CAPITULO III

3. REDES NEURONALES ARTIFICIALES

3.1 Introducción

La mente humana surge como modelo para máquinas inteligentes ya que en principio es una obvia idea el imitar su comportamiento. Una simulación en computadora del funcionamiento del cerebro ha sido un objeto de estudio de la IA (Inteligencia Artificial) desde los años cuarenta del siglo pasado. Las poderosas cualidades de la mente en lo que respecta a pensamiento, recordación y solución de problemas, a inspirado a los científicos el intentar el modelamiento computarizado de su operación. Un grupo de investigadores se propuso crear un modelo computacional que coincida con la funcionalidad de la mente de una fundamental manera y el resultado ha sido la neurocomputación.

3.2 La neurocomputación

3.2.1 Analogía con el cerebro

La neurona es la unidad celular fundamental del sistema nervioso y, en particular, del cerebro. Cada neurona es una unidad sencilla de microprocesamiento que recibe y combina señales desde muchas otras neuronas a través de estructuras de procesos de entrada llamadas dendritas.

Si la señal combinada es lo suficientemente fuerte, activa el disparo de la neurona, lo que produce una señal de salida; la ruta de la señal de salida se da sobre un componente llamado axón. Esta simple transferencia de información tiene naturaleza química, pero tiene efectos colaterales de tipo eléctrico que podemos medir.

Se estima que el cerebro consiste de algunas decenas de miles de millones de neuronas densamente interconectadas. El axón (ruta de salida de la señal) de una neurona se divide y se conecta hacia las dendritas (ruta de entrada) de otra neuronas mediante una unión llamada sinápsis. La

transmisión sobre esta unión es química en naturaleza y la cantidad de señal trasferida depende de la cantidad de químicos (llamados neurotransmisores) liberados por el axón y recibidos por las dendritas.

Esta eficiencia sináptica (o resistencia) es la que es modificada cuando la mente aprende. La sinápsis combinada con el procesamiento de información en la neurona forman el mecanismo básico de memoria en el cerebro.

La figura 3.1 ilustra una neurona biológica.

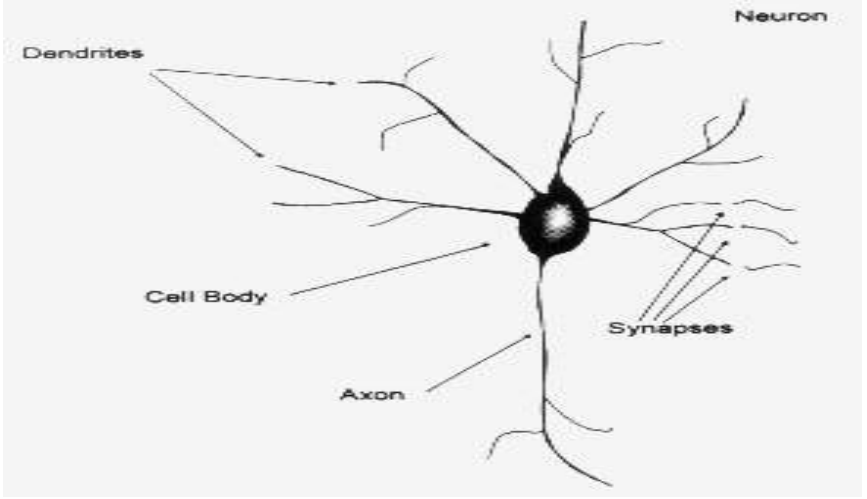

Figura 3.1 Neurona Biólogica

3.2.2 Redes Neuronales

En una red neuronal artificial la unidad análoga a la neurona biológica es referida como un “nodo” o “elemento de procesamiento”. Un nodo tiene muchas entradas (dendritas) y combina, usualmente a través de una suma, los valores de estas entradas. El resultado es un nivel de actividad interna para el nodo. Las entradas combinadas son luego modificadas por una función de transferencia. Esta función de transferencia puede ser de tipo umbral lo que hará, que sólo pase información si el nivel de actividad combinado llega a un cierto nivel, o puede ser una función continua de la combinación de las entradas. El valor de salida de la función de transferencia es generalmente pasado directamente hacia la ruta de salida del nodo.

La ruta de salida de un nodo puede ser conectada a entradas de otros nodos por medio de ponderaciones que corresponden (análogamente) a la resistencia sináptica de las conexiones neuronales. Como cada conexión posee una correspondiente ponderación o peso, las señales de las líneas de entrada hacia

un nodo son modificadas por estos pesos previamente antes de ser sumadas. Es decir, la función de suma es una sumatoria ponderada. En si mismo, este modelo simplificado de una neurona no es muy interesante; los efectos interesantes resultan de las maneras en que las neuronas sean interconectadas.

