supervisado		Modelo de Red
Aprendizaje por corrección de Error	OFF Line	Perceptrón Adaline/Madaline Backpropagation Brain State-in-a-Box Counterpropagation
Aprendizaje por Refuerzo	ON Line	Linear Reward Penalty Associative Rew. Penalty Adaptive Heuristic Critic
Aprendizaje Estocástico	OFF Line	Boltzmann Machine Cauchy Machine

Tabla 2.3. Tipos de redes con aprendizaje supervisado más conocidas

2.2.2.2. Redes con aprendizaje no supervisado

Las redes con aprendizaje no supervisado [Tabla 2.4.] (también conocido como autosupervisado) no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas redes son capaces de autoorganizarse.

Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presenten en su entrada. Puesto que no hay un supervisor que indique a la red la respuesta que debe generar ante una entrada concreta, cabría preguntarse qué es lo que la red genera en estos casos. Existen varias posibilidades en cuanto a la interpretación de la salida de estas redes, que dependen de su estructura y del algoritmo de aprendizaje empleado.

En algunos casos, la salida representa el grado de familiaridad entre la información que se le está presentando a la entrada y las informaciones que se le han

mostrado hasta entonces. En otro caso, podría realizar una clusterización (clustering) o establecimiento de categorías, indicando la red a la salida a qué categoría pertenece la información presentada a la entrada, siendo la propia red la que debe encontrar las categorías apropiadas a partir de correlaciones entre las informaciones presentadas. Una variación de esta categorización es el prototipado. En este caso la red obtiene ejemplares o prototipos representantes de las clases a las que pertenecen las informaciones de entrada.

También el aprendizaje sin supervisión permite realizar una codificación de los datos de entrada, generando a la salida una versión codificada de la entrada, con menos bits, pero manteniendo la información relevante de los datos.

Finalmente, algunas redes con aprendizaje no supervisado realizan un mapeo de caracterización (feature mapping), obteniéndose en las neuronas de salida una disposición geométrica que representa un mapa topográfico de las características de los datos de entrada, de tal forma que si se presentan a la red informaciones similares siempre sean afectadas neuronas de salida próximas entre sí, en la misma zona del mapa.

En cuanto a los algoritmos de aprendizaje no supervisado, en general se suelen considerar dos tipos que dan lugar a los siguientes aprendizajes:

- Aprendizaje Hebbiano.
- Aprendizaje Competitivo y Cooperativo. En el primer caso, normalmente se pretende medir la familiaridad o extraer características de los datos de entrada, mientras que el segundo suele orientarse hacia la clusterización o clasificación de dichos datos.

2.2.2.2.1. Aprendizaje hebbiano

Este tipo de aprendizaje se basa en el siguiente postulado formulado por Donald O. Hebb en 1949: "Cuando un axón de una celda A está suficientemente cerca como para conseguir excitar una celda B y repetida o persistentemente toma parte en su activación, algún proceso de crecimiento o cambio metabólico tiene lugar en una o ambas celdas, de tal forma que la eficiencia de A, cuando la celda a activar es B, aumenta". Por celda, Hebb entiende un conjunto de neuronas fuertemente conexionadas a través de una estructura compleja. La eficiencia podría identificarse con la intensidad o magnitud de la conexión: es decir, con el peso.

K52/13

Se puede decir, por tanto, que el aprendizaje hebbiano consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la correlación (multiplicación en el caso de valores binarios +1 y -1) de los valores de activación (salidas) de las dos neuronas conectadas:

$$\Delta w_{ij} = y_i y_j$$

Esta expresión responde a la idea de Hebb, puesto que si las dos unidades son activas (positivas), se produce un refuerzo de la conexión. Se trata de una regla de aprendizaje no supervisado, pues la modificación de los pesos se realiza en función de los estados (salidas) de las neuronas obtenidos tras la presentación de cierto estímulo (entrada), sin tener en cuenta si se deseaba obtener o no esos estados de activación.

Este tipo de aprendizaje fue empleado por Hopfield en la conocida red que lleve su nombre (Red Hopfield), introducida en 1982 y muy extendida en la actualidad debido principalmente a la relativa facilidad en su implementación en circuitos integrados VLSI.

Grossberg lo utilizó en 1968 en la red denominada Additive Grossberg y en 1973 en la red Shunting Grossberg. En ambos casos se trataba de redes de una capa con conexiones laterales y autorrecurrentes.

La red feedforward llamada Learning Matrix (LM), desarrollada por Steinbuch en 1961 también aprendía mediante correlación hebbiana. En esta red, las neuronas se conectaban en forma matricial asignándose un peso a cada punto de conexión, peso que se obtenía mediante una regla tipo hebbiana.

Otras redes que utilizan el aprendizaje hebbiano son: la red feedforward/feedback de 2 capas denominada Bidirectional Associative Memory (BAM), desarrollada por Kosko en 1988, la red desarrollada por Amari en 1972 llamada Temporal Associative Memory (TAM), con la misma topología que la anterior, pero ideada para aprender la naturaleza temporal de las informaciones que se le muestran (para recordar valores anteriores en el tiempo): la red feedforward de dos capas conocida Linear Associative Memory (LAM) introducida por Anderson en 1968 y refinada por Kohonen, y la red Optimal Linear Associative Memory (OLAM) desarrollada de forma independiente por Wee en 1968 y por Kohonen y Ruohonen en 1973.

Existen muchas variaciones del aprendizaje hebbiano: por ejemplo, Sejnowski en 1977 utilizó la correlación de la covarianza de los valores de activación de las neuronas, Sutton y Barto en 1981 utilizaron la correlación del valor medio de una neurona con la varianza de la otra. Klopf en 1986 propuso una correlación entre las variaciones de los valores de activación en dios instantes de tiempo sucesivos, aprendizaje el que denominó drive-reinforcement y que utilizó ene redes del mismo nombre con topología feedforward de dos capas.

Otra versión de este aprendizaje es el denominado hebbiano diferencial que utiliza la correlación de las derivadas en el tiempo de las funciones de activación de las neuronas. El aprendizaje hebbiano diferencial es utilizado en la red feedforward/feedbak de dos capas denominada ABAM (Adaptive Bidirectional Associative Memory) introducida por Kosko en 1987. También este autor en 1987 presentó una red con la misma topología que la anterior, pero utilizando otra versión de este aprendizaje, el denominado aprendizaje hebbiano difuso; esta red tenía por nombre Fuzzy Associative Memory (FAM), y se basaba en la representación de las informaciones que debía aprender la red en forma de conjuntos difusos.

Finalmente, hay que comentar la existencia de las redes basadas en mecanismos de aprendizaje que resultan de la combinación de la correlación hebbiana con algún otro método, como es el caso de las redes Boltzmann, Machine y Cauchy Machine (mencionadas en el apartado 2.1.3), que los combinan con el ya comentado simulated annealing. También la red feedforward de tres capas llamada counterpropagation (CPN), desarrollada por Hecht-Nielsen en 1987, utiliza un aprendizaje que es combinación del hebbiano y de un tipo de aprendizaje competitivo introducido por kohonen denominado learning vector quantization (LVQ).

2.2.2.2.2. Aprendizaje competitivo y cooperativo

En las redes con aprendizaje competitivo (y cooperativo), suele decirse que las neuronas compiten (y cooperan) unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje, se pretende que cuando se presente a la red cierta información de entrada, sólo una de las neuronas de salida de la red, o una por cierto grupo de neuronas, se active (alcance su valor de respuesta máximo). Por tanto, las neuronas compiten por activarse, quedando finalmente una, o una por grupo, como neurona vencedora (winner take all unit), quedando anulado el resto de las neuronas, las que son forzadas al valor de respuesta mínimo.

La competencia entre neuronas se realiza en todas las capas de la red, existiendo en estas neuronas conexiones recurrentes de autoexcitación y conexiones de inhibición (signo negativo) por parte de neuronas vecinas. Si el aprendizaje es cooperativo, estas conexiones con las vecinas serán de excitación (signo positivo).

El objetivo de este aprendizaje es categorizar (clusterizar) los datos que se introducen en la red. De esta forma, las informaciones similares son clasificadas formando parte de la misma categoría, y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado, a través de las correlaciones entre los datos de entrada.

Una forma de aplicar este tipo de aprendizaje fue propuesta por Rumelhart y Zisper en 1985 quienes utilizaban redes mutilcapa dividiendo cada capa en grupos de neuronas, de tal forma que éstas disponían de conexiones inhibidoras con otras neuronas de su mismo grupo, y conexiones excitadoras con las neuronas de la siguiente capa. En una red de este tipo, después de recibir diferentes informaciones de entrada, cada neurona en cada grupo se especializa en la respuesta a determinadas características de los datos de entrada.

En este tipo de redes, cada neurona tiene asignado un peso total, suma de todos los pesos de las conexiones que tiene a su entrada. El aprendizaje afecta sólo a las neuronas ganadoras (activas), redistribuyendo este peso total entre sus conexiones, sustrayendo una porción a los pesos de todas las conexiones que llegan a la neurona vencedora y repartiendo esta cantidad por igual entre todas las conexiones procedentes de unidades activas. Por tanto la variación del peso de una conexión entre una unidad i y otra j será nula si la neurona j no recibe excitación por parte de la neurona i (no vence en presencia de un estímulo por parte de i, y se modificará (se reforzará) si es excitada por dicha neurona i.

Un ejemplo de este tipo de aprendizaje es el desarrollado por kohonen conocido como Learning Vector Quantization (LVQ), aplicado a redes feedforward de dos capas. El LVQ puede aplicarse de forma diferente, según se precise obtener una o varias unidades vencedoras en la capa de salida. Existe también una extensión supervisada del LVQ basada en un mecanismo por corrección de error.

Una variación del aprendizaje supervisado aplicado a redes multicapa consiste en imponer una inhibición mutua entre neuronas únicamente cuando están a

cierta distancia unas de otras (suponiendo que las neuronas se han dispuesto geométricamente, por ejemplo formando capas bidimensionales).

Existe entonces un área o región de vecindad alrededor de las neuronas que constituye su grupo local.

Fukushima empleó esta idea en 1975 en una red multicapa llamada Cognitron, fuertemente inspirada en la anatomía y fisiología del sistema visual humano, y en 1980 en una versión mejorada de la anterior, denominada Neocognitron. Esta red disponía de un gran número de capas con una arquitectura muy específica de interconexiones entre ellas, y era capaz de aprender a diferenciar caracteres, aunque estos se presentasen a diferente escala, en diferente posición o distorsionados.

El aspecto geométrico de la disposición de las neuronas de una red también es la base de un caso particular de aprendizaje competitivo introducido por kohonen en 1982 conocido como feature mapping aplicado en redes con una disposición bidimensional de las neuronas de salida, que permiten obtener mapas topológicos o topográficos en los que, de algún modo, estarían representadas las características principales de las informaciones presentadas a la red. De esta forma, si la red recibe informaciones con características similares, se generarán mapas parecidos, puesto que serían afectadas neuronas de salida próximas entre sí.

Para concluir este apartado, hay que comentar la existencia de otro caso particular del aprendizaje competitivo, denominada teoría de la resonancia adaptativa desarrollado por Carpenter y Grossberg en 1986 y utilizado en la red f/f de dos capas conocida como ART (en sus dos variantes: ART1, que trabaja con información binaria, y ART2, que maneja información analógica). Esta red realiza un prototipado de las informaciones que recibe a la entrada, generando como salida un ejemplar o prototipo que representa a todas las informaciones que podrían considerarse pertenecientes a la misma clase o categoría.

La teoría de la resonancia adaptativa se basa en la idea de hacer resonar la información de entrada con los prototipos de las categorías que reconoce la red; si entra en resonancia con alguno (es suficientemente similar), la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipo (para que se parezca algo más al dato presentado). Cuando no resuena con ningún prototipo, no se parece a ninguno de los existentes (recordados por la red) hasta ese momento, la red se encarga de crear una nueva categoría con el dato de entrada como prototipo de la misma.

