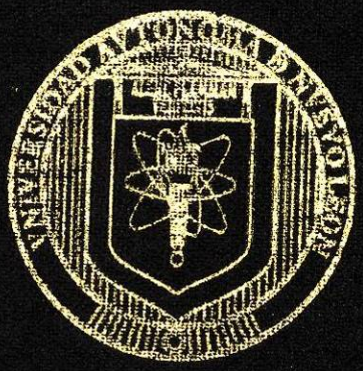
UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE CIENCIAS FISICO MATEMATICAS



PERCEPTRONES MULTICAPA UTILIZANDO LA RETROPROPAGACION COMO ALGORITMO DE ENTRENAMIENTO

POR
MAGDA GUADALUPE MIJANGOS VAZQUEZ

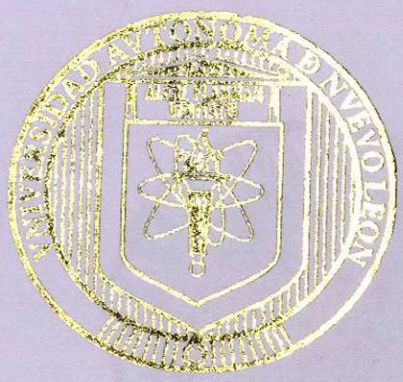
COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE LICENCIADO EN CIENCIAS COMPUTACIONALES

MARZO 2006

TL
QA76
.87
.M55
2006
c.1


1080092627

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE CIENCIAS FISICO MATEMATICAS



PERCEPTRONES MULTICAPA UTILIZANDO LA RETROPROPAGACION COMO ALGORITMO DE ENTRENAMIENTO

POR

MAGDA GUADALUPE MIJANGOS VAZQUEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE LICENCIADO EN CIENCIAS COMPUTACIONALES

MARZO 2006

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO MATEMATICAS

PERCEPTRONES MULTICAPA UTILIZANDO LA RETROPROPAGACIÓN COMO ALGORITMO DE ENTRENAMIENTO

Por

MAGDA GUADALUPE MIJANGOS VAZQUEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

Marzo del 2006

Prologo

La información presentada es un tema de las redes neuronales en particular se trata el tema de los perceptrones multicapas . En esta tesis se pretende mostrar un panorama mas general acerca de los perceptrones Multicapas y los avances que este ha tenido ha través del tiempo .El contenido fue seleccionado del tal forma que fuese de fácil entendimiento para el lector, se presenta las definiciones básicas de cada una de las parte que componen una red neuronal y los diferentes algoritmos de entrenamiento, las consideraciones que se tienen que hacer al momento de diseñar una red neuronal. Así como las aplicaciones en que se pueden utilizar .

Por ultimo se muestra la información acerca del software utilizado para presentar un ejemplo de la aplicación de este tipo de Perceptrones, en este caso se utilizo el software Tiberius el cual tiene como algoritmo de entrenamiento la retropropagación. La sociedad Financiera de objetivos limitados hizo el favor de proporcionar la información necesaria para poder realiza la práctica. Ellos necesitaban saber si una persona podría caer en cartera vencida o no. Con la información proporcionada se analizaron los datos que podían proporcionar un mejor resultado, una vez seleccionados los datos se procede a darles entrenamiento, después del entrenamiento utilizando el software Tiberius se obtuvieron resultados favorables ya que se muestra la salida correspondiente para cada dato y en base a esa información se podría tomar una decisión si esa persona podría hacer en un estado de deudos o no. Cabe resaltar que como es un software de prueba tiene restricciones de solo utilizar 250 datos para realización del entrenamiento.

Dedicada a:

A mis padres por todo el apoyo y motivación que me dieron durante mi formación académica

A mi esposo por la ayuda y compresión durante este difícil camino y por su paciencia todo este tiempo, gracias.

A mi hija por su cariño y ternura.

A mis hermanos por que siempre me motivaron a seguir adelante.

Agradecimientos

Agradezco al M.T. Manuel Serrano por su ayuda y por todo el tiempo que dedicó en el desarrollo de la tesis, por sus valiosas recomendaciones y los consejos dados para mejorar la calidad de la tesis.

Agradezco a la Lic. Rosario Sánchez ya que en todo el trayecto de estudio de la licenciatura y después de terminar siempre estuvo apoyándome y motivándome, recibí mucho apoyo de su parte.

Agradezco al Lic. Rogelio Sepúlveda y al Ing. Miguel Ángel Cárdenas por sus valiosas recomendaciones para mejorar la presentación de la tesis.

A las secretarias que siempre me apoyaron y me ayudaron en mi estancia en la facultad.

A todos mis maestros por su apoyo a lo largo de la licenciatura.

Y a todos mis amigos por motivarme.

TABLA DE CONTENIDO

Lista de Figuras página

Notación

K o N	Neurona
w_{kj}	Representación del peso sináptico
xj	Señal de entrada
Fk	Función de activación
Bk	Bias
Yk	Señal de salida
$w_k = (b_k, w_{1k}, \ldots, w_{jk}, \ldots, w_{Nk})^T$	Vector de pesos
$x = (1, x_1, \ldots, x_N)^T$	Vector de entrada
u_k	Umbral
NET	Suma de productos para cada neurona
δ	Salida deseada – salida obtenida
α	Constante o tasa de aprendizaje.
β	Constante que determina el efecto $t+1$ del cambio de los pesos en el instante t.
net_j	La entrada neta que recibe la neurona j.

Capitulo I

INTRODUCCION

Desde hace mucho tiempo el hombre se ha preguntado cómo funciona el cerebro, esa caja maravillosa que encierra tantos secretos y que ha provocado la curiosidad de científicos e investigadores, ellos han creado teorías y métodos matemáticos para tratar de comprender e imitar el funcionamiento del cerebro. Esto ha contribuido a la Inteligencia Artificial, ya que lo que se pretende es crear entes que sean capaces de actuar por sí mismos y de aprender de sus experiencias, para que estos entes artificiales puedan aprender es necesario un entrenamiento por medio de las redes neuronales artificiales. El estudio de las redes neuronales artificiales surge como resultado de la interrogante del funcionamiento del cerebro.

Para saber un poco más de la historia de las redes neuronales artificiales se mostrará a continuación como se dieron estos avances y las personas que contribuyeron a ello.

Orígenes de las redes neuronales Artificiales

Durante la década de 1920 a 1930 se intentó utilizar la teoría de la conmutación telefónica como punto de partida de un sistema de conocimiento similar al del cerebro. Entre 1940 y 1950 los científicos comenzaron a pensar seriamente en las redes neuronales, utilizando como concepto la noción de que las neuronas del cerebro funcionan como interruptores digitales (on - off) de manera similar al también recién desarrollado computador digital. Así nace la idea de "revolución cibernética" que maneja la analogía entre el cerebro y el computador digital.

Siguiendo los avances de las redes neuronales, éstos se dieron de la siguiente manera.

1936 - Alan Turing. Fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, quienes en 1943, lanzaron una teoría acerca de la forma de trabajar de las neuronas (Un Cálculo Lógico de la Inminente Idea de la Actividad Nerviosa - Boletín de Matemática Biofísica 5: 115-133). Ellos modelaron una red neuronal simple mediante circuitos eléctricos.

1943 – Surge la Teoría de las Redes Neuronales Artificiales. Walter Pitts junto a Bertran Russell y Warren McCulloch intentaron explicar el funcionamiento del cerebro humano, por medio de una red de células conectadas entre sí. Partiendo del menor suceso psíquico (estimado por ellos): el impulso todo/nada, generado por una célula nerviosa.

El ciclo "sentidos - cerebro - músculos", mediante la retroalimentación, produciría una reacción positiva si los músculos reducen la diferencia entre una condición percibida por los sentidos y un estado físico impuesto por el cerebro.

También definieron la memoria como un conjunto de ondas que se reflejan en un circuito cerrado de neuronas.

1949 - Conductividad de la sinápsis en las Redes Neuronales. Seis años después de que McCulloch y Pitts mostraran sus Redes Neuronales, el fisiólogo Donald O. Hebb (de la McGill University) expuso que éstas (las redes neuronales) podían aprender. Su propuesta tenía que ver con la conductividad de la sinápsis, es decir, con las conexiones entre neuronas. Hebb expuso que la repetida activación de una neurona por otra a través de una sinápsis determinada, aumenta su conductividad, y la hacía más propensa a ser

activada sucesivamente, induciendo a la formación de un circuito de neuronas estrechamente conectadas entre sí.