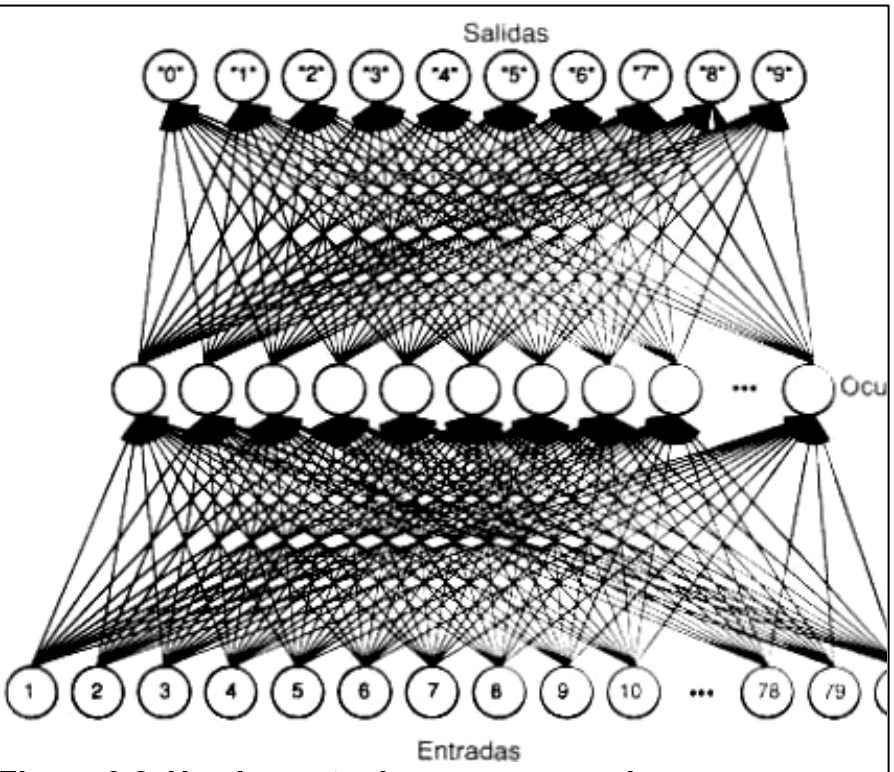

Figura 3.2 Un elemento de proceso o nodo

Una red neuronal consiste de muchos nodos unidos o conectados de la manera en que se muestra en la figura 3.2. Los nodos son usualmente organizados en grupos llamados

"capas". Una red típica consiste de una secuencia de capas con total o aleatorias conexiones entre capas sucesivas.

Existen usualmente dos capas con conexiones hacia el mundo exterior: un búfer de entrada donde los datos se le presentan a la red, y un búfer de salida que contiene la respuesta de la red a una entrada dada. Las capas intermedias se las denomina "capas ocultas".

3.2.3 Operación de las redes

Hay dos fases principales en la operación de una red – Aprender y Recordar. En la mayoría de las redes esto es distinto.

"Aprendizaje" es el proceso de adaptación o modificación de los pesos en respuesta hacia estímulos presentados en la capa de entrada y opcionalmente la capa de salida. Un estímulo presentado en la capa de salida corresponde hacia una respuesta deseada debido a una entrada dada; esta

respuesta deseada debe ser provista por un “instructor con conocimiento”. En tal caso el aprendizaje es referido como “aprendizaje supervisado”.

Si la salida deseada es diferente de la entrada, la red entrenada es conocida como “red hetero-asociativa”. Si, para todo ejemplo de entrenamiento, el vector de salida requerido es igual al vector de entrada, la red entrenada se la llama “auto-asociativa”. Si no se muestra ninguna salida requerida, el aprendizaje se denomina “no supervisado”.

Un tercer tipo de aprendizaje, que cae entre supervisado y no supervisado, es el “aprendizaje de reforzamiento” donde un instructor externo indica si es que la respuesta es buena o mala. En algunas instancias, la red puede ser calificada sólo después de que algunas entradas han sido procesadas por la red.

Cualesquiera tipo de aprendizaje que sea usado, una esencia característica de toda red es su “regla de aprendizaje”. La regla de aprendizaje especifica cómo los pesos se adaptan en respuesta a un ejemplo de entrenamiento. El aprendizaje puede requerir mostrar a la red muchos ejemplos, muchas miles de veces, o solamente una vez.

Los parámetros que gobiernan la regla de aprendizaje pueden cambiar a través del tiempo mientras que la red progrese en su aprendizaje. El control a largo plazo de los parámetros de aprendizaje es conocido como un calendario de aprendizaje.

“Recordar” se refiere a cómo la red procesa un estímulo presentado en su búfer de entrada y crea una respuesta en su búfer de salida. A menudo la recordación es una parte integral del proceso de aprendizaje así como cuando una respuesta deseada de la red deba ser comparada con el valor real de la salida para crear una señal de error.

La forma más sencilla de red neuronal no tiene conexiones de retroalimentación de una capa hacia otra o hacia sí misma. Tal red se denomina "feedforward network" (o algo así como red de pro-alimentación).

Es este caso, la información pasa del búfer de entrada, hacia capas intermedias y hacia la capa de salida, de manera progresiva, usando las características de las funciones de suma ponderada y de transferencia de la particular red en cuestión.

Las Feedforward networks son interesantes debido a no-linealidades en las transformaciones. En algunas de estas redes, cierta cantidad de retroalimentación se crea para que la red tenga sensibilidad al tiempo, es decir, tener puntos de referencia temporales. Estas se llaman redes de recurrencia o de retroalimentación.

Si hay conexiones de retroalimentado, la información reverberará alrededor de la red, sobre la capas o dentro de las mismas, hasta que exista algún criterio de convergencia que sea alcanzado. La información es luego pasada hacia la capa de salida.

3.2.3.1 Operación de las capas

Dos operaciones se usan en la neurocomputación que pueden afectar una red totalmente como un todo, estas son la Normalización y la Competición.

- Normalización: toma el vector de valores, correspondiente a la salida de una capa completa, y la escala para que con esto la salida total (por ejemplo la suma de los componentes del vector) sea un valor fijado. Esto se hace en sistemas biológicos al conectar cada elemento de una capa a cada otro elemento. Estas conexiones permiten al nodo que sienta individualmente la salida total de la capa y ajuste sus propios valores

concordantemente. El resultado de la normalización es que la total actividad en la capa permanece aproximadamente constante.

- Competición: se refiere a la interacción que un nodo puede tener con cada otro nodo en la misma capa. A diferencia de la normalización, donde todos los nodos ajustan sus salidas para crear un nivel fijado de actividad, en la competición sólo una o unas pocas neuronas ganan y producen una salida. Una forma común de competencia es cuando el nodo con mayor actividad es el único en su nivel que se dispara (que significa que da salida a su actual estado).