Tipo de aprendizaje no supervisado		Modelo de Red
Aprendizaje Hebbiano	OFF LINE	Hopfield
		Lerning Matrix
		Temporal Associative Memory
		Linear Associative Memory (LAM)
		Optimal LAM
		Drive-Reinforcement
		Fuzzy Associative Memory
	ON LINE	Additive Grossberg
		Shunting Grossberg
		Bidirectional Associative Memory (BAM)
		Adaptive BAM
Aprendizaje competitivo/cooperativo	OFF LINE	Learning Vector Quantizaet
		Cognitron/Neocognitron
		Topology Preserving Map
	ON LINE	Adaptive Resonance Theory

Tabla 2.4. Redes con aprendizaje no supervisado más conocidas

2.2.3. Representación de la información de entrada y salida.

Las redes neuronales pueden también clasificarse en función de la forma en que se representan las informaciones de entrada y las respuestas o datos de salida [Tabla 2.5]. Así, en un gran número de redes, tanto los datos de entrada como de salida son de naturaleza analógica: es decir, son valores reales continuos, normalmente estarán normalizados y su valor absoluto será menor que la unidad. Cuando esto ocurre, las funciones de activación de las neuronas serán también continuas, del tipo lineal o sigmoidal.

Otras redes, por el contrario, sólo admiten valores discretos o binarios a su entrada generando también salidas de este tipo. En este caso, la función de activación será del tipo escalón. Debido a su mayor sencillez, es habitual encontrar algunos modelos de redes reales que aunque inicialmente habían sido desarrollados como discretos por sus autores, posteriormente se ha realizado una versión continua de los mismos, como el caso del modelo de Hopfield (Discrete Hopfield, Continuos Hopfield) y el denominado Adaptive Resonance Theory (ART,1 ART2). Existe también un tipo de redes (que podrían denominarse híbridas) en las que las informaciones de entrada pueden ser valores continuos, aunque las salidas de la red son discretas.

Redes Continuas E: Análogicas S: Análógicas	Redes Híbridas E: Análogicas E: Bianaria	Redes Discretas S: Binarias S: Binaria
Backpropagation	Perceptrón	Discrete Hopfield
Brain State in a Box	Adaline/Madaline	Lerning Matrix
Continuos Hopfield	Linear Associative Reward Penalty	Temporal Associative Memory
LAM	Adpative Heuristic Critic	Didirectional Ass. Memory
Optimal LAM		Cognitron/Neocognitron
Drive Reinforcement		Adaptive Resonance Theory I
Counter Propagation		Boltzman Machine
Additive Grossberg		Cuachy Machine
Shunting Grossberg		
Adaptive BAM		
Leraning Vector Quantizer		
TPM		
ART2		
Continuos Hopfield		

Tabla 2.5. Clasificación de las redes neuronales en función del tipo de información de E/S

2.3. Redes neuronales con conexiones hacia adelante

2.3.1. El Perceptrón

Este fue el primer modelo de red neuronal artificial desarrollado por Rosenblatt en 1958. Un Perceptrón, formado por varias neuronas lineales para recibir las entradas a la red y una neurona de salida, es capaz de decidir cuándo una entrada presentada a la red pertenece a una de las dos clases que es capaz de reconocer.

La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia del tipo escalón. La regla de decisión es responder +1 si el patrón presentado pertenece a la clase A y -1 si es de la clase B. La salida dependerá de la entrada x_i neta (suma de las entradas x_i ponderadas) y del valor umbral Θ.

Una técnica utilizada para analizar el comportamiento de redes como el Perceptrón es representar en un mapa las regiones de decisión creadas en el espacio multidimensional de entradas a la red. Así se visualiza qué patrones pertenecen a una clase y cuáles a otra. El Perceptrón separa las regiones por un hiperplano cuya ecuación queda determinada por los pesos de las conexiones y el valor umbral de la función de activación de la neurona. En este caso, los valores de los pesos pueden fijarse o adaptarse utilizando diferentes algoritmos de entrenamiento de la red.

2.3.1.1. Regla de aprendizaje del Perceptrón

El algoritmo de aprendizaje del Perceptrón es de tipo supervisado, lo cual requiere que sus resultados sean evaluados y se realicen las oportunas modificaciones del sistema si fuera necesario. Como ya se dijo, el Perceptrón no puede aprender a realizar cualquier tipo de clasificación, sólo puede resolver problema de orden 1. Esa limitación se debe a que un Perceptrón usa un separador lineal como célula de decisión, con lo cual no es posible realizar sino una sola separación lineal (por un hiperplano).

Como ejemplo de funcionamiento de una red neuronal de tipo Perceptrón, se verá cómo resolver el problema de la función OR. Para esta función, la red debe ser capaz de devolver, a partir de los cuatro patrones de entrada, a qué clase

pertenece cada uno. Es decir, para el patrón de entrada 00 debe devolver la clase 0 y para las restantes la clase 1. Las entradas serán dos valores binarios. La salida que produce, sin considerar el umbral, es:

$$y = f(\sum_i w_i x_i) = f(w_1 x_1 + w_2 x_2)$$

donde:

x_1, x_2: entradas a la neurona (para las neuronas de la capa de entrada, la salida es igual a la entrada).

w_1, w_2: Pesos de las conexiones entre las neuronas de la capa de entrada y la de la capa de salida.

f: Función de activación.

Si $w_1x_1 + w_2x_2$ es mayor que 0, la salida será 1, y en caso contrario, será -1 (función escalón). Como puede observarse, la sumatoria que se pasa como parámetro a la función f es la expresión matemática de una recta donde w_1 y w_2 son las variables y x_1 y x_2 las constantes. Durante el aprendizaje, se irán variando los pesos obteniéndose distintas rectas.

Lo que se pretende es encontrar una recta que divida al plano en dos espacios de las dos clases de valores de entrada. Para la función OR se deben separar los valores 01,10 y 11 del valor 00.

Si se pretende resolver el problema de la función AND de la misma manera, se llega a la conclusión de que es imposible ya que no existe ninguna recta que pase por el origen de coordenadas y que además separe a los valores 00, 01 y 10 del valor 11. Por esto, es necesario introducir un término independiente para poder realizar esta tarea.

Para ello, como se explicó antes, se considera una entrada de valor fijo 1 a través de una conexión con peso w_0 igual a $-\Theta$ y cuyo valor será ajustado durante el aprendizaje. Ahora el parámetro que se le pasa a la función f de la neurona es $w_1x_1 + w_2x_2 + w_0 1$. Si aplicamos esta solución para el caso de la red que calcula la función OR, aumentamos el número de soluciones, ya que, además de las rectas sin término independiente (w_0=0) existirán otras con término independiente que también resuelvan el problema.

K52/13

K52/13

A continuación se analizará el algoritmo de entrenamiento (por corrección de error) del Perceptrón con N elementos procesales de entrada y uno de salida[Hilera José R., Martínez Victor J., 1994]:

1) Inicialización de los pesos y el umbral

Inicialmente se asignan valores aleatorios a cada uno de los pesos (w_i) de las conexiones y al umbral ($-w_0=\Theta$).

2) Presentación de un nuevo par (Entrada, Salida esperada).

Presentar un nuevo patrón de entrada $X_p = (x_1\ x_2\ \ldots\ x_N)$ junto con la salida esperada $d(t)$.

3) Cálculo de la salida actual

$$y(t) = f(\sum_{i=1}^{N} w_i(t)\, x_i(t) - \Theta)$$

4) Adaptación de los pesos

$$w_i(t+1) = w_i(t) + \alpha[d(t) - y(t)]\, x_i(t)$$

donde d(t) representa la salida deseada y será 1 si el patrón pertenece a la clase A y -1 si es de la clase B. En estas ecuaciones, α es un factor de ganancia en el rango 0.0 a 1.0. Este factor debe ser ajustado de forma tal que satisfaga tanto los requerimientos de aprendizaje rápido como la estabilidad de las estimaciones de los pesos (en el ejemplo de la función OR, se considera α=1). Este proceso se repite hasta que el error que se produce para cada uno de los patrones es cero o bien menor que un valor preestablecido. Los pesos no se cambian si la red ha tomado la decisión correcta.

5) Volver al paso 2).

Este algoritmo es extensible al caso de múltiples neuronas en la capa de salida. El Perceptrón será capaz de aprender a clasificar todas sus entradas, en un número finito de pasos, siempre y cuando los conjuntos de los patrones de entrada sean linealmente separables. En ese caso se puede demostrar que la red aprende en un número finito de pasos.

2.3.1.2. Limitaciones de una red Perceptrón

Un Perceptrón solo puede resolver problemas que sean linealmente separables [Acosta B. María Isabel, Zuluaga M. Camilo A., Salazar Harold, 2000], esto es, problemas cuyas salidas estén clasificadas en dos categorías diferentes y que permitan que su espacio de entrada sea divido en estas dos regiones por medio de un hiperplano definido según la ecuación del Perceptrón, es decir

$$\sum_i w_i(t)x_i(t) - \Theta$$

Para mayor claridad, se analizará un ejemplo de un problema que no es linealmente separable como es el caso de la función XOR [Figura 2.5.].

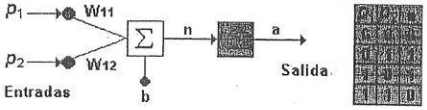


Figura 2.5. El problema de la XOR.

Se pretende que para los valores de entrada 00 y 11 se devuelva la clase 0 y para los patrones 01 y 10 la clase 1. Como puede verse de la figura 2.6 el problema radica en que no existe ninguna línea recta que separe los patrones de una clase de los de la otra.

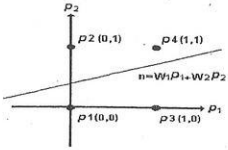


Figura 2.6 Representación de la función XOR.

Los cuatro puntos en la figura 2.6 son las posibles entradas de la red; la línea divide al plano en dos regiones, por lo que se podrían clasificar los puntos de una región como pertenecientes a la clase que posee salida 1 y los de la otra región como pertenecientes a la clase que posee salida 0, sin embargo no hay ninguna forma de posicionar la línea para que los puntos correctos para cada clase se encuentren en la misma región. El problema de la compuerta XOR no es linealmente separable y una red tipo Perceptrón no esta en capacidad de clasificar correctamente los patrones de esta función. Debido a esta limitación del Perceptrón y a su amplia publicación en el libro de Minsky y Papert, el estudio de las redes neuronales se estanco durante casi 20 años.

El proceso para determinar si un problema es linealmente separable o no, se realiza gráficamente sin problema, cuando los patrones de entrada generan un espacio de dos dimensiones, como en el caso de las funciones AND, OR o de la XOR; sin embargo, esta visualización se dificulta cuando el conjunto de patrones de entrada es de tres dimensiones, y resulta imposible de observar gráficamente cuando los patrones de entrada son de dimensiones superiores; en este caso se requiere plantear condiciones de desigualdad que permitan comprobar la separabilidad lineal de los patrones, esto se realiza con base en la ecuación de salida del Perceptrón

$$\sum_{i=1}^{N} w_i(t)x_i(t) - \Theta > 0 \text{ para aquellos patrones que tengan salida 1.}$$

$$\sum_{i=1}^{N} w_i(t)x_i(t) - \Theta < 0 \text{ para aquellos patrones con salida 0.}$$

Para el caso de la XOR, se tiene:

$$0 \cdot w_{11} + 0 \cdot w_{12} + b < 0 \text{ (P1)} \; ; \; 0 \cdot w_{11} + 1 \cdot w_{12} + b > 0 \text{ (P2)}$$

$$1 \cdot w_{11} + 0 \cdot w_{12} + b > 0 \text{ (P3)} \; ; \; 1 \cdot w_{11} + 1 \cdot w_{12} + b < 0 \text{ (P4)}$$

Para que el problema sea linealmente separable, las inecuaciones anteriores no deben contradecirse. Como se ve, las últimas tres ecuaciones exigen que $w_{11} > |b|$, $w_{12} > |b|$ y $w_{11} + w_{12} < |b|$, lo cual es una contradicción. Esta es una forma de comprobar que el problema de la Xor no es linealmente separable. Esta alternativa es buena cuando la dimensión del problema analizado es tan grande que su representación gráfica es imposible de realizar.