1951 - Primera Red Neuronal. Minsky se inspiró en Skinner para gestar su primera idea "oficial" sobre inteligencia artificial, su Red Neuronal. En aquel entonces entabló amistad con otro brillante estudiante, Dean Edmonds, el cual estaba interesado en el estudio de una nueva ciencia llamada Electrónica.

Durante el verano de 1951, Minsky y Edmonds montaron la primera máquina de redes neuronales, compuesta básicamente de 300 tubos de vacío y un piloto automático de un bombardero B-24. Llamaron a su creación "Sharc", se trataba nada menos que de una red de 40 neuronas artificiales que imitaban el cerebro de una rata. Cada neurona hacía el papel de una posición del laberinto y cuando se activaba daba a entender que la "rata" sabía en qué punto del laberinto estaba.

Las neuronas que estaban conectadas alrededor de la activada, hacían la función de alternativas a seguir por el cerebro, la activación de la siguiente neurona, es decir, la elección entre "derecha" o "izquierda" en este caso estaría dada por la fuerza de sus conexiones con la neurona activada. Por ejemplo, la "rata" completaba bien el recorrido eligiendo a partir de la quinta neurona la opción "izquierda" (que correspondería a la sexta), es entonces cuando las conexiones entre la quinta y sexta se hacen más fuertes (dicha conexión era realizada por el piloto automático), haciendo desde este momento más propensa esta decisión en un futuro. Pero las técnicas Skinnerianas (que eran las que se habían puesto en funcionamiento en esta red neuronal) no podrían llevar muy lejos a este nuevo engendro, ya que, en sí, esto no es inteligencia, pues la red neuronal nunca llegaría a trazar un plan.

1956 - Se organizó en Dartmouth la primera conferencia sobre IA. Aquí se discutió el uso potencial de las computadoras para simular "todos los aspectos del aprendizaje o cualquier otra característica de la inteligencia" y se presentó la primera simulación de una red neuronal, aunque todavía no se sabían interpretar los datos resultantes.

1959 - Widrow publica una teoría sobre la adaptación neuronal y unos modelos inspirados en esa teoría, el Adaline (Adaptative Linear Neuron) y el Madaline (Multiple Adaline). Estos modelos fueron usados en numerosas aplicaciones y permitieron usar, por primera vez, una red neuronal en un problema importante del mundo real: Filtros adaptables para eliminar ecos en las líneas telefónicas.

1962 - Rosemblatt publica los resultados de un ambicioso proyecto de investigación, el desarrollo del Perceptrón, un identificador de patrones ópticos binarios, y salida binaria. Las capacidades del Perceptrón se extendieron al desarrollar la regla de aprendizaje delta, que permitía emplear señales continuas de entrada y salida.

1969 - Minsky y Papert realizan una seria crítica del Perceptrón, revelando serias limitaciones, como su incapacidad para representar la función XOR, debido a su naturaleza lineal. Este trabajo creó serias dudas sobre las capacidades de los modelos conexionistas y provocó una caída en picada de las investigaciones.

1974 - Paul Werbos. Desarrolló la idea básica del algoritmo de aprendizaje de propagación hacia atrás (backpropagation); cuyo significado quedó definitivamente aclarado en 1985.

1977 - Stephen Grossberg: Teoría de Resonancia Adaptada (TRA). La Teoría de Resonancia Adaptada es una arquitectura de red que es diferente de todas las demás

previamente inventadas. La misma simula otras habilidades del cerebro: memoria a largo y corto plazo.

1985 - John Hopfield. Provocó el renacimiento de las redes neuronales con su libro: "Computación neuronal de decisiones en problemas de optimización."

1986 - David Rumelhart/G. Hinton. Redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation).

A partir de 1986, el panorama fue alentador con respecto a las investigaciones y el desarrollo de las redes neuronales. En la actualidad, son numerosos los trabajos que se realizan y publican cada año, las aplicaciones nuevas que surgen (sobre todo en el área de control) y las empresas que lanzan al mercado productos nuevos, tanto hardware como software (sobre todo para simulación).

1992 - Westland propuso un modelo de red neuronal autoorganizada en estos sistemas informativos, que filtra la información relevante para cada ejecutivo ya que la sobrecarga informativa es uno de los principales problemas de estos sistemas al estar basados en correo electrónico.

1993 - Por iniciativa de la London Business School, se celebró en Londres la primera reunión internacional sobre aplicaciones de redes neuronales al tratamiento de la información financiera.

1994 y 1995 - Martín y Serrano proponen un modelo híbrido que combina el modelo neuronal de mapas autoorganizados de Kohonen con otros modelos estadísticos y neuronales que obtienen una puntuación o *Z score*.

En un futuro cercano estarán disponibles artefactos inteligentes que tendrán altos coeficientes intelectuales y se verán completamente diferentes a las máquinas inteligentes actuales (video cámaras, lavadoras, etc). Habrá computadoras pequeñas y rápidas que serán comunes en nuestra vida, trabajo y esparcimiento. Los grandes sistemas y redes también se volverán inteligentes. Estarán enlazados entre sí los satélites de comunicaciones, redes de crédito y salud, entretenimiento y apuestas, automóviles, calles y redes de tráfico, redes de noticias y encuestas, redes gubernamentales y hasta redes de espionaje. En sí, las máquinas se reducirán y contarán con censores y procesadores de señales más finos. Y serán capaces de generar sus propias redes y reglas difusas.

Capítulo II

2. Redes Neuronales Biológicas

Para entender mejor las redes neuronales primero será necesario definir el cerebro y tener claro qué es una neurona y cómo como funciona. La teoría y modelado de redes neuronales artificiales está inspirada en la estructura y funcionamiento de los sistemas nerviosos. Donde la neurona es el elemento fundamental.

2.1. El cerebro como sistema

El funcionamiento del cerebro es similar al de un sistema. Para que un sistema funcione necesita de entradas y éstas, mediante algún proceso, logran generar una salida. El cerebro consiste de un gran número de elementos altamente conectados llamadas neuronas

2.2 ¿Qué es una neurona?

Se definirá lo que es y como funciona una neurona desde un punto de vista biológico. Muchos autores han proporcionado diferentes definiciones de las neuronas las cuales se muestran a continuación:

Una neurona es la unidad de procesamiento de información sobre la que se fundamenta la operación de una red neuronal artificial. ***(Xavier Padern)***

Las neuronas son los elementos básicos de la estructura cerebral, que se interconectan masivamente para formar estructuras altamente complejas, no lineales y con funcionamiento paralelo. (Ramón y Cajal(1911))

La neurona es la unidad funcional del sistema nervioso. Tomando éstas definiciones y redefiniendo lo que es una neurona la vamos a definir de la siguiente manera:

La neurona es la unidad fundamental del cerebro la cual se encarga de recibir impulsos y mediante un proceso se genera una reacción o salida.

La estructura básica de una neurona natural es:

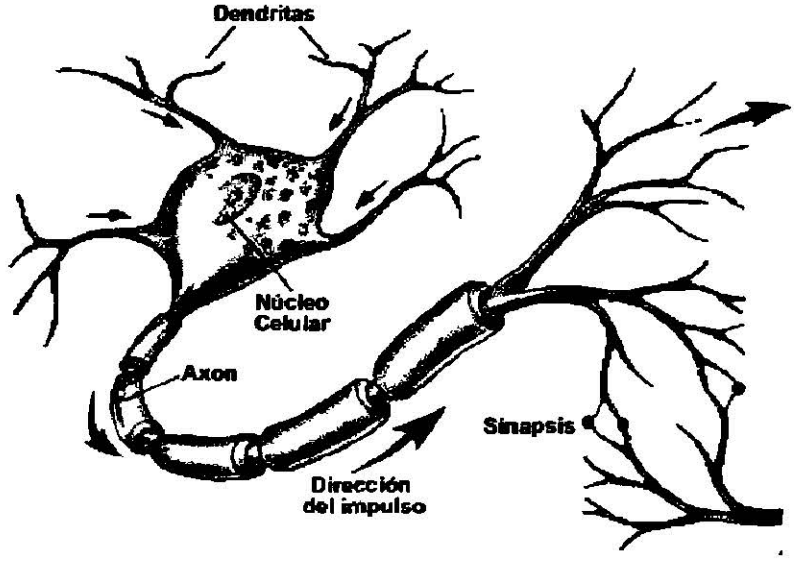


Figura 2.1_Estructura básica de una neurona

Dendritas: Ramificaciones nerviosas que trasmiten señales electroquímicas hacia el cuerpo de la célula. Su función es recoger los estímulos que llegan a la neurona

Cuerpo de la célula: Suma las señales electroquímicas de entrada proporcionadas por las dendritas.