3.2.4 Qué hace a la neurocomputación diferente?

La neurocomputación difiere de la inteligencia artificial común y la computación tradicional en algunas formas importantes.

3.2.4.1 Aprendizaje a través de ejemplos

A diferencia de los tradicionales sistemas expertos donde el conocimiento se hace explícito a través de reglas, las redes neuronales generan sus propias reglas al aprender de ejemplos mostrados hacia ellas. El aprendizaje se logra a partir de una regla de entrenamiento que adapta o modifica los pesos de conexión de la red en respuesta de entradas de ejemplos y (opcionalmente) las deseadas salidas de aquellas entradas.

En el aprendizaje no-supervisado, se muestran a la red sólo los estímulos de entrada y la red por sí misma se organiza internamente para que cada nodo oculto responda fuertemente hacia un conjunto diferente de estímulos de entrada o a un conjunto de estímulos cercanamente relacionado. Estos conjuntos de estímulos de entrada representan grupos en el espacio de entrada que típicamente representan distintos conceptos del mundo real.

En el aprendizaje supervisado, por cada estímulo de entrada, un estímulo de salida deseado se presenta al sistema y la red se configura a sí misma gradualmente para alcanzar esa correspondencia deseada de entrada / salida. Tal aprendizaje es generalmente una variante de alguno de estos tres tipos:

- Aprendizaje Hebbiano: donde un peso de conexión de una ruta de entrada hacia un nodo es incrementado si la entrada es de alto nivel y si la salida deseada también. En términos biológicos, esto quiere decir que una ruta neuronal se ve reforzada cada vez que se es activada la sinápsis que correlaciona ambas neuronas.

- Aprendizaje de la Regla Delta: se basa en reducir el error entre la salida actual de un nodo y la salida deseada al modificar las ponderaciones. Esta es la regla que usaremos en nuestro caso; cabe notar

que este concepto se asemeja mucho al proceso de mínimos cuadrados en la regresión lineal.

- Aprendizaje Competitivo: en el que los elementos de procesamiento compiten entre ellos, y aquel que obtenga la respuesta más fuerte hacia una entrada dada se modifica a sí mismo para tratar de tornarse en aquella entrada.

3.2.4.2 Memoria de Asociación Distribuida

Una característica importante de las redes neuronales es la forma en que guardan información. La memoria neurocomputacional es distribuida. Los pesos de conexión son las unidades de memoria de una red neuronal. Los valores de las ponderaciones representan el actual estado de conocimiento de la red. Un elemento de conocimiento, representado, por ejemplo, por un par de entrada / salida deseado, se distribuye a través muchas de las unidades de memoria en la red y comparte estas unidades de

memoria con otros elementos de conocimiento guardados en la red.

Ciertas memorias neurocomputacionales son asociativas en que, si la red entrenada se presenta con una entrada parcial, la red escogerá la opción más cercana en memoria para dicha entrada, y generará una salida que corresponderá a una entrada completa. Si la red es auto-asociativa (que es, la entrada es igual a la salida deseada para todos los pares de ejemplos usados para entrenar la red), el presentarle a la red vectores parciales de entrada resultarán en su completación.

La naturaleza de la memoria de una red neuronal conlleva hacia respuestas razonables al presentársele entradas incompletas, ruidosas, o jamás vistas. La propiedad se la denomina “generalización”. La calidad y significancia de una generalización depende de la aplicación particular, y en el tipo y la sofisticación de la

red. Redes multicapa no-lineales (en particular, redes “back-propagation”) aprenden acerca de características en sus capas ocultas y combinan esto para producir las salidas. El conocimiento en las capas ocultas puede combinarse para formar respuestas “inteligentes” a estímulos nuevos.

3.2.4.3 Tolerancia a las fallas

Mientras que los sistemas tradicionales de computación se vuelven inútiles por hasta una pequeña cantidad de daño en la memoria, los sistemas de neurocomputación son tolerantes ante las fallas. La tolerancia a las fallas se refiere a el hecho que en la mayoría de las redes neuronales, si algunos nodos son destruidos, truncados, desactivados, o sus conexiones pequeñamente alteradas, entonces el comportamiento de la red se degrada apenas un poco más.

El desempeño sufre, pero el sistema no es llevado a tan abrupta falla. Los sistemas neurocomputacionales son tolerantes ante las fallas porque la información no está contenida en un solo lugar, pero está distribuida a lo largo del sistema.

Esta característica de degradación “agraciada” hace a los sistemas neurocomputacionales extremadamente convenientes para aquellas aplicaciones en que una falla en equipos de control significa desastre, por ejemplo, en una planta de energía nuclear, guía de mísiles, misiones espaciales, etc.

3.2.4.4 Reconocimiento de patrones

Los sistemas neurocomputacionales son adeptos a muchas tareas de reconocimiento de patrones, aún mucho más que los tradicionales sistemas expertos o estadísticos.

La habilidad humana de traducir los símbolos de esta página hacia palabras con significado e ideas es una forma de reconocimiento de patrones.

Las tareas de reconocimiento de patrones requieren la habilidad de coincidir grandes cantidades de información de entrada simultáneamente, y luego generar salidas generalizadas o categorizadas. También requieren respuestas razonables hacia entradas ruidosas.

Los sistemas neurocomputacionales poseen estas capacidades así mismo como la habilidad de aprender y construir estructuras únicas para problemas específicos o particulares, por lo tanto son especialmente útiles en reconocimiento de patrones.

La habilidad de seleccionar combinaciones de características, pertinentes al problema, les dan una

ventaja por encima de los sistemas basados en la estadística. La habilidad de deducir estas características por sí mismas les da una ventaja por sobre los sistemas expertos utilizados en clasificación de patrones. Es evidente que esta es la principal razón por la que una red neuronal puede servir para clasificar el perfil de un cliente en un análisis crediticio.