La solución al problema de clasificación de patrones de la función XOR se encontraría fácilmente si se descompone el espacio en tres regiones [Figura 2.7.]: una región pertenecería a una de las clases de salida y las otras dos pertenecerían a la segunda clase, así que si en lugar de utilizar únicamente una neurona de salida se utilizaran dos, se obtendrían dos rectas por lo que podrían delimitarse tres zonas; para poder elegir entre una zona u otra de las tres, es necesario utilizar otra capa con una neurona cuyas entradas serán las salidas de las neuronas anteriores; las dos zonas o regiones que contienen los puntos (0,0) y (1,1) se asocian a una salida nula de la red y la zona central se asocia a la salida con valor 1. De esta forma es posible encontrar una solución al problema de la función XOR, por tanto se ha de utilizar una red de tres neuronas, distribuidas en dos capas para solucionar este problema.

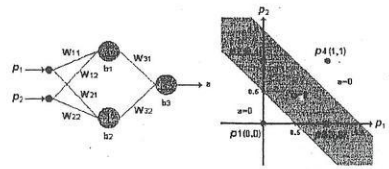


Figura 2.7. Red para resolver el problema de la XOR

Los valores de los pesos y umbrales son:

$$w_{11}=1 \;\; w_{12}=1$$
$$w_{21}=1 \;\; w_{22}=1$$
$$w_{31}=1 \;\; w_{32}=-1.5$$
$$b_1=0.5 \;\; b_2=1.5 \;\; b_3=0.5$$

2.3.1.3. Aplicación de una red Perceptrón para controlar el movimiento de un robot

2.3.1.3.1. Descripción del problema

Un robot es un dispositivo automático que realiza acciones especificas, que dependen de las necesidades del proceso en que se encuentre involucrado, en este caso se tiene un robot que cuenta con cuatro sensores de proximidad en distintas ubicaciones que permanentemente detectan si hay objetos que se encuentren a una distancia superior o inferior a la preestablecida.

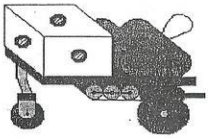


Figura 2.8. Esquema del robot a controlar

Con base en esto se decide si dar marcha adelante o atrás a cada uno de los dos motores que posee [Figura 2.8]. En las lecturas de los sensores podrían darse 16 posibles combinaciones ($16=2^4$) y para cada combinación cada uno de los dos motores podría dar marcha adelante o marcha atrás.

El comportamiento del robot lo describe la tabla 2.6. Cuando los sensores detecten un objeto que se encuentra a una distancia inferior a la predeterminada se dirá que el objeto se encuentra cerca y esto se representa por medio de un 1 y cuando se detecte un objeto que se encuentra a una distancia mayor que la predeterminada se dirá que el objeto esta lejos lo cual se indica con un -1; dependiendo de estas lecturas los motores podrán dar marcha adelante, lo que se representara por un 1 o dar marcha atrás (-1).

S1	S2	S3	S4	M1	M2
1	1	1	1	-1	-1
-1	1	1	1	-1	1
1	1	-1	-1	1	-1
-1	-1	-1	-1	1	1
1	-1	1	1	1	-1
1	1	-1	1	-1	1
1	1	1	-1	1	-1

Tabla 2.6. Patrones de entrenamiento

2.3.1.3.2. Justificación de la red elegida

Este tipo de problema generalmente es resuelto suministrándole al robot una base de datos que contiene todas las posibles situaciones que se podrían presentar y sus respectivas soluciones, En este caso se necesitaría almacenar las respuestas para ambos motores ante las 16 posibles combinaciones en las lecturas de los sensores. Cuando el número de variables de entrada y el número de salidas es mucho mayor, la cantidad de datos necesarios para especificar cada posible situación crece indefinidamente, debido a esto se requerirían dispositivos con gran capacidad de almacenamiento; *en contraste una red neuronal puede entrenarse con un número representativo de patrones y aprender el comportamiento del sistema utilizando dispositivos de menos capacidad de almacenamiento y costo.*

Una red tipo Perceptrón puede ser entrenada con patrones de cualquier dimensión en la entrada y en la salida con datos binarios. Por la simplicidad del

problema este tipo de red es la mas adecuada [Acosta B. María Isabel, Zuluaga M. Camilo A., Salazar Harold, 2000].

Para garantizar que el problema puede ser resuelto por una red neuronal tipo Perceptrón se debe comprobar que los patrones de entrenamiento son linealmente separables, para esto se deben plantear las desigualdades generadas por cada patrón de entrenamiento, en este caso cada patrón de cuatro dimensiones generara dos desigualdades (una por cada salida), estas desigualdades no deben contradecirse, si esto ocurriera el problema no podría ser resuelto por una red tipo Perceptrón de una sola capa y deberá buscarse otro tipo de solución.

Debido a la naturaleza bipolar de la salida, la función de transferencia a utilizar es del tipo escalón.

No se desarrollará aquí, pero el lector puede comprobar que las desigualdades antes mencionadas no se contradicen entre sí.

2.3.1.3.3. Entrenamiento de la red

A la red se le presentaran los 7 patrones de la tabla 2.6, para los cuales dependiendo de las lecturas de los sensores, se le dirá al robot qué hacer específicamente y luego se probará la red con los casos restantes para comprobar la capacidad de generalización de la misma ante un patrón nunca antes visto.

Los estados de lecturas de los sensores y de operación de los motores fueron designados con 1 y -1 puesto que para la convergencia del proceso de entrenamiento resulta más ventajosos propagar valores de 1 y -1 que de 1 y 0.

Debido a la naturaleza de la salida y de la entrada de la red, la función de transferencia

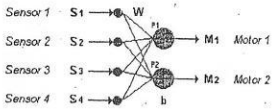


Figura 2.9. Esquema de la red utilzada

adecuada es el escalón.

Se creará una red de 4 entradas con una neurona tipo Perceptrón para cada salida [Figura 2.9.], teniendo así una salida bidimensional, los pesos iniciales aleatorios de la red son:

Pesos	S1	S2	S3	S4
M1	0.8636	-0.1627	0.0503	0.3443
M2	-0.0680	0.6924	-0.5947	0.6762

Umbrales	
M1	1
M2	1

Tabla 2.7. Configuración inicial de pesos y umbrales

Los pesos finales de la red entrenada que satisface todos los patrones de entrada y salida son:

Pesos	S1	S2	S3	S4
M1	0.8636	-4.1627	0.0503	-3.6557
M2	-4.0680	−0.6924	-4.5947	4.6762

Umbrales	
M1	1
M2	1

Tabla 2.8. Configuración final de pesos y umbrales

La red fue simulada para la totalidad de combinaciones posibles de entrada para comprobar que no exista error en el aprendizaje de los patrones de entrenamiento y para observar su capacidad de generalización en los casos restantes

La respuesta de la red a todos los patrones de entrenamiento fue exitosa, como puede observarse en la tabla 2.9

	S1	S2	S3	S4	M1	M2
P1	1	1	1	1	-1	-1
P2	-1	1	1	1	-1	1
P3	1	1	-1	-1	1	-1
P4	-1	-1	-1	-1	1	1
P5	1	-1	1	1	1	-1
P6	1	1	-1	1	-1	1
P7	1	1	1	-1	1	-1

Tabla 2.9. Respuesta de la red ante los patrones de entrenamiento

La red fue simulada para las posibles combinaciones restantes obteniéndose los siguientes resultados:

	S1	S2	S3	S4	M1	M2
C1	-1	-1	1	1	1	1
C2	-1	1	-1	1	-1	1
C3	1	-1	-1	1	1	1
C4	-1	-1	-1	1	1	1
C5	-1	1	1	-1	-1	-1
C6	1	-1	1	-1	1	-1
C7	-1	-1	1	-1	1	-1
C8	-1	1	-1	-1	-1	1
C9	1	-1	-1	-1	1	-1

Tabla 2.10. Respuesta de la red ante patrones no presentados en el entrenamiento

Las combinaciones que no formaban parte del conjunto de entrenamiento, al ser presentadas a la red fueron aproximadas al patrón aprendido con menor distancia euclidiana [Tabla 2.10].

Para las combinaciones C1, C2 y C4 la red decidió dar marcha adelante a ambos motores; para la combinación C5 la red decidió dar marcha atrás a ambos motores, para las combinaciones C6, C7 y C9 la red decidió dar marcha adelante a M1 y marcha atrás a M2 y para las combinaciones C2 y C8 la red decidió dar marcha atrás a M1 y marcha adelante a M2

Una red tipo Perceptrón de una sola capa es una buena solución a un problema que involucre patrones linealmente separables.

La capacidad de generalización de las redes neuronales juega un papel importante cuando las posibles combinaciones de patrones de entrada son tantas que resultaría imposible especificarle a un dispositivo que hacer en cada caso, puesto que la red se entrena con un número de patrones representativo y no con la totalidad de ellos, ahorrando tiempo de computo en la solución del problema.

En las aplicaciones desarrolladas con redes neuronales juega un papel importante la tolerancia a fallas que las caracteriza, pues en caso de fallar uno o varios sensores (como en este caso) la red siempre producirá una salida que en la mayoría de los casos es la mas acertada, debido a que la misma, después de un proceso de aprendizaje exitoso, esta en condiciones de generalizar el comportamiento del sistema.

2.3.2. La red Backpropagation

En 1986, Rumelhart, Hinton y Williams formalizaron un método para que una red neuronal aprendiera la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes utilizando más niveles de neuronas de los que empleó Rosenblatt en su Perceptrón. Este método conocido como **Backpropagation** está basado en la generalización de la regla delta y ha ampliado considerablemente el rango de aplicación de las redes neuronales.

El algoritmo de propagación hacia atrás o retropropagación es una regla de aprendizaje que se puede aplicar en modelos de redes con más de dos capas de neuronas. Una característica importante de este algoritmo es la representación interna del conocimiento que es capaz de organizar en las capas ocultas para lograr cualquier correspondencia entre entradas y salidas.

De forma simplificada, el funcionamiento de una red backpropagation consiste en un aprendizaje de un conjunto predefinido de pares entrada-salida dados como ejemplo, empleando un ciclo propagación-adaptación de dos fases: primero se aplica un patrón de entrada en la capa de entrada de la red el cual se propaga hacia las capas superiores hasta generar una salida, se compara el resultado obtenido en cada neurona de salida con el valor deseado para esa neurona y se obtiene un error para dicha unidad. A continuación, estos errores se transmiten hacia atrás hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite capa por capa hasta llegar a la entrada y hasta que cada neurona haya recibido un error que describa su aporte al error total. Según el valor del error recibido, se reajustan los pesos de las conexiones entre cada par de neuronas en la red, de manera de que el error total cometido para ese patrón disminuya.

2.3.2.1. *La regla delta generalizada*

La regla propuesta por Widrow en 1960 (regla delta) ha sido extendida a redes con capas intermedias (regla delta generalizada) con conexiones hacia delante (feedforward) y cuyas células tienen funciones de activación continuas, no decrecientes y derivables (lineales o sigmoidales), dando lugar al algoritmo de retropropagación (backpropagation).

Este algoritmo utiliza una función o superficie de error asociada a la red, buscando el estado estable de mínimo error a través del camino descendente de la superficie de error. Para ello, realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

2.3.2.1.1. Funcionamiento del algoritmo

Dada una neurona (U_j) y su salida y_j, el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad U_i (w_{ij}) para un patrón de aprendizaje p es:

$$\Delta w_{ij}(t+1) = \alpha \delta_{pj} y_{pi}$$

En donde el subíndice p se refiere al patrón de aprendizaje concreto y α es la constante de o tasa de aprendizaje.

El punto en el que difieren la regla delta generalizada de la regla delta es el valor concreto de δ_{pj}. Por otro lado, en las redes multinivel con capas ocultas no se conoce la salida deseada de las unidades de estas capas para poder calcular los pesos en función del error cometido. Sin embargo, inicialmente sí podemos conocer la salida deseada de las neuronas de salida. Para la unidad U_j de salida, definimos:

$$\delta_{pj} = (d_{pj} - y_{pj}) f'(net_j)$$

donde d_{pj} es la salida deseada para la neurona j y el patrón p y net_j es la entrada neta de la neurona j.

Si U_j no es de salida, se tiene:

$$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj} \right) f'(net_j)$$

donde el rango de k cubre todas las neuronas que están conectadaa a la salida de U_j. El error que se produce en una neurona oculta es la suma de todos los errores cometidos por las neuronas a las que está conectada su salida multiplicados por el peso de la conexión correspondiente.