Axón: Filamento que lleva la señal desde el cuerpo de la célula hacia otras neuronas.

Sinápsis: Punto de contacto entre un axón de una célula y una dendrita de otra célula.

Funcionamiento de una neurona

En esta parte se mostrará cómo se lleva acabo la función y las características de cada elemento de una neurona. Cada neurona puede tener infinitas entradas llamadas Dendritas que condicionan el estado de su única salida, el Axón. Este Axón puede ir conectado a una Dendrita de otra neurona mediante la Sinápsis correspondiente, como se muestra en la figura 2.2

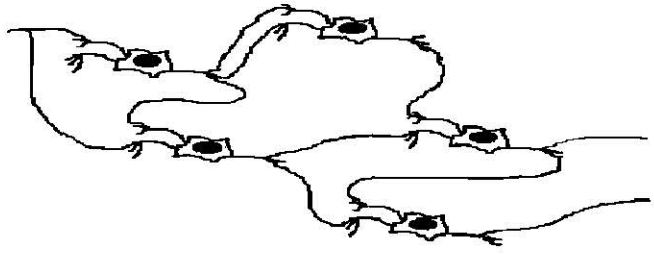


Figura 2.2 Representación de la sinapsis

El Axón da un nivel eléctrico dependiendo de las entradas y a la importancia que le da a cada una de ellas. Así, una neurona puede no reaccionar ante un nivel muy alto de una de sus entradas, o dar una salida muy favorable cuando otra de ellas está mínimamente activa.

2.3. Red neuronal

Una red neuronal es "un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona".

Los procesos del cuerpo humano se relacionan de una u otra manera con la actividad de las neuronas. Las mismas son unos componentes relativamente simples del ser humano, pero cuando millares de ellas se conectan en forma conjunta se hacen muy poderosas

Otro factor importante es que presenta un grado de adaptabilidad que se concreta en las capacidades de aprendizaje y generalización. Recordando un poco lo que es el aprendizaje, simplemente se define como la capacidad de resolver problemas basándose en información de experiencias pasadas. Íntimamente relacionada con el aprendizaje está la generalización, que podría definirse como la capacidad para abstraer la información útil, más allá de los casos particulares. De esta manera, la Red Neuronal es capaz de responder ante casos desconocidos.

Por lo tanto, las Redes Neuronales:

Consisten de unidades de procesamiento que intercambian datos o información.

Se utilizan para reconocer patrones, incluyendo imágenes, tendencias financieras, manuscritos y secuencias de tiempo. Tienen capacidad de aprender y mejorar su funcionamiento.

Una primera clasificación de los modelos de redes neuronales podría ser, atendiendo a su similitud con la realidad biológica:

1) El modelo de tipo biológico. Éste comprende las redes que tratan de simular los sistemas neuronales biológicos, así como las funciones auditivas o algunas funciones básicas de la visión. Además proponen un modelo matemático acerca del funcionamiento del cerebro

2) El modelo dirigido a aplicación. Este modelo no tiene por qué guardar similitud con los sistemas biológicos. Su arquitectura está fuertemente ligada a las necesidades de las aplicaciones para la que es diseñada. El conocimiento que se posee sobre el sistema nervioso en general no es completo y se deben de definir otras funcionalidades y estructuras de conexión distintas a las redes consideradas desde la perspectiva biológica.

Algunos puntos fuertes de este tipo de redes son:

Auto-organización y adaptividad: Ofrecen posibilidades de un proceso robusto y adaptativo, entrenamiento adaptativo y redes auto-organizadas.

Proceso no lineal: Aumenta la capacidad de la red de aproximar, clasificar y la inmunidad al ruido

Proceso paralelo: Normalmente se utiliza un gran número de células de proceso con altos niveles de interconectividad.

2.4. Cerebro Humano

El cerebro es capaz de realizar muchas tareas más rápido que cualquier computador convencional, debido en parte a su estructura masivamente paralela, donde sus neuronas están operando simultáneamente.

El sistema del cerebro es distribuido. Esto quiere decir que la información no se almacena localmente en ciertas zonas concretas de la Red Neuronal sino que se halla presente por toda ella, en concreto, se almacena en la sinapsis entre las neuronas. De igual forma, la computación es también distribuida. Al calcular la respuesta de la red neuronal, intervienen todos y cada uno de los nodos. Además, este caracter distribuido hace que la red presente tolerancia a fallos (si se pierde una parte de las neuronas no se pierde toda la información).

La figura 2.3 nos muestra cómo se lleva a cabo el funcionamiento del cerebro.

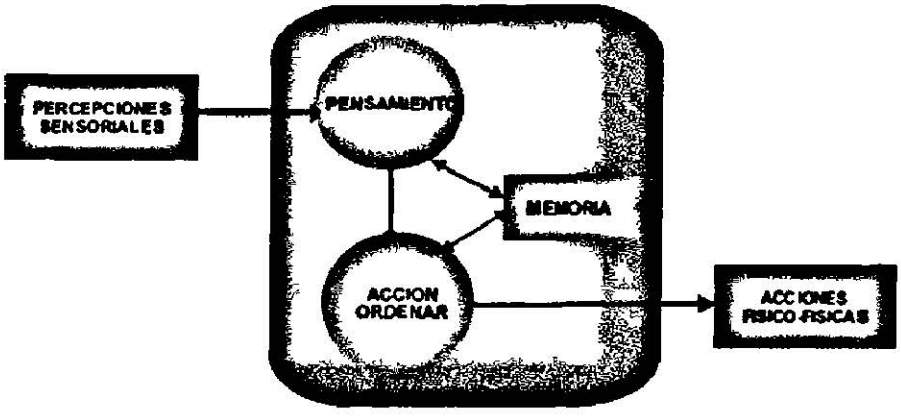


Figura 2.3 Representación del Funcionamiento del cerebro

Encontramos entonces que existen diversos estímulos al cerebro, los cuales pueden ser llamados "percepciones sensoriales". El ser humano es un sistema que parte de estímulos sensoriales, los cuales utiliza como entradas (inputs) para su proceso de pensamiento. La palabra percepción abarca no solo a las incitaciones que el medio ambiente provoca en los cinco sentidos sino también a las elucubraciones que la propia mente del sujeto pueda discernir.

El cerebro del sujeto es estimulado por la interrelación simultánea de percepciones, las cuales activan el proceso denominado pensamiento. Éste a su vez puede requerir vivencias almacenadas en la memoria del individuo, que funciona en este enfoque como

un archivo o base de datos que contiene información llamada en este contexto " Recuerdos ".

El proceso de pensamiento puede tener como subproducto la creación de datos a ser guardados, los cuales se transformarán a su vez en nuevos recuerdos.
Por ende, se puede caracterizar al pensamiento como un proceso de consulta y escritura de la base de datos " memoria ".

Basándose en estos procesos sobreviene el proceso de elaboración y planificación de la acción a tomar como resultado del proceso intelectual.
Esta acción puede ser física o psíquica. En el primer caso, músculos determinados del cuerpo son programados para moverse en un sentido definido.
Ejemplo: El aumento de la temperatura ambiente provoca en el individuo la percepción de calor y éste encamina sus acciones hacia la ingesta de líquido para no deshidratarse.
En el segundo caso, una percepción puede llevar a la recuperación de un recuerdo de la memoria (base de datos).
Ejemplo: La percepción de un aroma floral provoca el recuerdo de una placentera experiencia pasada ocurrida en un campo lleno de flores.

El cerebro está constituido por un vasto conjunto de células llamadas neuronas, las cuales se intercomunican entre sí por medio de impulsos eléctricos, mediante las llamadas conexiones sinápticas. Las neuronas forman, entonces, una red donde cada una de ellas es un nodo, entrelazada mediante conexiones que transmiten señales eléctricas.

Una neurona puede recibir y transmitir señales eléctricas desde y hacia varias otras neuronas.

El pensamiento entonces consiste físicamente en una serie de interacciones eléctricas entre las neuronas, las cuales cumplen las siguientes funciones básicas:

a) Recibir Señales: Cada neurona recibe uno o varios mensajes eléctricos emanados de las neuronas adyacentes.

b) Condicionar la descarga: La neurona actúa como una sistema distribuidor de tránsito que toma la energía trasladada hasta ella mediante la acción descripta en a) y decide cuáles de sus salidas activar y cuales no.

c) Descargarse (Emitir una respuesta) y reforzar conexiones. La energía es expulsada hacia las neuronas adyacentes a las cuales está conectada.
Conjuntamente con esto, las conexiones sinápticas se refuerzan. De esta manera, se establece una preferencia de salida proporcional a la cantidad de veces que se ha activado la misma.
En las primeras etapas de la vida, cuando se realiza el aprendizaje del cerebro, se entrena a nuestras neuronas mediante el éxito o fracaso de una acción a unos estímulos sensoriales. Cuando cierta acción realizada en respuesta a alguna entrada sensorial es exitosa (por ejemplo, al beber agua calma la sed), las conexiones sinápticas entre un grupo de neuronas se fortalecen, de manera que cuando se tenga una sensación sensorial

parecida, la salida será la correcta. De esta forma se forman fuertes conexiones entre grupos de neuronas, que pueden servir para realizar otras acciones complejas.