3.2.4.5 Poder de Síntesis

Ciertas redes neuronales son capaces de aprender mapeos (correspondencias funcionales) complejos y continuos de una o más entradas hacia una o más salidas. Esta habilidad para sintetizar funciones continuas complejas es análoga a la habilidad de los sistemas biológicos para aprender movimientos coordinados, por ejemplo mover una pierna y pegarle a una pelota.

3.2.5 El Perceptrón de Frank Rosenblatt

En los distintos textos sobre el tema existen varios investigadores que siempre son mencionados por sus trabajos al respecto, tales como McCulloch y Pitts, Minsky, Papert, Block, y el más famoso Rosenblatt.

En 1957, Frank Rosenblatt, en Cornell, publicó el primer mayor proyecto de investigación en neurocomputación: el desarrollo de un elemento llamado "perceptrón". El perceptrón de Rosenblatt provocó una gran cantidad de interés investigativo en la neurocomputación.

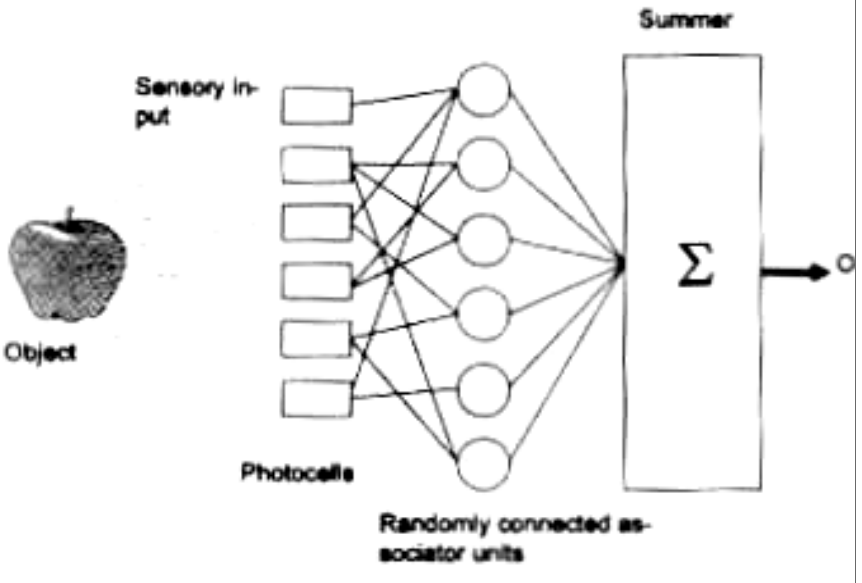

Figura 3.3 Un perceptrón

El perceptrón es un sistema de clasificación de patrones que puede identificar ambos, abstractos y geométricos. El primer perceptrón era capaz de algún aprendizaje y era robusto con respecto a que su operación era únicamente degradada después de dañarse partes de sus componentes.

El perceptrón también poseía una gran cantidad de plasticidad, significando esto que puede ser restringida a un punto en que no haya errores luego que algunas células sean destruidas. Además, el perceptrón era capaz de hacer generalizaciones limitadas y podía categorizar propiamente patrones a pesar de ruido existente en las entradas.

El perceptrón fue primariamente apuntado hacia el reconocimiento óptico de patrones. Una rejilla de 400 fotocélulas, correspondientes a neuronas sensibles a la luz en la retina, recibían el estímulo óptico primario. Estas fotocélulas estaban conectadas por unidades asociadoras que recogían los impulsos eléctricos de las fotocélulas.

Las conexiones entre las unidades asociadoras que recogían los impulsos eléctricos eran hechas al cablear aleatoriamente los asociadores hacia las células. Si la entrada de las células excedía cierto umbral, las unidades de asociación provocaban una respuesta para provocar una salida.

Debido a que era un aparato en desarrollo, el perceptrón también tenía ciertas limitaciones. El perceptrón no tenía estructuras de estados y era inapropiado para descripciones en términos de la teoría de los autómatas. Otra gran limitación, que sería enfatizada por Minsky y Papert, era la inhabilidad del perceptrón para representar el problema básico de la función O-Exclusiva (XOR). Por supuesto esto es el resultado de la naturaleza lineal del perceptrón.

Uno de los cambios más significativos hechos en los años sesentas después de Rosenblatt ha sido el desarrollo de sistemas multicapas, que pueden aprender y categorizar estructuras complejas y no lineales. Esto se hace típicamente

al usar una función de transferencia no lineal, pero puede también surgir de la normalización y competición.