2.3.2.1.2. Adición de un momento en la regla delta generalizada

En el algoritmo de retropropagación del error se utiliza una amplitud de paso que viene dada por la tasa de aprendizaje α. A mayor tasa, mayor es la modificación de los pesos en cada iteración con lo que el aprendizaje será más rápido pero, por otro lado, puede dar lugar a oscilaciones. Para filtrar estas oscilaciones, se añade en la expresión del incremento de los pesos, un término (momento) β, de manera de que dicha expresión quede:

$$w_{ji}(t+1) = w_{ji}(t) + \alpha \delta_{pj} y_{pi} + \beta(w_{ij}(t) - w_{ji}(t+1)) =$$
$$\Delta w_{ji}(t+1) = \alpha \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t)$$

donde β es una constante que determina el efecto en t+1 del cambio de los pesos en el instante t.

Con este momento se consigue la convergencia de la red en menor número de iteraciones., ya que si en t el incremento de un peso era positivo y en t+1 también, entonces el descenso por la superficie de error en t+1 es mayor. Sin embargo, si en t el incremento era positivo y en t+1 es negativo, el paso que se da en t+1 es más pequeño, lo cual es adecuado ya que esto significa que se ha pasado por un mínimo y que los pesos deben ser menores para poder alcanzarlo.

2.3.2.2. *Estructura y aprendizaje de la red backpropagation*

En una red backpropagation [Figura 2.10] existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (salvo la de entrada) recibe entrada de todas las neuronas de la capa previa y genera salida hacia todas las neuronas de la capa siguiente (salvo las neuronas de salida). No hay conexiones hacia atrás feedback ni laterales o autorrecurrentes.

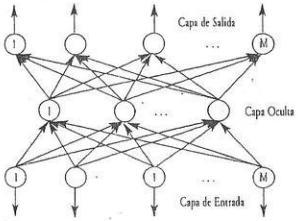


Figura 2.10. Arquitectura de una red Backpropagation

A continuación se presentan, a modo de síntesis, los pasos y fórmulas a utilizar para aplicar el algoritmo de entrenamiento [Hilera José R., Martínez Victor J., 1994].

Paso 1

Inicializar los pesos de la red con valores pequeños aleatorios.

Paso 2

Presentar un patrón de entrada: $X_p=x_{p1},x_{p2},\ldots,x_{pn}$ y especificar las salida deseada que debe generar la red: $d_1,d_2,\ldots,d_M$ (si la red se utiliza como un clasificador, todas las salidas deseadas serán 0, salvo una, que será la de la clase a la que pertenece el patrón de entrada).

Paso 3

Calcular la salida actual de la red, para ello presentamos las entrada a la red y vamos calculando la salida que presenta cada capa hasta llegar a la capa de salida ésta será la salida de la red $y_1,y_2,\ldots,y_M$. Los pasos son los siguientes:

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.
- Para una neurona j oculta:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ij}^{h} + \Theta_j^{h}$$

donde el índice h se refiere a magnitudes de la capa oculta (hidden), el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-ésima neurona oculta.

- Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_j^{h}(net_{pj}^{h})$$

- Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida (capa o: output)

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{jk}^{o}\, y_{pj} + \Theta_k^{o}$$

$$y_{pk} = f_k^{o}(net_{pk}^{o})$$

Paso 4

Calcular los términos de error para todas las neuronas.

Si la neurona k es una neurona de la capa de salida, el valor de delta es:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk})\, f_k^{o\prime}(net_{pk}^{o})$$

La función f debe ser derivable por lo cual no se puede emplear un escalón. Existen dos funciones que pueden servir: la función lineal ($f_k(net_{jk})=net_{jk}$) y la función sigmoidal definida según:

$$f_k(net_{jk}) = \frac{1}{1 + e^{-net_{jk}}}$$

La elección de la función de salida depende de la forma de representar los datos de salida: si se desea que las neuronas de salida sean binarias, se emplea la función sigmoidal, en otro caso es tan aplicable una función como la otra.

Para la función lineal tenemos $f_k^{o\prime} = 1$ mientras que la derivada de una función sigmoidal es:

$$f_k^{o\prime} = f_k^{o}(1 - f_k^{o}) = y_{pk}(1 - y_{pk})$$

por lo que el término de error para las neuronas de salida queda:

$$\delta_{pk}^{o} = d_{pk} - y_{pk}$$

para la salida lineal, y

$$\delta^o_{pk} = (d_{pk} - y_{pk})\, y_{pk}\, (1 - y_{pk})$$

para la salida sigmoidal.

Si la neurona j no es de salida se tiene:

$$\delta^h_{pj} = f'_j\,(net_{pj}{}^h) \sum_k \delta^o_{pk}\, w^o_{jk}$$

Donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término de propagación hacia atrás. Para la función sigmoidal:

$$\delta^h_{pj} = x_{pi}\,(1 - x_{pi}) \sum_k \delta^o_{pk}\, w^o_{jk}$$

Donde k se refiere a todas las neuronas de la capa superior a la de la neurona j. Así, el error que se produce en una neurona oculta es proporcional a la suma de los errores conocidos que se producen en las neuronas a las que está conectada la salida de ésta, multiplicados por el peso de la conexión.

Paso 5

Actualización de los pesos.

Para ello, utilizamos el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la forma siguiente:

Para la capa de salida:

$$w^o_{jk}\,(t+1) = w^o_{jk}\,(t) + \Delta\, w^o_{jk}\,(t+1);$$
$$\Delta w^o_{jk}\,(t+1) = \alpha \delta^o_{pk}\, y_{pj}$$

y para los pesos de las neuronas de la capa oculta:

$$w_{ij}{}^h\,(t+1) = w_{ij}{}^h(t) + \Delta\, w_{ij}{}^h\,(t+1);$$
$$\Delta w_{ij}{}^h\,(t+1) = \alpha \delta_{ij}{}^h\, x_{pi}$$

En ambos casos, para acelerar el proceso de aprendizaje, se puede añadir un término momento de valor $\beta(w^o_{jk}\,(t) - w^o_{jk}\,(t-1))$ en el caso de la neurona de salida y $\beta(w^h_{ij}\,(t) - w^h_{ij}\,(t-1))$ en el caso de una neurona oculta.

Paso 6.

El proceso se repite hasta que el término de error

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^2$$

resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

2.3.2.3. *Consideraciones sobre el algoritmo de aprendizaje*

El algoritmo de backpropagation encuentra un valor mínimo de error (local o global) mediante la aplicación de pasos descendentes (gradiente descendente). Cada punto de la superficie de la función de error corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error.

Por tanto, un problema de este algoritmo de entrenamiento es que busca minimizar la función de error pudiendo caer en un mínimo local o algún punto estacionario sin llegar a encontrar el mínimo global. Sin embargo hay que tener en cuenta que no siempre es necesario encontrar el mínimo global, sino que puede ser suficiente con un error menor a un máximo preestablecido.

2.3.2.3.1. Control de la convergencia

En las técnicas de gradiente descendente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o cerca que se está del mínimo. Con incrementos grandes se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños. Aunque se tarde más, se evita la situación anterior.

La velocidad de convergencia se controla con el parámetro α. Normalmente debe ser un número pequeño (de 0,05 a 0,25) para asegurar la convergencia y evitar oscilaciones.

Una forma de incrementar la velocidad de convergencia consiste en añadir un término momento consistente en sumar una fracción del cambio anterior a la variación de peso actual.

En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas ocultas o en los parámetros de aprendizaje o, simplemente, se vuelve a empezar con un conjunto distinto de pesos originales y se suele resolver el problema.

2.3.2.3.2. Dimensionamiento de la red

No se pueden dar reglas concretas para determinar el número de neuronas o de capas de una red para resolver un problema concreto. Lo mismo ocurre con el número de patrones de entrenamiento.

En general tres capas son suficientes aunque algunas veces es más fácil que la red aprenda con más de una capa oculta. La cantidad de neuronas ocultas que debe tener cada capa es difícil de determinar, no existe regla que lo haga. Si la red se simula por software, lo mejor siempre es tratar de mantener el número de neuronas por capa oculta lo más bajo que se pueda, ya que cada neurona implica un costo de procesamiento extra.

2.3.3. Aplicaciones de las redes backpropagation

A continuación indicaremos algunos de los campos más generales de aplicación de las redes neuronales backpropagation [Hilera José R., Martínez Victor J., 1994].

- Codificación de la información.
  La arquitectura de la red tiene un papel fundamental en este tipo de aplicación. La idea principal consiste en que la información de entrada se recupere en la salida a través de un código interno. Si esto se consigue, en la fase de funcionamiento solo habrá que proporcionar a la red la información codificada, compuesta por los estados de activación de las neuronas ocultas y los pesos correspondientes obtenidos durante la fase de aprendizaje para generar correctamente la salida deseada.

- Traducción de texto en lenguaje hablado.
  Aplicación desarrollada por Sejnpowsky y Rosenberg. En la capa de entrada se codifica cada letra así como el contexto local que consiste en las tres letras anteriores y las tres posteriores a través de una ventana que se va moviendo por el texto secuencialmente. En la capa de salida se codifica la información correspondiente a la pronunciación de una serie de fonemas más ciertas características que dan idea de sus vecinos. Con un número adecuado de unidades ocultas, la red establece la correspondencia texto-fonemas que permitirá, durante la fase de funcionamiento, generar automáticamente los fonemas correspondientes a un determinado texto.
- Reconocimiento óptico de caracteres (OCR).
  Se trata de una aplicación típica de reconocimiento de patrones, siendo en este caso los caracteres de un texto los patrones que hay que reconocer. Se puede utilizar una red backpropagation para aprender las diferentes formas de los caracteres que se representarán como imágenes compuestas por cierto número de píxels, generando la red como salida el código (por ejemplo ASCII) correspondiente al caracter de entrada. El objetivo es que la red tenga capacidad para generalizar; es decir, que reconozca formas de caracteres no utilizadas en la fase de aprendizaje siempre que no difieran demasiado de los patrones mostrados durante el entrenamiento.

2.4. El modelo de Hopfield

2.4.1. Memoria Asociativa

2.4.1.1. La naturaleza de la memoria asociativa

"Recordar" usualmente consiste en asociar algo con una característica notoria, por ejemplo alguien podría decir algo, como el nombre de un artista, e inmediatamente se recuerda una cadena de eventos o experiencia relacionados al artista. El sentido del olfato es conocido por ser especialmente evocativo en esta manera.

El paradigma común puede describirse de la siguiente forma: existen algunas colecciones subyacentes de datos que se mandan e interrelacionan de alguna manera y que se guardan en memoria; los datos pueden ser pensados por lo tanto, como formando un patrón almacenado. En los ejemplos de recolección anteriores, el dato es el cluster de memorias asociadas con la celebridad o recuerdos vividos en la niñez. Cuando se presenta parte del patrón de datos en la forma de una señal sensorial, el resto de los patrones (memoria) es renombrado o asociado con él. Se debe notar que a menudo no importa qué parte del patrón es usado como señal, el patrón entero es siempre restaurado.

2.4.1.2. *Una analogía física de la memoria*

Las redes que se usan para ejecutar asociaciones son ejemplos específicos de una clase de sistemas físicos que se pueden pensar como que realizan las mismas cosas. Estos sistemas permiten que la operación de la red sea vista como un sistema dinámico y físico y que su

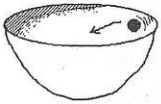


Figura 2.11. El modelo de 'bol'

conducta pueda ser descripta en términos de la energía de la red.

Por ejemplo, considérese un bol [Figura 2.11] en el cual existe una pelota que tiene permitido rodar libremente.

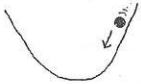


Figura 2.12. Sección del bol

Si se supone que la pelota es soltada desde alguna parte superior del bol, la misma rodará de un lado a otro alrededor de éste para terminar reposando en el fondo [Figura 2.12.].

La descripción física de qué ha pasado puede ser expresada en términos de la energía del sistema. La pelota inicialmente tiene una cierta energía potencial. Cuando se suelta la pelota, la energía potencial es liberada y la pelota rueda alrededor del bol (gana energía cinética). Eventualmente la pelota se detiene, se dice que su energía (potencial y cinética) está en cero. El punto principal es que la pelota termina por detenerse en el mismo lugar cada vez y este lugar está determinado por el mínimo de energía del sistema (pelota + bol).