La principal importancia de haber visto las neuronas, redes neuronales, y el cerebro de modo detallado fue para tener un panorama más claro de cómo se encuentran constituidos y cómo se lleva acabo su funcionamiento. Ya que estos temas son de vital importancia para temas posteriores.

Capítulo III

3. Fundamentos de sistemas neuronales artificiales.

Este capítulo introducirá una nomenclatura fundamental y los conceptos matemáticos básicos que describen y analizan los procesos de los sistemas neuronales artificiales.

Una red artificial consiste en un conjunto de elementos simples de proceso conectados entre sí y entre los que se envían información a través de las conexiones:

- Conjunto de unidades de procesamiento.
- Conexiones entre unidades (asociando a cada conexión un peso o valor).
- Función de salida y activación para cada unidad de procesamiento.

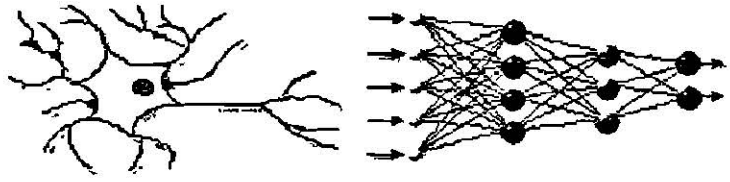


Figura 3.1 Representación de una Red artificial

3.1. Definición de redes neuronales.

A lo largo de todo este tiempo se han manejado diversas definiciones de las redes neuronales, se mostrarán a continuación algunas de ellas.

Las redes neuronales, también llamadas "redes de neuronas artificiales", son modelos bastante simplificados de las redes de neuronas que forman el cerebro. Y, al igual que éste, intentan "aprender" a partir de los datos que se le suministran.

Una red neuronal es un procesador masivamente paralelo distribuido que es propenso por naturaleza a almacenar conocimiento experimental y hacerlo disponible para su uso. Este mecanismo se parece al cerebro en dos aspectos:

El conocimiento es adquirido por la red a través de un proceso que se denomina aprendizaje. El conocimiento se almacena mediante la modificación de la fuerza o peso sináptico de las distintas uniones entre neuronas.

Una red neuronal es un modelo computacional con un conjunto de propiedades específicas, como son la habilidad de adaptarse o aprender, generalizar u organizar la información, todo ello basado en un procesamiento eminentemente paralelo.

Una red neuronal es un modelo computacional que comparte algunas de las características del cerebro; consta de varias unidades sencillas que trabajan en paralelo sin un control central. Las conexiones entre las unidades tienen pesos numéricos que los elementos de aprendizaje pueden modificar.

Una red neuronal artificial (RNA) es un modelo computacional inspirado en redes neuronales biológicas que pueden ser consideradas como un sistema de procesamiento de información con características como: aprendizaje a través de ejemplos, adaptabilidad, robustez, capacidad de generalización y tolerancia a fallas.

Si redefinimos las redes neuronales artificiales su definición seria la siguiente: Es un modelo computacional capaz de procesar información y aprender a través del tiempo de experiencias pasadas.

3.2.Modelo de una neurona.

El modelo de Neurona y la arquitectura de una red Neuronal, describen cómo la red transforma sus entradas en las salidas. Todo esto, puede ser visto simplemente como cálculos.

Los elementos individuales de cálculo que forman la mayoría de los modelos de sistemas neuronales artificiales, reciben el nombre de elemento de procesamiento o Neurona Artificial.

Analizando el modelo matemático más simple de una red neuronal se compone de:

Un conjunto de sinapsis o conexiones caracterizadas individualmente por un peso o fuerza de la conexión w_{kj}.

Si $w_{kj} > 0$ la sinapsis es excitadora y si $w_{kj} < 0$ es inhibitoria.

Un sumador que combina linealmente las señales de entrada.

Una función de activación que limita la amplitud de la salida de una neurona, normalmente (0,1) ó (-1,1), y un umbral o sesgo que ajusta el punto de activación.

Matemáticamente: $$u_k = \sum_{j=1}^{N} w_{kj} x_j$$

$y_k = u_k - \theta_k$

$y_k = f(v_k)$

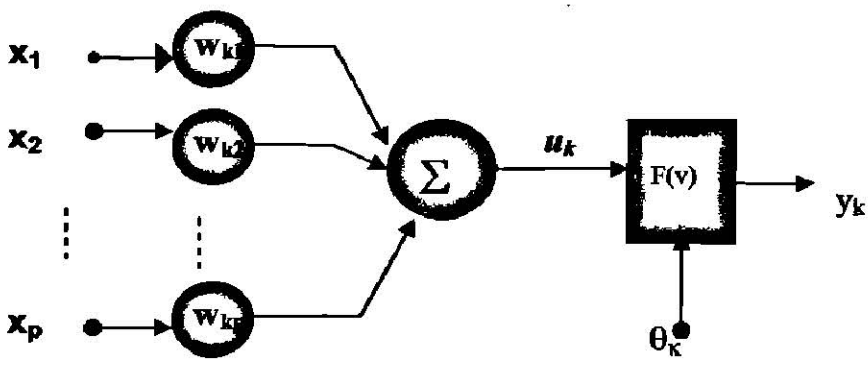


Figura 3.2 Representación Matemática de una Red Neuronal

Analizando el Modelo de una red neuronal desde un punto de vista más general.

Una neurona es la unidad de procesamiento de información sobre la que se fundamenta la operación de una red neuronal artificial. El diagrama de la Figura 3.2 muestra el modelo básico de una neurona donde pueden identificarse tres elementos característicos:

1. Un conjunto de conexiones o sinapsis, caracterizada cada una de ellas por un peso sináptico, de forma que la señal de entrada xj, presente en la neurona k, se verá multiplicada por el peso sináptico wjk. Este peso, puede variar en un rango que incluye tanto a valores negativos (entrada inhibidora) como valores positivos (entrada excitadora).

2. Una regla de propagación que determina la entrada efectiva o nivel de excitación de la neurona k, denotada por sk, a partir de todas las entradas individuales a la misma.

Habitualmente, se considera como entrada efectiva a la suma de todas las señales de entrada xj a la neurona k, ponderadas por sus correspondientes pesos sinápticos wjk.

3. Una función de activación, denotada por Fk, que determina el estado o salida yk de la neurona a partir del nivel de excitación de la neurona. El objetivo de esta función (que generalmente presenta un comportamiento no lineal), es limitar la amplitud de la señal de salida dentro de un rango de valores normalizado (por ejemplo, dentro del intervalo [0,1] o [-1,1]). El modelo de la Figura 3.3 incluye una entrada externa adicional, denominada polarización o "bias" y denotada por bk, cuyo fin es el de poder aumentar o disminuir el umbral de excitación de la neurona dependiendo de si es un valor positivo o negativo, respectivamente.

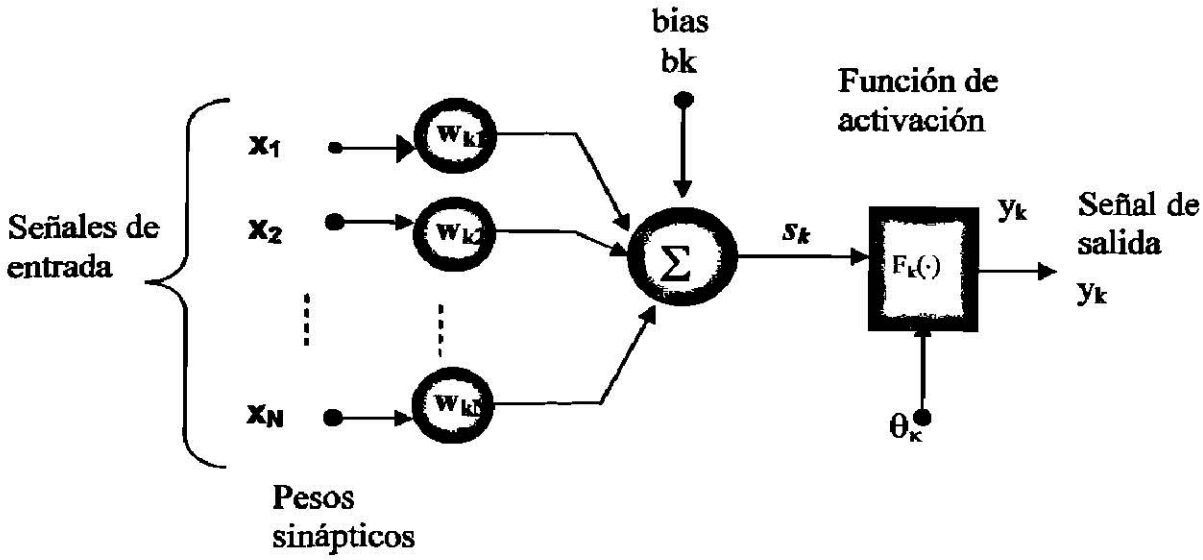


Figura 3.3. Modelo genérico de una neurona.