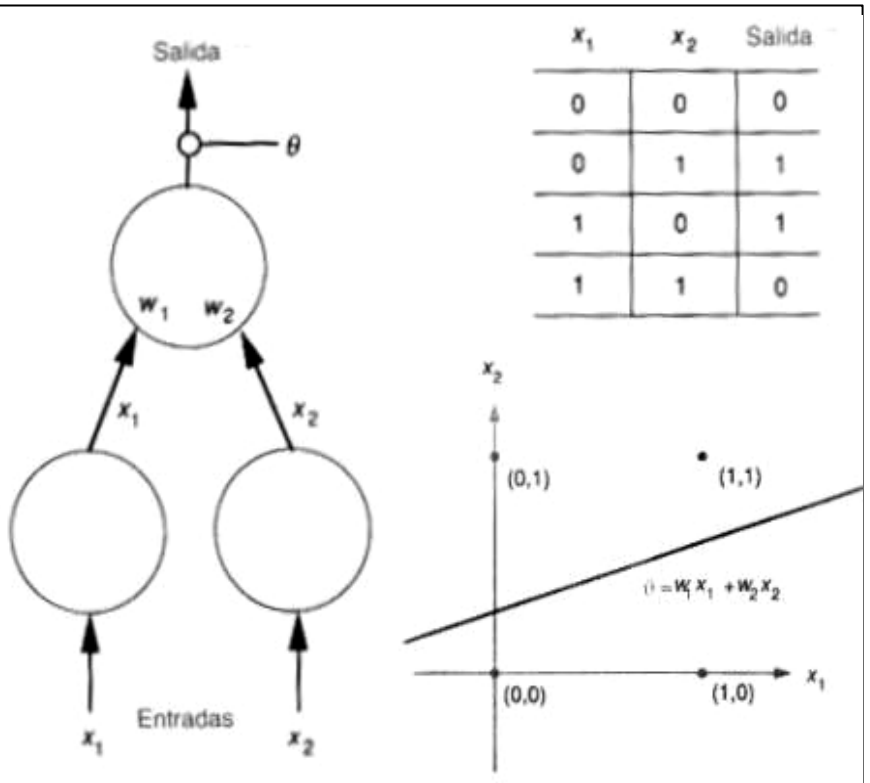

Figura 3.4 Problema XOR

Es fácil entender por que un perceptrón no puede resolver problemas de separabilidad lineal. El problema XOR se ilustra en la figura 3.4. No existe perceptrón alguno que sea capaz de discriminar entre ambos grupos.

Esto se resuelve al utilizar un conjunto de perceptrones (una red) por ejemplo, en la figura 3.5. se da una solución al problema XOR utilizando dos perceptrones. Así, una red neuronal primitiva, como lo es un perceptrón, se puede tornar en una herramienta que permita discriminar patrones aunque estos no tengan características lineales como XOR.

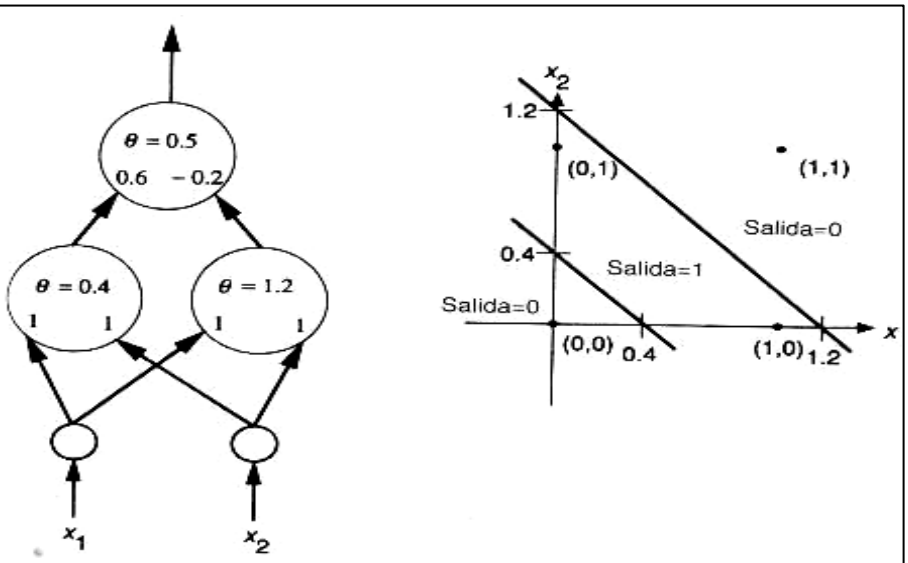

Figura 3.5 Una solución para el problema XOR utilizando dos perceptrones

No existe línea recta capaz de separar ambos grupos, pero si existe una función no-lineal capaz de hacerlo. Una red multicapa puede emular este comportamiento y esa es una cualidad que aprovecharemos para hacer la discriminación para el análisis crediticio de clientes de Bellsouth. También es por esta razón que en un previo capítulo se describió brevemente lo que significaba un análisis discriminante.

3.2.6 Propagación hacia atrás (Back-Propagation)

"Back-propagation" es una técnica para resolver el problema de asignamiento de crédito mencionado por Minsky y Papert en su libro *Perceptrons*. David Rumelhart, es una persona asociada con la invención de las redes de back-propagation. David Parker introdujo un algoritmo similar casi al mismo tiempo y otros han estudiado redes similares.

Una red de perceptrones es capaz de entrenar los nodos de salida para aprender a clasificar patrones de entrada, dado que las clases son "linealmente separables". Las clases más complejas no-linealmente separables pueden ser separadas con una red multicapa. De todas formas, si la salida esta errada, cómo determinamos qué nodo o qué conexión ajustar?

Este es el problema de "asignamiento de crédito" (justamente el que nos conviene en nuestro particular problema). Back-propagation resuelve esto, al asumir que todos los elementos

de procesamiento y conexiones son de alguna forma culpables de una respuesta errónea.

La responsabilidad del error del error se fija al propagar el error de salida hacia atrás a través de las conexiones de la capa previa. Este proceso se repite hasta que se llegue a la capa de entrada.

El nombre “back-propagation” se deriva de este método de distribuir la culpa por el error.