Existe otra manera de pensar en este proceso. Puede asumirse que la pelota converge siempre al mismo lugar dentro del bol porque recuerda exactamente la posición del fondo. Si la posición o estado de la pelota en cualquier momento está dada por las tres coordenadas espaciales (x, y, z), la locación del fondo del bol, x0, representa el patrón que es almacenado. Puede pensarse en la posición inicial de la pelota como la representación del conocimiento parcial o señal de búsqueda que será ingresada a la red.

Si se utiliza una superficie arrugada [Figura 2.13.] en lugar de una con una única depresión

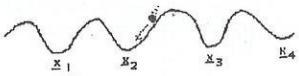


Figura 2.13. Superficie arrugada

(bol), estamos en el caso de una memoria con más de una información almacenada.

Si ahora la pelota comienza a rodar en alguna parte sobre esta superficie, se detendrá eventualmente sobre la depresión local más cercana al punto de inicio. Siguiendo con la última línea de razonamiento, esto es porque la pelota recuerda el patrón almacenado que está más cercano a su patrón parcial inicial o señal de entrada. El funcionamiento de una RNA de Hopfield, así como el de otras redes recurrentes, puede graficarse de esta manera.

2.4.2. Arquitectura de una red de Hopfield

El modelo de Hopfield [Figura 2.14.] consiste en una red monocapa con N neuronas cuyos valores de salida son binarios: 0/1 ó -1/+1. En la versión original del modelo (DH: Discrete Hopfield) [Hoppfield J.J., 1982], las funciones de activación de las neuronas eran del tipo escalón, sin embargo, posteriormente Hopfield desarrolló una versión continua con entradas y salidas analógicas utilizando neuronas con funciones de activación tipo sigmoidal (CH: Continous Hopfield).

Cada neurona de la red se encuentra conectada a todas las demás (conexiones laterales), pero no consigo misma. Además los pesos asociados a las conexiones entre pares de neuronas son simétricos. Esto significa que el peso de la conexión de una neurona i con otra neurona j es de igual valor que el de la conexión de la neurona j con la i ($w_{ij}=w_{ji}$).

$$s_i(t+1) = f\left(\sum_{j=1}^{N} w_{ij}\, s_j(t) - \Theta_i\right)$$

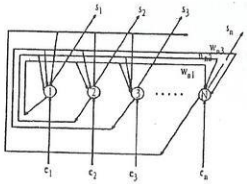


Figura 2.14. Estructura de una red Hopfield

La versión discreta de esta red fue ideada para trabajar con valores binarios -1 y +1 (aunque un ajuste en los pesos hace que se puedan emplear los valores 0 y 1). La función de activación de cada neurona es de tipo escalón:

$$f(x) = \begin{cases} +1 & x > \Theta_i \\ -1 & x < \Theta_i \end{cases}$$

Cuando el valor de x coincide exactamente con Θ_i la salida de la neurona i permanece con su valor anterior.

En el modelo de Hopfield discreto suele adoptarse un valor proporcional a la suma de los pesos de las conexiones de cada neurona con el resto.

$$\Theta_i = k \sum_{j=1}^{N} w_{ij}$$

Si se trabaja con valores binarios -1 y +1, suele considerarse el valor nulo para Θ_i, si los valores binarios son 0 y 1, se toma un valor de 1/2 para k.

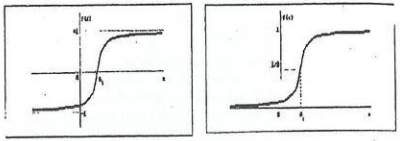
En el caso de las redes de Hopfield continuas, se trabaja con valores reales en los rangos [-1,+1] o [0,1]. En ambos casos, la función de activación de las neuronas es del tipo sigmoidal. Si se trabaja con el primer rango la función utilizada es la tangente hiperbólica [Figura 2.15 (a)]. Si el rango es [0,1], se utiliza el sigmoide de la Figura 2.15.(b)

En ambos casos, a es un parámetro que determina la pendiente de la función sigmoidal.

2.4.3. Funcionamiento de una red Hopfield

Una de las características del modelo de Hopfield es que se trata de una red autoasociativa. Así, varias informaciones (patrones) diferentes pueden ser almacenadas en la red, como si de una memoria se tratase, durante la etapa de aprendizaje. Posteriormente, si se presenta a la entrada alguna de las informaciones almacenadas, la red evoluciona hasta estabilizarse, ofreciendo entonces en la salida la información almacenada, que coincide con la presentada en la entrada. Si, por el contrario, la información de entrada no coincide con ninguna de las almacenadas, por estar distorsionado incompleta, la red evoluciona generando como salida la más parecida.

La información que recibe esta red debe haber sido previamente codificada y representada en forma de vector (como una configuración binaria si la red es discreta, y como un conjunto de valores reales si es continua) con tantas componentes como neuronas (N) tenga la red.



$$f(x-\Theta_i) = \frac{e^{\alpha(x-\Theta_i)} - e^{-\alpha(x-\Theta_i)}}{e^{\alpha(x-\Theta_i)} + e^{-\alpha(x-\Theta_i)}} \qquad f(x-\Theta_i) = \frac{1}{1 + e^{-\alpha(x-\Theta_i)}}$$

Figura 2.15. (a) y (b). Funciones de activación para la red Hopfield continua

Esa información es aplicada directamente a la única capa de que consta la red, siendo recibida por las neuronas de dicha capa (cada neurona recibe una parte de la información, un elemento del vector que representa dicha información). Si se considera en principio el caso de una neurona concreta de la red, esta neurona recibirfa como entrada las salidas de cada un a de las otras neuronas, valores que inicialmente coincidirían con los de entrada, multiplicadas por los pesos de las conexiones correspondientes. La suma de todos estos valores constituirá el valor de entrada neta de la neurona, a la que será aplicada la función de transferencia, obteniéndose el valor de salida correspondiente, 0/1 o -1/1 si la red es discreta, y un número real en el rango [0,1] o [-1,1] si es continua.

El proceso anterior continúa hasta que las salidas de las neuronas se estabilizan. Entonces, el conjunto de estos N valores de salida de todas las neuronas constituye la información de salida que ha generado la red, que se corresponderá con alguna de las informaciones que durante la etapa de aprendizaje fueron almacenadas en la misma.

El funcionamiento puede ser descrito en los siguientes pasos[Hilera José R., Martínez Victor J., 1994]:

1° En el instante inicial (t=0) se aplica la información de entrada (valores e_1, e_2, e_3 .... e_N)

$$s_i(t=0) = e_i \qquad 1 \le i \le N$$

Inicialmente, la salida de las neuronas coincide con la información aplicada a la entrada.

2° La red realiza iteraciones hasta alcanzar la convergencia (hasta que $s_i(t+1) = s_i(t)$).

$$s_i(t+1) = f\left(\sum_{j=1}^{N} w_{ij}\, s_j(t) - \Theta_i\right) \qquad 1 \le i \le N$$

Donde f es la función de transferencia de las neuronas de la red. En el modelo discreto, si se trabaja con valores binarios -1 y +1, la salida se obtendría según la función escalón:

$$s_i(t+1) = \begin{cases} +1 & \sum_{j=1}^{N} w_{ij}\, s_j(t) > \Theta_i \\ s_i(t) & \sum_{j=1}^{N} w_{ij}\, s_j(t) = \Theta_i \\ -1 & \sum_{j=1}^{N} w_{ij}\, s_j(t) < \Theta_i \end{cases}$$

El proceso se repite hasta que las salidas de las neuronas permanecen sin cambios durante algunas iteraciones. En ese instante, la salida (s_1, s_2,...,s_N) representa la información almacenada por la red que más se parece a la información presentada en la entrada (e_1, e_2,...,e_N).

Existe una variación del modelo desarrollada también por Hopfield que pretende parecerse un poco más al funcionamiento de las neuronas reales, se trata de la red Hopfield continua, con funciones de activación de las neuronas del tipo sigmoidal, que ofrece más posibilidades que la anterior, ya que permite almacenar patrones formados por valores reales (por ejemplo, imágenes en color o en blanco y negro con diferentes tonalidades de grises) y además facilita la resolución de determinados problemas generales de optimización que se analizarán más adelante.

Tanto en este caso como en el de la red discreta, se pueden distinguir diferentes versiones del modelo en función de la forma temporal en que se lleva a cabo la actualización de las salidas de las neuronas de la red. Si esta actualización se

realiza en forma simultánea en todas las neuronas de la red, se trata de una red con funcionamiento paralelo o sincrónico. La salida es generada al mismo tiempo por todas las neuronas de la red en cada iteración, de tal forma que en la próxima iteración (t+1) todas van a utilizar como entradas las salidas generadas por las otras en el instante anterior (t). Si, por el contrario, las neuronas trabajan de forma secuencial, actualizándose solo la salida de una neurona en cada iteración, se tratará de una red Hopfield con funcionamiento secuencial o asícrono. En este caso, ocurre que la salida a la que converge la red puede ser diferente en función del orden de la secuencia de activación de las neuronas.

También existe la posibilidad, sólo utilizada en las redes Hopfield continuas, que trabajan con valores no binarios, de que la generación de la salida de todas las neuronas se realice de forma simultánea (síncrona) y continuada en el tiempo. En este caso, el funcionamiento de la red tendría que ser representado en forma de una ecuación diferencial:

$$\tau_i \frac{ds_i}{dt_i} = f(\sum w_{ij} s_j - \Theta_i) - s_i$$

Donde f es la función de activación de las neuronas, que será de tipo sigmoidal y τ_i es un parámetro, denominado tasa de retardo, que pretende representar una característica biológica de las neuronas, como es el retraso que se produce en la recepción por parte de una neurona i de los valores de las salidas generados por las otras neuronas a las que está conectada.

Si se pretende simular una red Hopfield continua como la anterior, puede hacerse resolviendo la ecuación diferencial utilizando métodos numéricos, aunque también, y sobre todo, puede implementarse físicamente de forma directa mediante un circuito analógico que tenga comportamiento gobernado por una ecuación similar. En estos circuitos, los pesos entre neuronas se implementan como resistencias, las funciones de activación mediante amplificadores operacionales, y las tasa de retardo de cada neurona se consigue colocando en paralelo una resistencia y un capacitor a la entrada del amplificador correspondiente.

2.4.4. Aprendizaje

La red Hopfield tiene un mecanismo de aprendizaje OFF-LINE. Por tanto, existe una etapa de aprendizaje y otra de funcionamiento de la red. En al etapa de aprendizaje se fijan los valores de los pesos e función de las informaciones que se pretende que la red memorice. Una vez establecidos los pesos, la red entra en funcionamiento tal y como se describió previamente.

Esta red utiliza un aprendizaje no supervisado de tipo Hebbiano [Hoppfield J.J., 1982], de tal forma que el peso de una conexión entre una neurona i y otra j se obtiene mediante el producto de los componentes i-ésimo y j-ésimo del vector que representa la entrada. Si el número de patrones a aprender es M, el valor definitivo de cada uno de los pesos se obtiene mediante la suma de los M productos obtenidos por el procedimiento anterior, un producto por información a almacenar.

En el caso de la red Hopfield discreta, que trabaja con valores -1/+1, este algoritmo de aprendizaje puede expresarse de la siguiente forma:

$$w_{ij} \begin{cases} \sum_{k=1}^{M} e_i^{(k)} e_j^{(k)} & 1 \le i, j \le N; \ i \ne j \\ 0 & 1 \le i, j < N; \ 1 = j \end{cases}$$

siendo:

w_{ij}: Peso asociado a la conexión entre la neurona j y la neurona i, que coincide con w_{ij}.

$e_i^{(k)}$: el valor de la componente i-ésima del vector correspondiente a la información k-ésima que debe aprender la red.

N: numero de neuronas de la red

M: numero de informaciones que debe aprender la red.