Podemos describir Matemáticamente el modelo básico de una neurona, mediante el siguiente par de ecuaciones:

$$S_k = \sum_{j=1}^{N} wjk\ x\ j + b \qquad (3.1)$$

$$y_k = F_k(S_k) \qquad (3.2)$$

Considerando, la polarización bk como el peso sináptico de una entrada adicional cuyo valor de entrada siempre es 1, podemos reescribir (3.1) de una forma más homogénea, como sigue:

$$s_k = \sum_{j=0}^{N} w_{jk} x_j \qquad (3.3)$$

Por otro lado, siendo $w_k = (b_k, w_{1k}, \ldots, w_{jk}, \ldots, w_{Nk})^T$ el vector de pesos y $x = (1, x_1, \ldots, x_N)^T$ el vector de entrada, podemos escribir de forma vectorial que

$$yk = Fk(w_k^T \cdot x) \qquad (3.4)$$

3.3. Función de activación (activation function).

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un "estado de activación". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado. La función de activación calcula el estado de actividad de una neurona.

3.4.Tipos de funciones de activación.

La función de activación, denotada por $F_k(S_k)$, define el nuevo estado o salida de la neurona en términos del nivel de excitación de la misma. A continuación se describen varios tipos de función de activación.

3.4.1 Función umbral.

Una de las funciones de activación típicas, es una particularización de la función unitaria de Heaviside, también llamada función umbral (threshold function), definida como sigue:

$$y_k = F_k(s_k) = \begin{cases} 1 & si\ s_k \geq 0 \\ 0 & si\ s_k < 0 \end{cases} \qquad (3.5)$$

A las neuronas con este tipo de función de activación, se les conoce en la literatura como neuronas de McCulloch-Pitts, y su importancia es fundamentalmente histórica ya que se considera que marcan el principio de la disciplina de las redes neuronales artificiales (McCulloch y Pitts, 1943). Este tipo de neurona fue introducido como un modelo simplificado de las neuronas biológicas, así como un componente conceptual útil para construir circuitos que pudieran realizar tareas computacionales.

3.4.2. Función semilineal.

La función semilineal (piecewise function) se define como:

$$y_k = F_k(s_k) = \begin{cases} 1 & si\ S_k \geq \frac{1}{2a} \\ as_k + \frac{1}{2} & si\ -\frac{1}{2a} < S_k < \frac{1}{2a} \\ 0 & si\ S_k \leq -\frac{1}{2a} \end{cases} \qquad (3.6)$$

Donde el parámetro a determina la pendiente de la recta definida en el tramo $-\frac{1}{2a} \leq Sk \leq \frac{1}{2a}$.

En este caso, además de las zonas de saturación (salida 1) y no saturación (salida 0), se considera una zona de operación lineal en torno al umbral de excitación de la neurona (en este caso, valor 0 para la entrada efectiva). En la zona lineal, la salida es proporcional (en un factor a o ganancia) al valor de la entrada efectiva, que no es más que una combinación lineal de los diferentes valores de entrada.

Las neuronas que incorporan este tipo de función de activación pueden considerarse como una aproximación de un amplificador no lineal. Asimismo, la función semilineal tiende a adoptar la forma de la función umbral cuando la ganancia a tiende a infinito.

3.4.3. Función sigmoidal.

Las funciones de activación más comunes a la hora de construir redes neuronales artificiales son las sigmoidales (sigmoid function), es decir, funciones con forma de "s", estrictamente crecientes, suaves (que se manifiesta en el hecho de ser funciones diferenciables) y que presentan un equilibrio adecuado entre el comportamiento lineal y no lineal. Un ejemplo de función sigmoidal típica es la función conocida como logística, definida como:

$$y_k = F_k(s_k) = \frac{1}{1+e^{-as_k}} \qquad (3.7)$$

El rango de variación de dicha función es el intervalo [0,1], y al parámetro a se le denomina pendiente de la curva. Este valor determina su forma de modo que si, por ejemplo, a tiende hacia infinito, la función logística tiende a alcanzar la forma de la función umbral. Asimismo, dicha función es diferenciable, lo cual es una característica importante desde el punto de vista teórico de las redes neuronales artificiales.

Todas las funciones de activación definidas previamente, tienen su rango de variación en el intervalo [0,1], si bien a veces, se requiere que dicho rango sea el intervalo [-1,1]. En esta situación en vez de considerar la función umbral, se considera la función signo (simétrica de la función umbral respecto al origen), es decir:

$$y_k = F_k(s_k) = \begin{cases} -1 & si\ S_k > 0 \\ 0 & si\ S_k = 0 \\ 1 & si\ S_k < 0 \end{cases} \qquad (3.8)$$

En el caso de la función semilineal, considerando que la pendiente de la recta en la región lineal de operación viene determinada por el valor del parámetro a, ésta se redefine como sigue:

$$y_k = F_k(s_k) = \begin{cases} 1 & si\ S_k \geq \frac{1}{a} \\ as_k & si\ -\frac{1}{a} < S_k < \frac{1}{a} \\ -1 & si\ S_k \leq -\frac{1}{a} \end{cases} \qquad (3.9)$$

En el caso de las funciones sigmoidales, la función típica que se utiliza para limitar la salida dentro del intervalo [-1, 1] es la función tangente hiperbólica, que se define como:

$$y_k = F_k(s_k) = \tanh(as_k) = \frac{e^{as_k} - e^{-as_k}}{e^{as_k} + e^{-as_k}} \qquad (3.10)$$

3.4.4. Función lineal.

Un caso especial de función de activación, es considerar la función identidad o función lineal (linear function), en cuyo caso la salida de la neurona k coincide exactamente con la entrada efectiva a la misma, es decir:

$$y_k = F_k(s_k) = S_k \qquad (3.11)$$

3.4.5. Función de activación estocástica.

Los tipos de función de activación, expuestos hasta ahora, son deterministas. Es decir, la salida está definida con total precisión para todos los valores de entrada. Sin embargo, en algunas aplicaciones de las redes neuronales artificiales es deseable contemplar un modelo de neurona estocástica, en el sentido de quc la decisión de "disparar" o no la misma sea probabilística. En este sentido, se puede escribir que

$$y_k = \begin{cases} 1 \ \ con \ \ probabilid\,ad & p(s_k) \\ -1 \ \ con \ \ probabilid\,ad & 1 - p(s_k) \end{cases} \qquad (3.12)$$

Una elección estándar para $p(s_k)$ es la siguiente función sigmoidal, (Little, 1974)

donde T es una "pseudotemperatura" que se usa para controlar el nivel de ruido y por tanto, expresa la incertidumbre sobre la activación de la neurona. Cuando T tiende a 0, este modelo estocástico de neurona se reduce a la forma determinista, es decir, al modelo de McCulloch-Pitts (función lineal)

$$p(s_k) = 1 \frac{1}{1 + e^{-\frac{s_k}{T}}} \qquad (3.13)$$

Capítulo IV

4. Aplicaciones de las redes neuronales artificiales

Desde el punto de vista de las aplicaciones prácticas de las RNA, su principal ventaja frente a otras técnicas reside en el procesado paralelo, adaptativo y no lineal. Se han desarrollado aplicaciones de RNA para fines tan variados como visión artificial, procesamiento de señales e imágenes, reconocimiento del habla y de caracteres, sistemas expertos, análisis de imágenes médicas, control remoto, control de robots, inspección industrial y exploración científica.

Podríamos clasificar todas estas aplicaciones en varios dominios o tipos de aplicaciones, indicando además el tipo de redes más utilizado para cada tarea:

* Asociación y clasificación: heteroasociadores y redes competitivas.
* Regeneración / reconstrucción de patrones: redes de satisfacción de demanda.
* Regresión y generalización: heteroasociadores.
* Optimización: heteroasociadores.