3.2.6.1 La red de Propagación hacia atrás

La típica red de propagación hacia atrás siempre tiene una capa de entradas, una capa de salidas y por lo menos una capa oculta. No hay límite teórico para el número de capas ocultas pero típicamente serán una o dos. Algún trabajo se ha hecho que indica que un máximo de cuatro capas (tres ocultas y una de salida) son requeridas para resolver problemas

arbitrariamente complejos de clasificación de patrones. Cada capa está totalmente conectada con su p

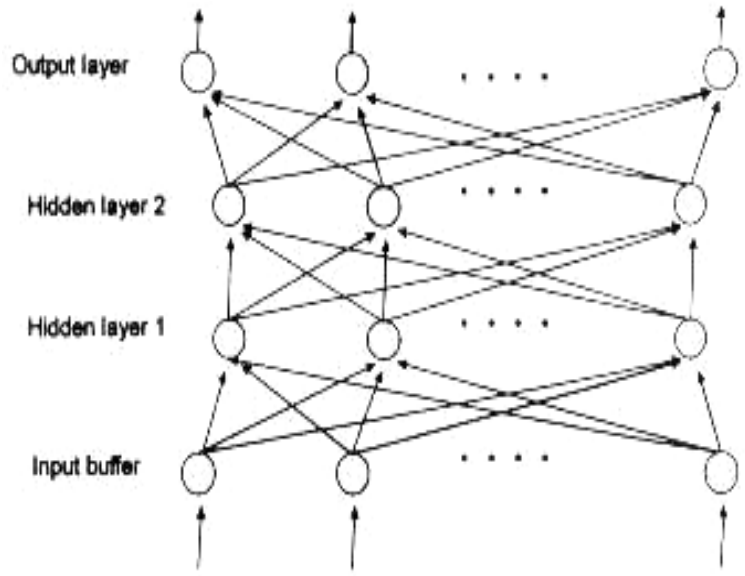

Figura 3.6 Red típica de Back-Propagation

Las flechas indican el flujo de la información durante la recordación. Durante el aprendizaje, la información es también propagada hacia atrás a través de la red y usada para actualizar los pesos de conexión. La red puede ser o hetero-asociativa o auto-asociativa, como se mencionó anteriormente.

Para evita el confundirnos de una capa a otra, una notación clara se hace necesaria para describir la regla de aprendizaje. Usamos un supraíndice en paréntesis rectos para indicar qué capa de la red está siendo considerada. El resto de la notación es como sigue:

- $x_j^{[s]}$ actual estado de salida de la j-ésima neurona en la capa *s*.
- $w_{ji}^{[s]}$ peso en la conexión entre la i-ésima neurona de la capa *(s-1)* y la j-ésima neurona en la capa *s*.
- $I_j^{[s]}$ suma ponderada de las entradas hacia la j-ésima neurona en la capa *s*.

Un elemento de la red de propagación hacia atrás transfiere entonces sus entradas como sigue:

$$x_j^{[s]} = f\left(\sum_i (w_{ji}^{[s]} \cdot x_i^{[s-1]})\right) = f\left(I_j^{[s]}\right) \qquad [1]$$

Donde *f* es tradicionalmente la función sigmoide pero puede ser cualquier función diferenciable. La función sigmoide se define como

$$f(z) = \left(1.0 + e^{-z}\right)^{-1} \qquad [2]$$

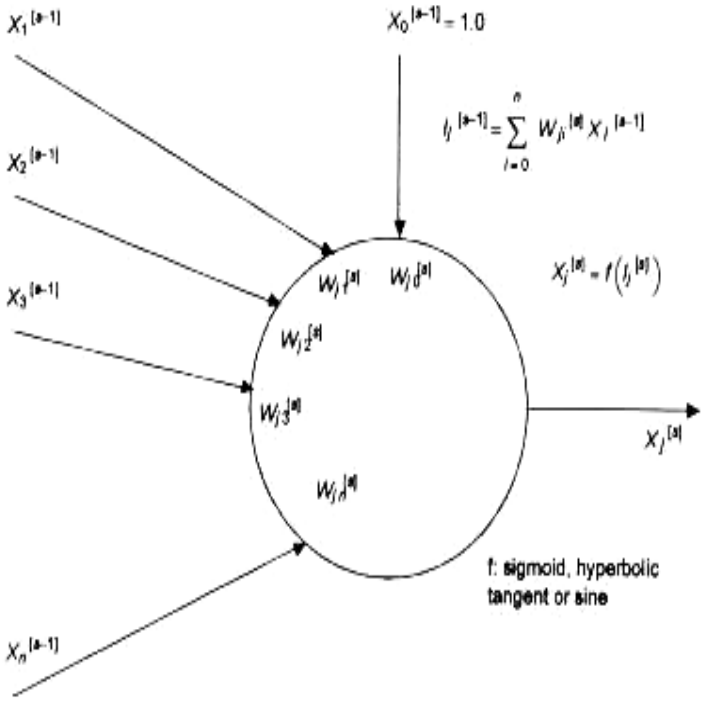

Figura 3.7 Elemento de proceso típico en Back-Propagation

3.2.6.2 Propagando hacia atrás el error local

Supongamos ahora que la red tiene alguna función de error global *E* asociada a ella, que sea una función diferenciable de todas las ponderaciones en la red. La función de error actual no es importante para entender el mecanismo de propagación inversa. El parámetro crítico que se pasa hacia atrás a través de las capas se define como

$$e_j^{[s]} = -\partial E / \partial I_j^{[s]} \qquad [3]$$

Veremos después que esto se puede considerar como una medida del error bcal en el j-ésimo nodo en el nivel *s*.

Utilizando la regla de la cadena dos veces seguidas nos da una relación entre el error local y un nodo en particular en el nivel *s* y los errores locales en los niveles *s+1* :

$$e_j^{[s]} = f'\left(I_j^{[s]}\right) \cdot \sum_k \left(e_k^{[s+1]} \cdot w_{kj}^{[s+1]}\right) \qquad [4]$$

Nótese que en la expresión [4], hay una capa sobre la capa *s*; por tanto, [4] sólo puede ser utilizada para capas internas.