Si la red trabaja con valores discretos 0/1 entonces los pesos se calculan según la expresión:

$$w_{ij} = \begin{cases} \sum_{k=1}^{M} (2e_i^{(k)} - 1)\,(2e_j^{(k)} - 1) & 1 \le i, j \le N;\ i \ne j \\ 0 & 1 \le i, j \le N;\ i = j \end{cases}$$

El algoritmo de aprendizaje también se suele expresar utilizando notación matricial. La siguiente matriz W representa los pesos de todas las conexiones de la red:

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} & , & w_{1N} \\ w_{21} & w_{22} & w_{23} & , & w_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ w_{N1} & w_{N2} & w_{N3} & , & w_{NN} \end{pmatrix}$$

Esta matriz es simétrica ($w_{ij} = w_{ji}$) y tiene una diagonal principal con valores nulos ya que no existen conexiones autorrecurrentes.

También se tendría el conjunto de los M vectores que representan las informaciones que ha de aprender la red:

$$\begin{aligned} E_1 &= (e_1^{(1)} \quad e_2^{(1)} \quad \dots \quad e_N^{(1)}) \\ E_2 &= (e_1^{(2)} \quad e_2^{(2)} \quad \dots \quad e_N^{(2)}) \\ &\dots \\ E_M &= (e_1^{(M)} \quad e_2^{(M)} \quad \dots \quad e_N^{(M)}) \end{aligned}$$

Utilizando esta notación, la matriz de pesos W, se obtiene del siguiente modo:

$$W = \sum_{K=1}^{M} [E_K^T E_K - I]$$

Donde E_K^T es la transpuesta de la matriz E_K e I es la matriz identidad de NxN que anula los pesos de las conexiones autorrecurrentes (w_{ii}).

A modo de ejemplo, considérense las informaciones presentadas en la figura 2.16. [Hilera José R., Martínez Victor J., 1994]:

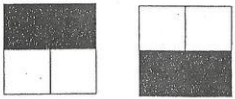


Figura 2.16. Informaciones a memorizar

Los píxeles negros podrían representarse mediante el valor +1 y los blancos con -1. En este caso, las informaciones a almacenar serían dos vectores (M=2) de 4 elementos (N=4). La red tendría 4 neuronas para que cada una reciba el valor de un píxel.

Los vectores de entrada son:

$$E_1 = [1 \quad 1 \quad -1 \quad -1] \qquad E_2 = [-1 \quad -1 \quad 1 \quad 1]$$

El aprendizaje de estas dos informaciones consiste en la obtención de la matriz de pesos W, así:

$$E_1^T E_1 - I = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} [1 \ 1 \ -1 \ -1] - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

$$E_2^T E_1 - I = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} [1 \;\; 1 \;\; -1 \;\; -1] - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

$$W = \sum_{K=1}^{2} E_K^T E_K - I = \begin{bmatrix} 0 & 2 & -2 & -2 \\ 2 & 0 & -2 & -2 \\ -2 & -2 & 0 & 2 \\ -2 & -2 & 2 & 0 \end{bmatrix}$$

Para comprobar el funcionamiento de la red, podríamos ver lo que ocurre al ingresar el patrón $E = [1 \;\; -1 \;\; -1 \;\; -1]$ que representa la figura 2.17.

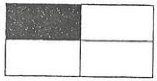


Figura 2.17. Patrón de entrada

Inicialmente, la salida de cada neurona coincide con la entrada. Entonces se produce la primera iteración obteniéndose las siguientes nuevas entradas netas en cada neurona:

$$E \cdot W = [1 \;\; -1 \;\; -1 \;\; -1] \begin{bmatrix} 0 & 2 & -2 & -2 \\ 2 & 0 & -2 & -2 \\ -2 & -2 & 0 & 2 \\ -2 & -2 & 2 & 0 \end{bmatrix} = [2 \;\; 6 \;\; -2 \;\; -2]$$

Si la función activación de cada neurona es sigmoidal sin desplazamiento ($\Theta_i = 0$) la salida de la red luego de la primera iteración será:

$$S = [1 \;\; 1 \;\; -1 \;\; -1]$$

Para la segunda iteración, las entradas netas serán:

$$E \cdot W = [1 \;\; -1 \;\; -1 \;\; -1] \begin{bmatrix} 0 & 2 & -2 & -2 \\ 2 & 0 & -2 & -2 \\ -2 & -2 & 0 & 2 \\ -2 & -2 & 2 & 0 \end{bmatrix} = [2 \;\; 6 \;\; -2 \;\; -2]$$

Entonces, después de esta segunda iteración, se obtiene la siguiente salida:

$$S = [1 \;\; 1 \;\; -1 \;\; -1]$$

Como puede verse, se repite la salida de la primera iteración, se ha llegado a una situación de estabilidad, en la que la red ha generado como salida el patrón almacenado que más se

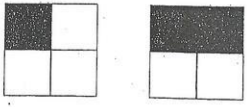


Figura 2.18. Comparación entre entrada y salida generada

parece a la entrada [Figura 2.18]

2.4.4.1. *La función energía*

Como ya se ha comentado, el aprendizaje de una red Hopfield es del tipo Hebbiano. La elección de este tipo de aprendizaje fue, entre otras razones, debido a que asegura la estabilidad de la red; es decir, la convergencia hacia una respuesta estable ante una entrada dada.

Muchas de las investigaciones acerca de la estabilidad de las redes se basan en el establecimiento de una función, denominada función energía de la red, para representar los posibles estados (puntos de equilibrio) de la red.

La función energía de una red Hopfield discreta tiene la siguiente forma[Acosta B. María Isabel, Zuluaga M. Camilo A., Salazar Harold, 2000]:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N} w_{ij}\, s_i\, s_j + \sum_{i=1}^{N}\Theta_i\, s_i$$

Donde:

w_{ij}: Peso de la conexión entre las neuronas i y j.

s_i: Valor de salida de la neurona i.

s_j: Valor de salida de la neurona j.

Θ_i: Umbral de la función de activación de la neurona i.

Esta expresión guarda una profunda similitud formal con la energía en la mecánica clásica. Trata de representar la evolución del sistema, considerando cada configuración (vector) de las salidas de las neuronas de la red como puntos en un espacio de dimensión N y relacionando el estado de la red en cada momento con un punto de ese espacio. La función energía puede imaginarse como una superficie con varios mínimos locales. Cuando en la red se han almacenado M informaciones o patrones, los posibles estados estables de la red serán también M. Estos M estados corresponden precisamente a los mínimos de la función energía. Cuando se presenta a la entrada de la red una información nueva, ésta evoluciona hasta alcanzar un mínimo de la función energía (el más cercano a la entrada presentada).

Cuando la red Hopfield se utiliza como memoria asociativa, el objetivo es conseguir que los patrones a almacenar se sitúen en los mínimos de la función

K52/13

energía, y por lo tanto, sean los estados estables de la red. Puede demostrase que esto se consigue si se verifica que los pesos de las conexiones autorrecurrentes son nulos (w_{ii}=0) y si el resto cumple la regla de Hebb $w_{ij} = \Sigma\, s_i\, s_j$, de ahí que Hopfield eligiera dicho aprendizaje.

Existe una variante para el caso de las redes Hopfield continuas donde la expresión de la función energía es:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N} w_{ij}\, s_i\, s_j + \sum_{i=1}^{N}\int_0^{s_i} f^{-1}(s)\,ds$$

donde f^{-1} es la inversa de la función de activación sigmoidal (f) de una neurona. Si f fuera por ejemplo

$$f(x) = 1/(1 + e^{-\alpha(x-\Theta_i)})$$

entonces la inversa sería $f^{-1}(s) = -(1/2\ \ln(-1 + 1/s) + \Theta_i)$ con lo que la función energía quedaría:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N} w_{ij}\, s_i\, s_j + \sum_{i=1}^{N}\Theta_i\, s_i - \frac{1}{\alpha}\sum_{i=1}^{N}\int_0^{s_i}\ln\left(\frac{1}{s}-1\right)ds$$

Puede verse que coincidiría con la expresión de la función energía de una red Hopfield discreta si no existiera el último término, cuyo valor depende del parámetro α que representa la pendiente o ganancia de la función sigmoidal de cada neurona. Por ello, habrá que tener mucho cuidado con la elección de dicho parámetro ya que influye directamente en los mínimos de la función energía de la red. Si α tiene un valor muy grande (función escalón) el término en cuestión se hace 0 y se tiene una red Hopfield discreta. Pero si α tiene valores pequeños, disminuye el número de mínimos de la función y por lo tanto de posibles estados estables de la red, reduciéndose todo a un solo mínimo cuando α es 0.

2.4.5. Limitaciones del modelo de Hopfield

Existen varios problemas asociados al modelo de red de Hopfield. Aquí se mostrarán los dos más significativos [Hoppfield J.J., 1982]. Uno de ellos tiene que ver con la capacidad máxima de estas redes, y el otro con la necesidad de ortogonalidad entre los patrones de entrenamiento.

Una de las limitaciones del modelo es la cantidad máxima de patrones que se pueden almacenar en una red, pues si se almacenan demasiadas informaciones la red puede converger a valores de salida diferentes de los aprendidos. Puede demostrarse que esta situación no se produce nunca si el numero de informaciones almacenadas es menor o igual que N/4ln(N), siendo N el número de neuronas de la red. Este número puede agrandarse si se permite la posibilidad de un mínimo error en la recuperación de la información. En este caso la cantidad máxima que se puede almacenar es 13.8% del total de neuronas.

Otra de las limitaciones del modelo, es que a pesar de cumplirse el requisito anterior, si dos o más patrones no son suficientemente diferentes (si no son ortogonales), puede ocurrir que cada uno de ellos no represente un mínimo de la función energía, con la probabilidad de que se generen salidas diferentes a todos ellos. También puede ocurrir que ante una entrada que coincida con una de las informaciones aprendidas, la red converja hacia una salida correspondiente a otra de las informaciones almacenadas que fuese muy parecida.

Este problema puede ser minimizado si se garantiza que entre las informaciones a aprender, se cumpla que cada par de patrones de entrada difieran en, al menos N/2 componentes, siendo N el número total de componentes por patrón. Esta condición puede expresarse como:

$$\sum_{i=1}^{N} \epsilon_i^{(k)} \epsilon_i^{(m)} \leq 0 \qquad \forall k \neq m$$

donde $\epsilon_i^{(k)}$ y $\epsilon_i^{(m)}$ son los valores binarios +1 o -1 de los componentes i-ésimos de dos vectores diferentes (m y k) a almacenar en la red. Esta condición puede ser relajada, estableciéndose una distancia mínima del 30% garantizándose todavía un funcionamiento aceptable. Considerando nuevamente valores binarios -1/+1, tenemos:

$$\sum_{i=1}^{N} \epsilon_i^{(k)} \epsilon_i^{(m)} \leq 0.7N - 0.3N = 0.4N$$

pues se permite un 70% de componentes iguales y un 30% de diferentes.

2.4.6. Aplicaciones

A pesar de las limitaciones comentadas en el apartado 2.4.5, el modelo de Hopfield ha sido uno de los más utilizados, como se ha descrito o con algunas modificaciones, siendo esto debido principalmente a la relativa facilidad de implementación física de esta red utilizando tecnología VLSI.

Las aplicaciones más conocidas están relacionadas con el reconocimiento de imágenes y de voz, el control de motores y la resolución de problemas de optimización como el problema del viajante, la bipartición de grafos y muchos otros.

2.4.6.1. Reconocimiento de imágenes

En este tipo de aplicaciones, la red Hopfield puede reconstruir versiones distorsionadas o parciales de imágenes almacenadas durante la fase de aprendizaje. Una limitación del modelo tiene que ver con la imposibilidad de recuperar imágenes rotadas o trasladadas. Para esto habría que utilizar otras reglas de entrenamiento diferentes a la de Hebb que tengan implícitas las invarianzas necesarias como para que se pueda llevar a cabo el reconocimiento deseado.

2.4.6.2. Resolución de problemas de optimización

Una de las aplicaciones de las redes neuronales en general y del modelo de Hopfield en particular, es la resolución de problemas de optimización complejos, aquellos cuyo tiempo de resolución por medio de algoritmos clásicos crece exponencialmente o factorialmente con el tamaño del problema a resolver. Estos problemas reciben el nombre genérico de problemas N-P completos. Ejemplos de estos problemas son el problemas de colocar N reinas en un tablero de ajedrez de NxN sin que entren en jaque, el problema del viajante, el particionamiento de grafos, etc.