Separándolas según las distintas disciplinas, algunos ejemplos de sus aplicaciones son:

Biología:

Aprender más acerca del cerebro y otros sistemas.

Obtención de modelos de la retina.

Empresa:

Reconocimiento de caracteres escritos.

Identificación de candidatos para posiciones específicas.

Optimización de plazas y horarios en líneas de vuelo.

Explotación de bases de datos.

Evaluación de probabilidad de formaciones geológicas y petrolíferas.

Síntesis de voz desde texto.

Medio Ambiente.

Analizar tendencias y patrones.

Previsión del tiempo.

Finanzas:

Previsión de la evolución de los precios.

Valoración del riesgo de los créditos.

Identificación de falsificaciones.

Interpretación de firmas.

Manufactura:

Robots automatizados y sistemas de control (visión artificial y censores de presión, temperatura, gas, etc.).

Control de producción en líneas de proceso.

Inspección de calidad.

Filtrado de señales.

Medicina:

Analizadores del habla para la ayuda de audición de sordos profundos.

Diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (encefalograma, etc.).

Monitorización en cirugía.

Predicción de reacciones adversas a los medicamentos.

Lectoras de Rayos X.

Entendimiento de causa de ataques epilépticos.

Militares:

Clasificación de las señales de radar.

Creación de armas inteligentes.

Optimización del uso de recursos escasos.

La mayoría de las aplicaciones comentadas aquí se han desarrollado mediante asociadores de patrones y algoritmo de propagación posterior. Otras, como el problema del viajante de comercio, el control adaptativo y la compresión de imágenes, mediante redes de satisfacción de demanda (redes de Hopfield), y en algunas tareas de clasificación se han usado redes competitivas.

Comentemos acerca de algunas aplicaciones y cómo se desarrollan:

4.1. Procesamiento de lenguaje natural.

Conversión de texto escrito a lenguaje hablado.

NETtalk (Sejnowski T. & Rosemberg.): toma como entradas textos escritos y como salidas deseadas los códigos elegidos para representar los fonemas correspondientes. Mediante la ayuda de un sintetizador (DECtalk) se transforman los códigos en fonemas. Durante el proceso de aprendizaje se observó como iba mejorando su habilidad desde un nivel de bebé hasta el nivel de un niño de 6 años, aprendiendo a hacer distinciones difíciles como pronunciar una c suave o fuerte según el contexto. Si bien esto se había conseguido antes, la novedad más importante reside en que mediante la red neuronal no es necesario definir y programar un montón de complejas reglas, pues la red extrae automáticamente el conocimiento necesario.

Aprendizaje de gramáticas.

(Rumelhart,D. & McClelland, J.) Estudiaron la forma en que construimos las reglas sobre el lenguaje, y trataron de enseñar a una red neuronal el pasado de los verbos ingleses. El sistema fue mejorando y al final era capaz de generalizar y conjugar verbos desconocidos.

Compresión de imágenes. No hay referencia

(Cottrell, G.W. y otros) Han conseguido codificar imágenes con una relación de compresión de hasta 8:1 sin tener que idear ninguna regla y alta fidelidad en la reconstrucción.

Reconocimiento de caracteres.

Reconocimiento de escritura manual.

Nestor, Inc => Leen lo escrito mediante una tarjeta digitalizadora. Tras aprender, son capaces de reconocer escrituras que nunca habían visto antes. Se ha empleado por ejemplo para reconocer kanji (escritura japonesa), eliminando la gran dificultad que presenta este lenguaje para introducirlo en el computador.

El Neocognitrón (Kunihiko Fukishima): simula la forma en que la información visual avanza en la corteza cerebral. Consigue un reconocimiento muy avanzado de patrones con gran capacidad de abstracción y generalización, que lo hacen capaz de reconocer patrones con distinta orientación y altos niveles de distorsión.

4.2. Reconocimiento de patrones en imágenes.

Clasificación de objetivos: En este campo se han desarrollado numerosas aplicaciones como la clasificación de imágenes de sonar y radar, la detección de células cancerosas, lesiones neurológicas y cardíacas, prospecciones geológicas, etc.. Son muy útiles para procesar imágenes de las que no se sabe bien cuales son las características esenciales o

diferenciales, ya que las redes no necesitan disponer de reglas explícitas previas para realizar la clasificación, sino que extraen el conocimiento necesario.

4.3. Visión artificial en robots industriales.

Predicción

Se han obtenido mejores resultados a la hora de predecir series "caóticas" usando retropropagación (Lapedes & Farber) que mediante métodos lineales y polinomiales.

4.4. Filtro de ruido.

Las redes neuronales artificiales son mejores preservando la estructura profunda y el detalle, que los filtros tradicionales cuando eliminan el ruido. La primera aplicación profesional de las redes neuronales consistió en un filtro para eliminar ruido en las líneas telefónicas (Widrow, 1959).

Capítulo V

5. Implementación en aplicaciones.

Como es sabido los sistemas computacionales y la mayoría de cuestiones tecnológicas son propensas a tener fallos pero las redes neuronales son la excepción. Se mostrará a continuación que las redes neuronales son tolerantes a fallos, también se verá la forma en que funcionan y por último, el tipo de tecnología que permite una mejor aplicación de las redes neuronales.

5.1. Tolerancia a fallos

Las redes neuronales fueron los primeros métodos computacionales con la capacidad inherente de tolerancia a fallos. Comparados con los sistemas computacionales tradicionales, los cuales pierden su funcionalidad cuando sufren algún pequeño error de memoria, en las redes neuronales, si se produce un fallo en un número no muy grande de neuronas y aunque el comportamiento del sistema se ve influenciado, no sufre una caída repentina y/o completa.

Hay dos aspectos distintos respecto a la tolerancia a fallos:

a) Las redes pueden aprender a reconocer patrones con ruido, distorsionados o incompletos. Esta es una tolerancia a fallos respecto a los datos.

b) Las redes pueden seguir realizando su función (con cierta disminución en su nivel de desempeño) aunque se destruya parte de la red.

La razón por la que las redes neuronales son tolerantes a los fallos es que tienen su información distribuida en las conexiones entre neuronas, existiendo cierto grado de redundancia en este tipo de almacenamiento. La mayoría de los ordenadores algorítmicos y sistemas de recuperación de datos almacenan cada pieza de información en un espacio único, localizado y direccionable. En cambio, las redes neuronales almacenan información no localizada. Por lo tanto, la mayoría de las interconexiones entre los nodos de la red tendrán sus valores en función de los estímulos recibidos, y se generará un patrón de salida que represente la información almacenada.

Operación en tiempo real

Una de las mayores prioridades, casi en la totalidad de las áreas de aplicación, es la necesidad de realizar procesos con datos de forma muy rápida. Las redes neuronales se adaptan bien a esto debido a su implementación paralela. Para que la mayoría de las redes puedan operar en un entorno de tiempo real, la necesidad de cambio en los pesos de las conexiones o entrenamiento es mínimo.

5.2. Fácil inserción dentro de la tecnología existente

Una red individual puede ser entrenada para desarrollar una única y bien definida tarea (tareas complejas, que hagan múltiples selecciones de patrones, requerirán sistemas de redes interconectadas). Con las herramientas computacionales existentes (no del tipo

PC), una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a una implementación hardware de bajo costo.

5.3.Redes neuronales y computadoras digitales.

Para entender el potencial de la computación neuronal, sería necesario hacer una breve distinción entre sistemas de computación neuronales y digitales: los sistemas neurológicos no aplican principios de circuitos lógicos o digitales.

Un sistema de computación digital debe ser síncrono o asíncrono. Si fuera asíncrono, la duración de los impulsos neuronales debería ser variable para mantener uno de los valores binarios por periodos de tiempo indefinido, lo cual no es el caso.

Si en un inicio fuera síncrono, se necesitaría un reloj global o maestro con el cual los pulsos estén sincronizados. Éste tampoco es el caso. Las neuronas no pueden ser circuitos de umbral lógico, porque hay miles de entradas variables en la mayoría de las neuronas y el umbral es variable con el tiempo, siendo afectado por la estimulación, atenuación, etc.

La precisión y estabilidad de tales circuitos no es suficiente para definir ninguna función booleana. Los procesos colectivos que son importantes en computación neuronal no pueden implementarse por computación digital. Por todo ello, el cerebro debe ser un computador analógico.