Si la función sigmoide se define en [2], entonces su derivada puede ser expresada como una función simple de sí misma como sigue

$$f'(z) = f(z) \cdot \left(1.0 - f(z)\right) \qquad [5]$$

Por lo tanto, de [1], la equación [4] puede ser reescrita como

$$e_j^{[s]} = x_j^{[s]} \cdot \left(1.0 - x_j^{[s]}\right) \cdot \sum_k \left(e_k^{[s+1]} \cdot w_{kj}^{[s+1]}\right) \qquad [6]$$

Dado que la función de transferencia es una sigmoide. El término de la sumatoria en [6] que es utilizado para retro-propagar errores es análogo al término de suma en [1] que es utilizado para propagar hacia delante las

entradas por la red. Entonces el mayor mecanismo en una red back-propagation es el propagar las entradas por las capas hasta la capa de salida, determinar el error en la salida, y luego propagar los errores hacia atrás desde la capa de salida hasta la capa de entrada usando [6] o más generalmente [4]. La multiplicación del error por la derivada de la función de transferencia escala el error.

3.2.6.3 Minimizando el error local

El objetivo del proceso de aprendizaje es el minimizar el error global *E* del sistema al modificar los pesos. Esta subsección mostrará cómo hacer esto basándonos en el conocimiento del error local en cada nodo.

Dado el actual conjunto de ponderaciones $w_{ji}^{[s]}$, necesitamos determinar cómo incrementar o decrementarlas de manera que decrezca el error

global. Esto puede hacerse usando una regla de descenso por el gradiente como sigue:

$$\nabla w_{ji}^{[s]} = -lcoef \cdot \left(\partial E / \partial w_{ji}^{[s]} \right) \qquad [7]$$

Donde *lcoef* es un coeficiente de aprendizaje. En otras palabras, cambiar cada peso de acuerdo al tamaño y dirección de el gradiente negativo en la superficie de error.

Las derivadas parciales en [7] pueden calcularse directamente del valor del error local discutido en la anterior subsección, porque, por la regla de la cadena y [1]:

$$\begin{aligned} \partial E / \partial w_{ji}^{[s]} &= \left(\partial E / \partial I_j^{[s]} \right) \cdot \left(\partial I_j^{[s]} / \partial w_{ji}^{[s]} \right) \\ &= -e_j^{[s]} \cdot x_i^{[s-1]} \end{aligned} \qquad [8]$$

Combinando [7] y [8] tenemos

$$\nabla w_{ji}^{[s]} = lcoef \cdot e_j^{[s]} \cdot x_i^{[s-1]} \qquad [9]$$

3.2.6.4 La función de error global

La discusión previa a asumido la existencia de una función de error global sin especificarla. Esta función es necesaria para definir los errores locales en la capa de salida para que estos puedan ser retro-propagados hacia el interior de la red.

Supongamos que un vector es presentado en la capa de entrada de la red y supongamos que la salida deseada *d* es especificada por un instructor. Sea *o* quien denote la salida producida por la red con su actual conjunto de pesos. Entonces una medida del error al lograr esa salida deseada está dada por

$$E = 0.5 \cdot \sum_k \left((d_k - o_k)^2 \right) \qquad [10]$$

Donde el subíndice *k* indexa los componentes de *d* y *o*. Aquí, el error local primario está dado por $d_k - o_k$. De la expresión [3], el "error local escalado" de cada nodo de la capa de salida está dado por

$$\begin{aligned} e_k^{(o)} &= -\partial E / \partial I_k^{(o)} \\ &= -\partial E / \partial o_k \cdot \partial o_k / \partial I_k \\ &= (d_k - o_k) \cdot f'(I_k) \end{aligned} \qquad [11]$$

E, como se definió en [10], define el error global de la red para un vector particular *(i, d)*. Una función en conjunto del error global puede definirse como la suma de todas las funciones de patrones específicos del error.

Cada vez que un particular vector *(i, d)* es mostrado, el algoritmo de back-propagation modifica los pesos para reducir ese componente en particular del la función de conjunto global de error.

3.2.6.5 Resumen del algoritmo estándar de propagación inversa

Dado un vector de entrada *i* y una salida deseada *d*, hacer lo siguiente:

1. Presente *i* a la capa de entrada de la red y propáguelo hacia la capa de salida para obtener un vector de salida *o*.

2. Mientras esta información está siendo propagada por la red, también preparará todas las entradas sumadas l_j y estados de salida x_j para cada nodo en la red.

3. Para cada nodo en la capa de salida, calcular el error local escalado como en [11] y luego calcular el peso delta usando [9].

4. Para cada capa *s*, comenzando por la capa previa a la de salida y terminando por la posterior a la de entrada, y para cada nodo en la capa *s*, calcular el

error local escalado como en [6] y luego calcule los pesos delta usando [9].

5. Actualice todos los pesos en la red al sumar el término delta a sus correspondientes pesos previos. A esta regla se le denomina Regla Delta Generalizada que es la ley de aprendizaje de una red de propagación hacia atrás.

3.2.6.6 Variaciones al algoritmo estándar

Término de momento

Uno de los problemas de un algoritmo de descenso por el gradiente es el asignar una tasa de aprendizaje apropiada. El cambiar los pesos como una función lineal de las derivadas parciales como está definido en [7] hace la suposición que la superficie de error es localmente lineal, donde “localmente” se define por el tamaño del coeficiente de aprendizaje.

En puntos de alta curvatura, esta asunción de linealidad no cuadra y un comportamiento divergente puede ocurrir cerca de dichos puntos. Es entonces importante el mantener el coeficiente de aprendizaje pequeño y eludir tal comportamiento.

Por otra parte un coeficiente de aprendizaje pequeño puede conllevar hacia una aprendizaje lento. El concepto de "término de momento" fue introducido para resolver esta dicotomía. La ecuación de peso delta [9] se modifica para que la una porción del anterior peso delta sea alimentado a través del actual peso delta:

$$\nabla w_{ji}^{[s]}(t) = lcoef \cdot e_j^{[s]} \cdot x_i^{[s-1]} + momentum \nabla w_{ji}^{[s]}(t-1) \quad [12]$$

Esto actúa como un filtro de baja permisión (low-pass filter) en los términos de pesos delta dado que las tendencias generales se refuerzan mientras que el comportamiento oscilatoria se auto cancela. Esto

permite entonces un pequeño valor para el coeficiente de aprendizaje y un aprendizaje más rápido.