La utilización del modelo de Hopfield para la resolución de problemas de optimización, se basa en la idea de intentar fijar el objetivo del problema mediante una expresión matemática, denominada función de coste u objetivo que haya que minimizar. Luego se compara la expresión de dicha función con la de la función de energía de la red, determinándose los valores de los pesos y umbrales en término de parámetros de la función objetivo para que ambas expresiones sean equivalentes. De esta forma, cuando se pone en funcionamiento una red Hopfield continua con los valores calculados, ésta itera hasta alcanzar un mínimo de la función energía que en este caso coincidiría con un mínimo de la función objetivo del problema a resolver. Generalmente, no es fácil encontrar el mínimo global de la función de coste debido a la presencia de mínimos locales de los que habrá que escapar realizando un complicado ajuste de parámetros constantes de la función de coste y de las ganancias de las funciones de activación de las neuronas de la red.

Algunos problemas de optimización que han sido resueltos utilizando redes Hopfield son: El problema del viajante, la bipartición de grafos, el problemas del emparejamiento ponderado, el diseño de circuitos conversores analógico-digitales, etc.

2.5. *El modelo de kohonen*

Existen evidencias que demuestran que en el cerebro hay neuronas que se organizan en muchas zonas, de forma que las informaciones captadas del entorno a través de los órganos sensoriales se representan internamente en forma de mapas bidimensionales. Por ejemplo, en el sistema visual se han detectado mapas del espacio visual en zonas del córtex (capa externa del cerebro). También en el sistema auditivo se detecta una organización según la frecuencia a la que cada neurona alcanza la mayor respuesta.

Aunque en gran medida esta organización neuronal está predeterminada genéticamente, es probable que parte de ella se origine mediante el aprendizaje. Esto sugiere, por tanto, que el cerebro podría poseer la capacidad inherente de formar mapas topológicos de las informaciones recibidas del exterior. De hecho, esta teoría podría explicar su poder de operar con elementos semánticos: algunas áreas del cerebro simplemente podrían crear y ordenar neuronas especializadas o

grupos con características de alto nivel y sus combinaciones. Se trataría, en definitiva, de construir mapas espaciales para atributos y características.

A partir de estas ideas, T. Kohonen presentó en 1982 un sistema con un comportamiento semejante. Se trataba de un modelo de red neuronal con capacidad para formar mapas de características de manera similar a como ocurre en el cerebro. El objetivo de kohonen era demostrar que un estímulo externo por sí solo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de los mapas.

Este modelo tiene dos variantes, denominadas LVQ (Learning Vector Quantization) y TPM (Topolgy Preserving Map) o SOM (Self Organizing Map). Ambas se basan en el principio de formación de mapas topológicos para establecer características comunes entre las informaciones (vectores) de entrada a la red, aunque difieren en las dimensiones de éstos, siendo de una sola dimensión en el caso de LVQ y bidimensional e incluso tridimensional en la red TPM.

2.5.1. Arquitectura de una red de Kohonen

Se trata de una red de dos capas con N neuronas de entrada y M de salida [Figura 2.19].

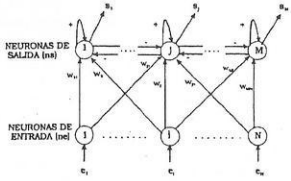


Figura 2.19. Estructura de una red de Kohonen

Cada una de las N neuronas de entrada se conecta a las M de salida a través de conexiones hacia adelante (feedforward).

Entre las neuronas de la capa de salida, puede decirse que existen conexiones laterales de inhibición (peso negativo) implícitas, pues aunque no estén conectadas, cada una de estas neuronas va a tener cierta influencia sobre sus vecinas. El valor que se asigne a los pesos de las conexiones feedforward entre las capas de entrada y salida (w_{ij}) durante el proceso de aprendizaje de la red va a depender precisamente de esta interacción lateral.

La influencia que cada neurona ejerce sobre las demás es función de la distancia entre ellas, siendo muy pequeñas cuando están muy alejadas. Es frecuente que dicha influencia tenga la forma de un sombrero mejicano [Figura 2.20]. Se han descubierto conexiones de este tipo entre las neuronas del sistema nervioso central de los animales.

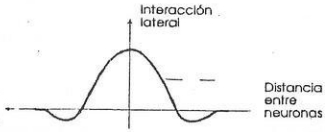


Figura 2.20. Función de interacción lateral entre neuronas

La versión del modelo denominada TPM trata de establecer una correspondencia entre los datos de entrada y un espacio bidimensional de salida, creando mapas topológicos de dos dimensiones, de tal forma que ante datos de entrada con características comunes se deben activar neuronas situadas en zonas próximas de la capa de salida.

La interacción lateral entre neuronas de la capa de salida sigue existiendo, aunque ahora hay que entender la distancia como una zona bidimensional que existe en torno a cada neurona. Esta zona puede ser circular, hexagonal o cualquier otro polígono regular centrado en la neurona en cuestión.

1652/13

2.5.2. Funcionamiento de una red de Kohonen

El funcionamiento de esta red es simple. Cuando se presenta a la entrada una información $E_K = (e_1^{(k)}\ e_2^{(k)}\ \dots\ e_N^{(k)})$, cada una de las M neuronas de la capa de salida la recibe a través de las conexiones feedforward con pesos w_{ij}. También estas neuronas reciben las entradas correspondientes a las interacciones laterales con el resto de las neuronas de salida y cuya influencia dependerá de las distancia a la que se encuentren.

Así, la salida generada por una neurona de salida j ante una vector de entrada E_k será:

$$s_j(t+1) = f\left(\sum_{i=1}^{N} w_{ij} e_i^{(k)} + \sum_{p=1}^{M} Int_{pj}\, s_p(t)\right)$$

donde Int_{pj} es una función del tipo sombrero mejicano que representa la influencia lateral de la neurona p sobre la neurona j. La función de activación de las neuronas de salida (f) será del tipo continuo, lineal o sigmoidal, ya que esta red trabaja con valores reales.

Es evidente que se trata de una red de tipo competitivo, ya que al presentarse una entrada E_k, la red evoluciona hasta alcanzar un estado estable en el que solo hay una neurona activada, la ganadora. La formulación matemática del funcionamiento de esta red puede simplificarse así [Hilera José R., Martínez Victor J., 1994]:

$$s_j = \begin{cases} 1 & MIN\,||E_K - W_j|| = MIN\left(\sqrt{\sum_{i=1}^{N}(e_i^{(k)} - w_{ij})^2}\right) \\ 0 & resto \end{cases}$$

donde $||E_K - W_j||$ es una medida de la diferencia entre el vector de entrada y el vector de pesos de las conexiones que llegan a la neurona j desde la entrada. Es en estos pesos donde se registran los datos almacenados por la red durante el aprendizaje. Durante el funcionamiento, lo que se pretende es encontrar el dato aprendido más parecido al de entrada para averiguar qué neurona se activará y en que zona del espacio bidimensional de salida se encuentra.

La red de kohonen realiza una tarea de clasificación ya que la neurona de salida activada ante una entrada representa la clase a la que pertenece dicha información. Además, como ante otra entrada parecida se activa la misma neurona o una cercana a la anterior se garantiza que las neuronas topológicamente cercanas sean sensibles a entradas físicamente similares. Por esto, la red es muy útil para establecer relaciones antes desconocidas entre conjuntos de datos.

2.5.3. Aprendizaje

El aprendizaje en el modelo de kohonen es de tipo OFF LINE, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento. En la etapa de aprendizaje se fijan los pesos de las conexiones feedforward entra las capas de entrada y salida.

Esta red emplea un aprendizaje no supervisado de tipo competitivo. Las neuronas de la capa de salida compiten por activarse y sólo una de ellas permanece activa ante una entrada determinada. Los pesos de las conexiones se ajustan en función de la neurona que haya resultado vencedora.

Durante la etapa de entrenamiento, se presenta a la red un conjunto de informaciones de entrada (vectores de entrenamiento) para que ésta establezca, en función de la semejanza entre los datos, las diferentes clases (una por neurona de salida) que servirán durante la fase de funcionamiento para realizar clasificaciones de nuevos datos que se presenten a la red. Los valores finales de los pesos de las conexiones feedforward que llegan a cada neurona de salida se corresponderán con los valores de los componentes del vector de aprendizaje que consigue activar la neurona correspondiente. Si existiesen más vectores de entrenamiento que neuronas de salida, más de un vector deberá asociarse a la misma clase. En tal caso, los pesos se obtienen como un promedio de dichos patrones.

Durante el entrenamiento habrá que ingresar varias veces todo el juego de entrenamiento para refinar el mapa topológico de salida consiguiendo que la red pueda realizar una clasificación más selectiva.

El algoritmo de aprendizaje es el siguiente [Hilera José R., Martínez Victor J., 1994]:

1) En primer lugar, se inicializan los pesos (w_{ij}) con valores aleatorios pequeños y se fija la zona inicial de vecindad entre las neuronas de salida.

1152/13

2) A continuación se presenta a la red una información de entrada (la que debe aprender) en forma de vector $E_K = (e_1^{(k)} \quad \ldots \; e_N^{(k)})$, cuyas componentes serán valores continuos.

3) Se determina la neurona vencedora a la salida. Esta será aquella j cuyo vector de pesos W_j sea el más parecido a la información de entrada Ek. Para ello se calculan las distancias entre ambos vectores, una para cada neurona de salida. Suele utilizarse la distancia euclídea o bien la siguiente expresión similar pero sin la raíz:

$$d_j = \sum_{i=1}^{N} (e_i^{(k)} - w_{ij})^2 \qquad 1 \leq j \geq M$$

donde:

$e_i^{(k)}$: Componente i-ésimo del vector k-ésimo de entrada.

w_{ij}: Peso de la conexión entra las neuronas i (de entrada) y j (de salida).

4) Una vez localizada la neurona vencedora j*, se actualizan los pesos de las conexiones feedforward que llegan a dicha neurona y a sus vecinas. Con esto se consigue asociar la información de entrada con cierta zona de la capa de salida.

$$w_{ij}(\tau + 1) = w_{ij}(t) + \alpha(t)\,[e_i^{(k)} - w_{ij^*}(t)] \qquad j \in Zona_{j^*}(t)$$

$Zona_{j^*}(t)$ es la zona de vecindad de la neurona vencedora j+. El tamaño de esta zona se puede reducir en cada iteración del entrenamiento aunque en la práctica es habitual mantener esa zona fija.

El término $\alpha(t)$ es el coeficiente de aprendizaje y toma valores entre 0 y 1. Este parámetro decrece con cada iteración. De esta forma, cuando se ha presentado todo el juego de datos un gran número de veces, a tiende a cero y las variaciones de pesos son insignificantes.

$\alpha(t)$ suele tener alguna de las siguientes expresiones:

$$\alpha(t) = \frac{1}{t} \qquad \alpha(t) - \alpha_1 \left(1 - \frac{t}{\alpha_2}\right)$$

5) El proceso se repite un mínimo de 500 veces ($t \geq 500$).

2.5.4. Aplicaciones

Las principales aplicaciones de una red de Kohonen tienen que ver con el reconocimiento de patrones (voz, texto, imágenes, señales, etc.), codificación de datos, compresión de imágenes y resolución de problemas de optimización, como el del viajante. También se ha utilizado este modelo en robótica, comprobándose su utilidad en el diseño de sistemas para controlar el movimiento de un brazo robótico den tres dimensiones. Lo que hace la red es aprender las magnitudes tensoriales necesarias para moverse en un entorno real.

2.5.4.1. Reconocimiento de voz

Una aplicación de este modelo al reconocimiento de voz lo constituye el conocido Mecanógrafo Fonético [Teuvo Kohonen, 1990]. Se trata de un prototipo ideado por el propio kohonen para convertir, en tiempo real, el habla en texto escrito a partir de un vocabulario limitado con una precisión del 92% y de forma independiente de la persona que hable y del ruido ambiente, con el inconveniente de requerir un proceso de aprendizaje relativamente largo.

La red empleada tiene 15 neuronas de entrada, a través de las cuales reciben las componentes en frecuencia que caracterizan los diferentes fonemas que componen las palabras. Se entrena la red para que ante fonemas parecidos active neuronas de salida próximas, creando finalmente lo que el autor denomina un mapa fonotópico.