Ni las neuronas ni las sinapsis son elementos de memoria biestable. Todos los hechos fisiológicos hablan a favor de las acciones de las neuronas como integradores analógicos, y la eficiencia de la sinapsis cambia de forma gradual, lo cual no es característico de sistemas biestables.

Los circuitos del cerebro no implementan computación recursiva y por lo tanto no son algorítmicos. Debido a los problemas de estabilidad, los circuitos neuronales no son suficientemente estables para definiciones recursivas de funciones como en computación digital. Un algoritmo, por definición, define una función recursiva.

5.4. Programando la Neurona en Pseudocódigo.

Los programadores de computadora utilizan determinadas técnicas para poder representar mejor su trabajo. Una de ellas es la utilización del llamado Pseudocódigo. Este se utiliza para determinar los pasos lógicos que se deberían utilizar para realizar un programa independientemente del lenguaje elegido.

PROGRAMA NEURONA

Si quisiéramos escribir el programa maestro con el cual funciona la célula tendríamos como resultado una estructura similar a la siguiente.

1. HACER MIENTRAS SE RECIBAN IMPULSOS DE LAS NEURONAS VECINAS.
2. RECOGER IMPULSO ELÉCTRICO.
3. DECIDIR A CUALES NEURONAS SE DEBE DESCARGAR LA ENERGÍA.
4. HACER MIENTRAS QUEDEN NEURONAS SIN HABER RECIBIDO IMPULSO.
5. SELECCIONAR NEURONA RECEPTORA.
6. DESCARGAR ENERGÍA A LA NEURONA RECEPTORA.
7. REFORZAR CONEXIÓN CON ESTA NEURONA.
8. VOLVER A SELECCIONAR NEURONA.
9. VOLVER A EMPEZAR.

Este proceso continúa hasta que no existen mas neuronas receptoras, a menos que las neuronas están conectadas a una terminal nerviosa (ojos, medula espinal etc.), lo cual activa la secuencia de acción correspondiente.

De estas afirmaciones se desprende la existencia de niveles o capas de neuronas, que tienen como función la potenciación o disminución de los impulsos iniciales. Esto es

coherente con las investigaciones de la fisiología cerebral, que coinciden en la existencia de diferentes zonas del cerebro especializadas en funciones determinadas.

Memoria

La memoria consiste desde este enfoque en la suma de los pesos de las ponderaciones o preferencias correspondientes a las conexiones en un conjunto o capa de neuronas relacionadas. La activación de este sector por un impulso nervioso obtiene como resultado que la última capa del sector (las neuronas relacionadas con otras no pertenecientes al mismo) ofrezca una salida hacia los centros neuronales que provocarán la sensación de un "recuerdo".

Tenemos entonces que el mismo elemento (la neurona) actúa como conductor de señales, programa de acción y memoria. En términos computacionales hablaríamos de bus de datos para la conexión sináptica, software de aplicación para decidir cuándo se carga y descarga el flujo eléctrico y memoria RAM y disco rígido para resguardar los valores acumulados de las cargas energéticas.

Axiomas para la simulación computacional del proceso biológico.

Para poder simular estos procesos mediante dispositivos computacionales se deben elaborar los siguientes axiomas, sin los cuales no tiene sentido la misma.

Axioma principal

No es importante la sustancia física de la cual están hechas las neuronas, sino su función. Las neuronas, al igual que el resto de la estructura de los seres vivos están compuestas por cadenas de átomos de carbono entrelazadas. Sin embargo, la composición química de las mismas no es un factor determinante a la hora de tratar de duplicarlas, sino su función. Esto lleva a la idea de que es posible crear una neurona artificial simulando sus funciones en un dispositivo conveniente como una computadora. De no existir este supuesto, se tendría que suspender el estudio de las redes neuronales artificiales ya que no se podría fabricarlas. Este axioma nos lleva a pensar en los tejidos cerebrales como una especie de "redes de carbono", con lo que algunas de sus funciones pueden ser simuladas por dispositivos creados en otro material. Una rueda puede estar hecha en madera, metal o caucho vulcanizado, pero siempre es una rueda.

Axioma secundario

El comportamiento "inteligente" (limitado en este caso al reconocimiento de patrones) está dado por la interrelación de los mensajes e interacciones de los nodos de la red.

_ Esto significa que de tenerse un sustituto artificial de la neurona y de aplicarse un mecanismo de interacción entre las mismas similar al utilizado por el cerebro humano, se podrían duplicar o simular algunas de las funciones abstractas del mismo. De no contarse con este axioma, tampoco sería posible proseguir con la investigación ya que no habría forma de reproducir el comportamiento aunque existieran neuronas artificiales. Considerando entonces la aplicabilidad de los teoremas enunciados, resta entonces definir equivalentes computacionales de los procesos descriptos anteriormente.

5.5. Equivalentes artificiales de los dispositivos biológicos

El cuadro siguiente muestra la equivalencia entre los elementos artificiales y los biológicos a fines de reproducir la trama neuronal

Elemento biológico	Elemento artificial
Neurona	NODO DE LA RED = POSICIÓN DE MEMORIA RAM + ESPACIO DISPONIBLE DE DISCO RÍGIDO + PROGRAMA DE SOFTWARE NEURONAL.
Entradas Sensoriales	DIGITALIZACION DE IMÁGENES. ARCHIVOS DE COMPUTADORA DE DIVERSOS FORMATOS.
Salidas Sensoriales omportamiento	CREACIÓN DE ARCHIVO DE COMPUTADORA CON DATOS PROCESADOS. ACTIVACIÓN DE PROGRAMA DE COMPUTADORA PREDEFINIDO. ACTIVACIÓN DE INTERFACES ROBOTICAS ADECUADAS.
Interconexión Sináptica	PREFERENCIAS Y PONDERACIONES DE LA TRANSFERENCIA DE VALORES ENTRE LAS DISTINTAS POSICIONES DE MEMORIA.

5.6. Pasos para la ejecución de una red neuronal

El proceso artificial consiste en crear un programa de computadora (software) que sea capaz de:

- Obtener datos digitalizados.
- Generar múltiples espacios de memoria. Esto se puede hacer mediante la creación de múltiples vectores.
- Dar a los mismos un orden de preferencia o ponderación simbolizando de este modo las distintas capas de neuronas.
- Asignar a la primera capa el status de capa de entrada, la cual se nutrirá entonces de los datos de entrada.
- Asignar a la última capa el status de capa de salida, permitiendo que sus valores sean transformados por la interacción de los demás elementos de la red.
- Asignar a las capas intermedias el status de capas ocultas, permitiendo que sus valores sean modificados por la interacción de los demás elementos de la red.
- Propagar hacia adelante: Realizar operaciones de transferencia entre los datos de las capas simulando de este modo las interconexiones sinápticas.
- Recolectar el valor remanente de la capa de salida, lo cual será tomado como el valor de la red para la entrada determinada.
- Una vez obtenida la salida de la red, asignar una acción determinada para la misma, ya sea la creación de un archivo con la información, o la activación de un dispositivo automático para la computadora.

Paso 1. Obtención de Datos Digitalizados.

La red neuronal debe partir como base del equivalente artificial de un estímulo para poder empezar a operar. En el organismo viviente éste está dado por una excitación de cualquiera de los sentidos o de una combinación de la excitación de varios de ellos.
En la computadora, este proceso debe dividirse en dos partes:

a) El programa de redes neuronales debe estar prendido, es decir se requiere una activación externa para que funcione, cosa que es automática en los seres vivos.

b) Se requiere una entrada, es decir un patrón al cual reconocer.
Cualquier entrada debe ser convertida a formato digital para de esa manera poder ser transformada en un archivo susceptible de ser cargado en la computadora.

Digitalizar información consiste en transformar la misma en una cadena (strings en inglés) de ceros y unos llamada: cadena de caracteres binarios. Esta cadena se almacena en un archivo o base de datos que guarde los datos de origen. Se denomina genéricamente a estos datos "Trama de Entrada".

Paso 2. Generación de espacios de memoria.

Una de las formas de simular artificialmente neuronas es la de asimilarlas a elementos de vectores o "arrays". Cualquier lenguaje de programación ofrece la posibilidad de crear vectores n-dimensionales llamados también "arrays". La cantidad de vectores depende de la arquitectura de la red a elegir. El siguiente ejemplo se ha realizado pensando en la arquitectura de una red de retropropagación con una capa oculta (Hidden Layer Backpropagation Network).

El número inicial de vectores involucrados será entonces:

Un vector para almacenar la trama de entrada cuyo número de elementos coincide con el número de elementos de dicha trama.

Ecuación 1:

$$E_1=(e_1..e_n). \qquad (?)$$

También un vector para almacenar la capa oculta de tamaño variable (que puede coincidir con el tamaño de la trama de entrada).