Actualización acumulada de los pesos

Otra técnica que tiene un efecto en la rapidez de la convergencia es el solamente actualizar los pesos después de que los pares de (entrada, salida deseada) sean presentados a la red, en lugar que después de cada presentación. Esto es referido como propagación hacia atrás acumulada (cumulative back-propagation) porque los pesos delta son acumulados hasta que el conjunto completo de pares es presentado.

El número de pares de entrada / salida presentados durante la acumulación es denominada “época” (epoch). Esto puede corresponder sea al conjunto completo o a un subconjunto de pares de entrenamiento.

Diferentes funciones de transferencia

En lugar de la función sigmoide, cualquier función "suave" puede ser utilizada como función de transferencia para un elemento de procesamiento. Comúnmente se cambia esta función por la de tangente hiperbólico, seno o lineal. Recuérdese que en un principio un perceptrón usaba una función umbral generalmente de tipo (0, 1). La tangente hiperbólica es solamente una versión bipolar de la sigmoide: la sigmoide (o logística) es una versión suave de la función umbral (0, 1) tanto que la tangente hiperbólica es una versión suave de una función umbral (-1, 1). Véase la figura 3.8.

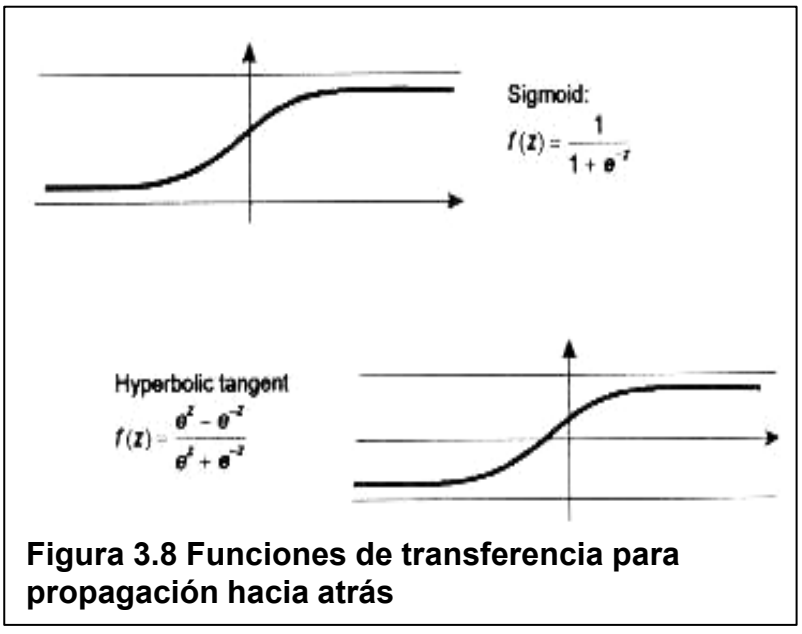

Figura 3.8 Funciones de transferencia para propagación hacia atrás

La tangente hiperbólica se define por

$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \qquad [13]$$

Como en la función sigmoide, la derivada de la tangente hiperbólica puede también ser expresada en términos de si misma:

$$f'(z) = (1.0 + f(z)) \cdot (1.0 - f(z)) \qquad [14]$$

Por tanto con este tipo de función de transferencia la función propagadora del error [4] se modifica en

$$e_j^{[s]} = \left(1.0 + x_j^{[s]}\right) \cdot \left(1.0 - x_j^{[s]}\right) \cdot \sum_k \left(e_k^{[s+1]} \cdot w_{kj}^{[s+1]}\right) \qquad [15]$$

Esto como para ejemplificar la diversidad de los resultados al cambiar las funciones de transferencia. Gran parte de los paquetes neurocomputacionales actuales sugieren las funciones de transferencia por "default". Sólo la experiencia al experimentar y la familiaridad con el problema harán modificar las funciones de transferencia, según ciertos expertos.

3.3 Observaciones

Todo lo antes expuesto tiene un solo fin: sustentar teóricamente la herramienta con que solucionaremos el problema de análisis crediticio. Una buena parte de la teoría existente busca solucionar problemas matemáticos de aproximaciones funcionales no lineales tales como el de la "O-Exclusiva", un claro caso de análisis discriminante.

Las redes de "propagación hacia atrás" son necesariamente la opción más adecuada para ser utilizada en la determinación de si un cliente es o no es un buen sujeto de crédito. Esto ocurre primariamente por la capacidad de la red de imitar una correspondencia funcional entre un conjunto de entrada y otro de salida.

Tratemos de hacer una abstracción del problema: a la operadora celular Bellsouth acuden cierto número de clientes con diferentes características crediticias que podemos agrupar como variables en un vector *i* de entrada. Así mismo, luego de aceptar al cliente, este sujeto de crédito genera un historial en una base de datos de cobranzas.

Evaluando al cliente en función de su historial mas no de sus características crediticias podremos decir que tan “buen pagador” es el cliente a través de sus pagos registrados en la base de datos. Sean estos resultados agrupados en un vector *d* de salida.

Entonces tendríamos ya material suficiente para entrenar una red de back-propagation a reconocer los patrones de comportamiento de los clientes utilizando sus variables crediticias. Esta es la red que, una vez entrenada, podrá repetir dicho comportamiento y evaluará a un cliente con los datos de la hoja de riesgo de la que se hace mención en el capítulo 1.