Durante el funcionamiento, cuando se pronuncia una palabra, ésta se divide en fonemas que se presentan secuencialmente a la red, la cual responde activando en orden las adecuadas neuronas de la capa de salida, recorriendo un camino en el mapa que se corresponde con la secuencia de entrada, dando lugar a una transcripción fonética de la palabra.

2.5.4.2. Reconocimiento de texto manuscrito

El modelo de kohonen también se ha empleado para construir mapas grafotópicos para convertir texto manuscrito en texto estándar con caracteres codificados en ASCII [Teuvo Kohonen, 1990]. En este caso, la red recibe como entrada la representación paramétrica de un stroke o porción de trazo limitado entre dos puntos sucesivos de escritura manuscrita.

La red de kohonen se entrena con diferentes strokes, generando un mapa grafotópico como el de la Figura 2.21, correspondiente al entrenamiento de una red de 15x15=255 neuronas de salida y 10 de entrada (10 valores por stroke) utilizando un conjunto de patrones de entrenamiento de 200 palabras. El círculo pequeño en el interior de una neurona indica el punto inicial del stroke y la cruz identifica los ejes cartesianos.

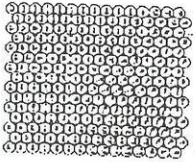


Figura 2.21 Esquema de una red neuronal que funciona como un mapa grafotópico

Después del aprendizaje, durante el funcionamiento, cuando se analiza una palabra manuscrita, esta se divide previamente en strokes que se presentan a la red secuencialmente, la cual responde activando en orden las adecuadas neuronas de la capa de salida, recorriendo un camino en el mapa que se corresponde con la secuencia de entrada, dando lugar a una transcripción grafológica de la palabra.

2.5.4.3. Problemas de optimización

El modelo de kohonen se ha utilizado para resolver el problema de optimización conocido como problema del viajante que busca el camino más corto entre una serie de N ciudades sin pasar dos veces por la misma.

La red que resuelve este problema tiene tantas neuronas de salida como ciudades tiene el recorrido y dos neuronas de entrada para las coordenadas en el plano de cada ciudad. La distribución de las neuronas de salida es circular estableciendo para cada neurona j una zona de vecindad que incluye la neurona anterior (j-1) y la posterior (j+1).

El proceso a seguir para resolver el problema consiste en entrenar a la red para que aprenda la situación geográfica de las N ciudades del recorrido. Cuando el entrenamiento ha concluido, al existir tantas neuronas de salida como ciudades, los vectores de pesos de la red deberán coincidir con las coordenadas de las N ciudades. Entonces el problema ha sido resuelto, siendo la solución el recorrido resultante de unir los puntos (ciudades) correspondientes a los pesos de las neuronas adyacentes de la capa de salida. La solución que se obtiene es buena aunque no puede garantizarse que sea la óptima, ya que el criterio seguido por la red al fijar la zona de vecindad indicada ha sido buscar siempre como siguiente ciudad la más próxima.

Cuando se resuelve el problema del viajante con una red de kohonen puede suceder que más de una neurona sea atraída por la misma ciudad quedando de ese modo ciudades fuera del recorrido. Esta situación puede evitarse utilizando un mayor número de neuronas que de ciudades.

2.5.4.4. *Descripción de una simulación realizada con una red de kohonen*

A continuación se describirá una situación en la cual se emplea una red de Kohonen para crear un mapa o imagen de una región desconocida a la que se llamará G [Anderson J., Rosenfeld E., 1988]. Las señales de entrada a la red provienen de una fuente de sonido moviéndose aleatoriamente alrededor de G y emitiendo señales sonoras de intensidad constante a intervalos de tiempo fijos. La red consta de dos axones de entrada que llevan las señales a las 1600 neuronas que constituyen la capa competitiva. Las neuronas se disponen sobre una matriz plana de 40 x 40 posiciones. Como se ha visto, cada neurona está caracterizada por un vector de pesos bidimensional $w_j=(w_{1j},w_{2j})$ el cual deberá ser ajustado gradualmente de forma tal que la neurona en cuestión se torne receptiva solo para una pequeño conjunto de señales de entrada que se corresponde con una pequeña región dentro de G. Esta zona se conoce como el campo receptivo de la neurona j. Las neuronas deberán coordinar la formación de sus campos receptivos de manera de que la disposición geográfica de las mismas en la matriz se corresponda con la disposición geográfica de los campos receptivos dentro de G.

En la Figura 2.22 puede verse la evolución de la asignación de neuronas a posiciones en G. Para cada neurona j ∈ A, se ha marcado la posición (x,y) de su campo receptivo en G tal y

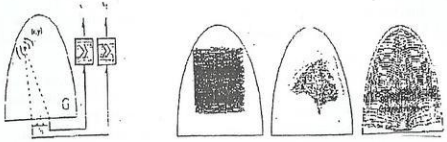


Figura 2.22 Evolución de la simulación

como fue asignado por el mapa. Las marcas se conectan entre sí solo si sus neuronas respectivas son adyacentes en la matriz. Como puede apreciarse, inicialmente la asignación es completamente aleatoria. Solo luego de una pequeña cantidad de iteraciones, la estructura gruesa de G ha sido aprendida hasta que finalmente, luego de 40.000 emisiones de señales sonoras, se logra conseguir una buena asignación de neuronas a posiciones.

En este caso, la red ha encontrado una transformación no lineal que mapea una región curva como es G en una matriz cuadrada como es A.

En la figura 2.23 puede verse cómo la transformación de coordenadas toma en consideración la densidad de probabilidad de las señales de entrada. En este caso se repite el mismo experimento pero ahora la emisión de señales dentro de una región circular en el centro de G tiene una frecuencia tres veces mayor que en el exterior. El resultado es una mayor densidad de neuronas asignadas a esta zona. Es por esto que se dice que *la asignación efectuada por una red de kohonen copia la función de densidad de probabilidad de las señales de entrada.*

En el ejemplo descripto, el espacio de estímulos G es mapeado a una región de igual

Figura 2.23. Resultados de la simulación para el problema con mayor densidad de probabilidad de emisión de señales sonoras en el centro del recinto

dimensión. Si el espacio de estímulos poseyera una dimensión mayor, el mapa trataría de proyectarlo lo más fielmente posible a través de una convolusión apropiada.

Para ilustrar este caso, consideremos el caso de una cadena unidimensional de neuronas como capa competitiva y tomemos como región de estímulos un cuadrado unitario. Una serie de vectores bidimensionales tomados aleatoriamente como puntos de este cuadrado serán presentados a la red en cuestión.

Nuevamente, la red tratará de mapear cada neurona a una posición dentro de la región de estímulos, obteniéndose los resultados de la figura 2.24.

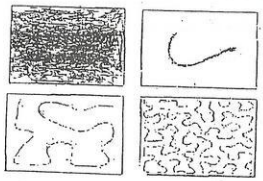


Figura 2.24 Resultado del mapeo a una región de dimensión mayor que la de la capa competitiva

2.5.5. Limitaciones de las redes competitivas

La red de Kohonen pertenece a una clase de redes neuronales conocida como de redes competitivas. En este tipo de redes, las neuronas compiten y cooperan entre sí para aprender a clusterizar las informaciones de entrada (tal y como se vio en el modelo de Kohonen):

Las redes competitivas presentan una serie de problemas o limitaciones que serán analizados a continuación [Acosta B. María Isabel, Zuluaga M. Camilo A., Salazar Harold, 2000].

Las redes competitivas, son bastante eficientes para resolver problemas de clasificación, sin embargo presentan algunos inconvenientes. El primero es la elección de una razón de aprendizaje que permita hallar un punto de equilibrio entre la velocidad de convergencia y la estabilidad final de los vectores de pesos. Una razón de aprendizaje cercana a cero, torna el aprendizaje muy lento pero garantiza que cuando un vector haya alcanzado el centro de la clase objetivo, se mantendrá allí indefinidamente. En contraste, una razón de aprendizaje cercana a uno genera un aprendizaje muy rápido, pero los vectores de peso continuarán oscilando aún después de que se haya alcanzado la convergencia. La indecisión que se presenta al escoger la razón de aprendizaje puede ser empleada como una ventaja si se inicia el entrenamiento con un valor alto y se decrementa en el transcurso del proceso de entrenamiento cuando sea necesario, desafortunadamente esta técnica no funciona si la red necesita continuamente ser adaptada a nuevos argumentos de los vectores de entrada (caso en que la red trabaja en forma Online,, aunque este no es el caso de la red de Kohonen que hemos visto). Un ejemplo de este problema se visualiza en la figura 2.25.

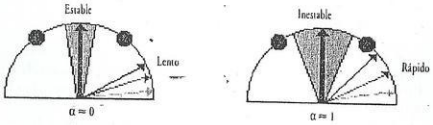


Figura 2.25. Problemas de convergencia en una red competitiva

Un problema de estabilidad más serio ocurre cuando las clases están muy juntas. En ciertos casos, un vector de pesos tratando de apuntar hacia una clase determinada, puede entrar al territorio de otro vector de pesos. En la figura 2.26 pueden observarse con círculos, como dos vectores de entrada son presentados repetidas veces a la red; el resultado,

Figura 2.26. Problemas de convergencia cuando las clases están muy juntas

es que los vectores de pesos que representan las clases de la mitad y de la derecha se encuentran a la derecha. Con seguridad, se presentará el caso en que una de las clases de la derecha será clasificada por el vector de pesos del centro

Un tercer problema con redes competitivas, es que es posible que el vector de pesos inicial de una neurona se encuentre muy lejos de cualquiera de los vectores de entrada y por lo tanto nunca gane la competición. La consecuencia será, la "muerte" de la neurona, lo que por supuesto no es recomendable.

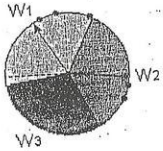


Figura 2.27. Muerte de una neurona

En la figura 2.27 el vector de peso w_3 nunca ganará la competición, sin importar cual sea el orden en que se le presenten los vectores de entrada. Una solución a este problema, consiste en adicionar una ganancia negativa a la entrada neta de cada neurona y decrementar así la ganancia total cada vez que la neurona gane la competición; esto hará que difícilmente una neurona gane varias veces la competición, a este mecanismo se le llama "conciencia".

Una capa competitiva tiene tantas clases como neuronas, lo que podría complicar algunas aplicaciones, especialmente cuando el número de clases no se conoce de antemano. En capas competitivas, cada clase consiste de una región convexa del espacio de entrada, las capas competitivas no pueden formar clases con regiones no convexas o clases que sean la unión de regiones no conectadas.

K52/13

2.6. Ejercicios

1) Explique con sus palabras lo que entiende por Red Neuronal.
2) Describa los elementos que constituyen una neurona artificial, y explique su funcionamiento.
3) Una neurona artificial con función de activación escalón puede comportarse como una compuerta lógica. Explique. Describa tres ejemplos.
4) Clasifique los distintos modelos de redes neuronales según la arquitectura y la regla de aprendizaje. Describa en detalle.
5) Describa arquitectura, regla de aprendizaje y funcionamiento de la red Perceptrón.
6) Explique qué tipos de problemas de clasificación pueden resolverse con un Perceptrón y cuáles no. Ejemplifique.
7) Describa arquitectura, regla de aprendizaje y funcionamiento de la red Backpropagation.
8) Explique de qué manera divide el espacio de entradas una red Backpropagation con una capa oculta ¿y con dos?
9) Mencione y describa al menos dos aplicaciones de la red Backpropagation.
10) Describa arquitectura, regla de aprendizaje y funcionamiento de la red Hopfield.
11) El comportamiento de una red Hopfield está regido por una función energía. Explique.
12) Mencione y describa las limitaciones de una red Hopfield.
13) Una red Hopfield puede utilizarse para resolver problemas de optimización. Explique cómo y cite 2 ejemplos.
14) Describa arquitectura, regla de aprendizaje y funcionamiento de la red Kohonen.
15) Una red de Kohonen realiza una clusterización de las informaciones de entrada. Explique el significado de esta afirmación y dé ejemplos.
16) Mencione y describa al menos dos aplicaciones de la red de Kohonen.