Ecuación 2:

$$\prod = (h_1..h_n). \qquad (?)$$

Además un vector para almacenar la trama de salida de un número de elementos variable según la cantidad que deba tener la red.

Ecuación 3:

$$\Omega = (s_1..s_n). \qquad (?)$$

En este paso se dimensionan (crean) los vectores y se les añade valores iniciales, generalmente ceros.

Paso 3. Orden de preferencia o peso para las capas de neuronas.

Como se mencionó en párrafos anteriores, la red trata de simular el movimiento de interconexión sináptica entre distintas capas neuronales, reforzando (ponderando) las conexiones más frecuentes, de tal manera que se cree una ruta más potente dirigida a la salida adecuada.

Esto se logra anexando a cada valor de la capa una ponderación llamada "peso" en el vocabulario específico.

Los pesos se conforman entonces como valores, los cuales se multiplican por los valores de las capas para lograr las "salidas" o resultados que se traspasan de una capa a otra.

Ecuación 4:

$$\Omega = \sum_{i=1}^{n} E(i) * \prod(i). \qquad (?)$$

Los pesos o valores de ponderación se establecen *a priori* de la ejecución efectiva de la red mediante un proceso llamado "entrenamiento" el cual se comentará a *posteriori.*

Para ejecutar una red, entonces se parte de la base que la red ya fue entrenada, lo que implica que existen guardados en una base de datos los valores iniciales de los pesos de los valores. Estos pesos se deben almacenar en vectores, los cuales habrá que dimensionar (crear), por lo que habrá que anexar más vectores a las definiciones explicadas en el paso anterior.

La red necesitará entonces:

Un vector para almacenar los pesos de las conexiones de la capa de entrada a la capa oculta. Este vector tendrá la misma cantidad de elementos que la trama de entrada y a su

vez tendrá almacenados las ponderaciones relativas a los patrones que la red puede reconocer, por lo que será un vector de dos dimensiones.

Ejemplo: Si la trama de entrada tiene 1000 elementos y la red puede reconocer 8 patrones diferentes, el vector tendrá una primera dimensión de 1000 y una segunda dimensión de 8.

Ecuación 5:

$$T(x,y)=(x_1..x_n,y_1,..y_n)\,.$$

En términos computacionales deberemos programar:

DIM PESOS (1000.8) para un lenguaje visual (para Windows) como VISUAL BASIC.

Un vector para almacenar los pesos de las conexiones de la capa oculta a la capa de salida.

Ecuación 6:

$$\Delta=(p_1,...p_n)\,.$$

Este vector debe contener al menos tantos elementos como los del vector de salidas.

Paso 4. Asignar al primer arreglo el status de capa de entrada, la cual se nutrirá de la trama a reconocer.

Los valores de la trama de entrada se encuentran en este punto en un archivo proveniente de la digitalización de los mismos. Se procede a cargarlos en el vector de entradas de la red. Al concluir este paso tendremos a la red lista para empezar a calcular.

Paso 5. Asignar al último arreglo el status de capa de salida, permitiendo que sus valores sean transformados por la interacción de los demás elementos de la red.

El vector de "Salidas" ya ha sido creado. Si no fueron dados valores iniciales, se debe proceder a realizar esto dando valor 0 a los elementos constitutivos.

Paso 6: Asignar a los arreglos intermedios el status de "capas ocultas", permitiendo que sus valores sean transformados por la interacción de los demás elementos de la red.

Se repite el proceso del paso 5 para las capas intermedias.

Paso 7. Propagar hacia adelante: Realizar operaciones de transferencia entre los datos de los vectores simulando de este modo las interconexiones sinápticas.

La transferencia entre datos de los vectores recibe el nombre de propagación. En el caso de la propagación hacia adelante consiste en obtener:

Partiendo de la trama de entrada que reside en el vector de entradas, y teniendo como ponderación a los valores de la "capa de pesos" , residentes en los vectores de "pesos de la capa de entrada" obtener valores para la capa oculta.

Almacenar estos valores en el vector de salidas de la red.

Los valores concretos a transferir a la siguiente capa se obtienen mediante la aplicación de una función de activación a los datos y luego a la multiplicación del resultante por los "pesos" correspondientes.

Una función de activación consiste en una función matemática que se aplica a los datos originales para obtener un resultado susceptible de poder ponderarse para generar el valor de entrada de la capa siguiente. Este peso simula la fuerza o debilidad de la conexión sináptica entre las neuronas. Cuando el valor del peso asociado sea importante, tendremos el equivalente de una conexión fuerte, y cuando sea insignificante tendremos una conexión débil. Un peso de valor cero significa una conexión nula. Si se hubiera elegido a la tangente hiperbólica como función de activación esto podría representarse como:

Ecuación 7:

$$f(x) = \tanh(x). \qquad (7)$$

La elección de una función de activación adecuada es una elección "artesanal" del diseñador de la red que marca inconfundiblemente su diseño. Dos redes neuronales para el mismo propósito diferirán entre sí en la elección de sus funciones de activación, entre otras cosas. Esto les da una característica de "firma " o estilo propio del diseñador. Algunos usan una función con el objeto de normalizar los datos, como podría ser una función sigmoidal o la función seno. Otros aplican elaboradas funciones hechas a medida de acuerdo a lo que esperan obtener de la misma.

El siguiente pseudocódigo esquematiza el párrafo anterior:

Valores arreglo de salida = fx(valores arreglo de entrada) X valores arreglo pesos

fx = función de activación.

Cuando la red ha obtenido una salida, se da por concluido el proceso de propagación hacia adelante.

Paso 8. Recolectar el valor remanente de la capa de salidas.

Este valor será el resultado de la búsqueda de la red. El valor puede almacenarse en una variable de memoria o grabarse en un campo de una base de datos para su uso posterior. Algunos programas convierten el resultado a otro formato antes de realizar estas opciones dado que el resultado sin procesar generalmente es un número. Una tabla conteniendo estos números como un campo y un campo asociado con las descripciones correspondientes tendrían el efecto de poder retornar dicha descripción como resultado.

Paso 9. Asignación de acciones determinadas para la salida de la red.

Las acciones que puede tomar un programa en base a la salida de la red pueden ser:

Informativas:

El programa se limita a informar que ha encontrado una salida para la trama presentada como entrada.

Ejemplo : Red neuronal clasificadora de botellas:

La red se nutre de la imagen digitalizada de una cámara de vídeo conectada a la misma. La cámara toma la línea de producción en donde circulan diversos modelos de botellas, debiendo notificar cuando se detecta alguna botella distinta a los tipos estándar programados. La entrada de la red es entonces la trama de las botellas, y la salida es la identificación del modelo pertinente. Una acción informativa es activar un mensaje en pantalla que advierta cada vez que se detecta una botella extraña.

Operativas:

En este caso la computadora realiza algún procedimiento además de informar el suceso. Siguiendo el ejemplo anterior, la computadora ordena a un brazo robot que quite la botella extraña de la línea de producción. Alternativamente, la computadora podría abrir una puerta - trampa en el trayecto de la línea, que haga caer la botella hacia un canasto de desperdicios.

Predicativas:

En este caso la computadora no se limita a observar un fenómeno y describirlo sino a vaticinar la evolución probable de una variable.

Ejemplo: Una red alimentada con datos climáticos podría predecir el clima para los próximos días a partir de datos climáticos actuales.

Utilizando nuevamente el Pseudocódigo para ejemplificar estos pasos es posible tener una primera aproximación a una red neuronal.

Ejemplo de un programa de red neuronal:

PROGRAMA RED NEURONAL.

1. INICIAR PROGRAMA.
2. DIMENSIONAR MÚLTIPLES VECTORES (DE ACUERDO A LA ESTRUCTURA DE LA RED).
3. SI EXISTE ARCHIVO O BASE DE DATOS DE ENTRADA ENTONCES:
4. LEER ARCHIVO Y ASIGNAR VALORES A VECTORES DE ENTRADA.

5. EN CASO CONTRARIO:
6. TERMINAR PROGRAMA.
7. SI EXISTE BASE DE DATOS DE PESOS PARA LA CAPA DE ENTRADA ENTONCES:
8. LEER ARCHIVO Y ASIGNAR VALORES A VECTORES DE PESOS.
9. EN CASO CONTRARIO:
10. TERMINAR PROGRAMA.
11. SI EXISTE BASE DE DATOS DE SALIDAS ENTONCES:
12. LEER ARCHIVO Y ASIGNAR VALORES A VECTORES DE SALIDAS.
13. EN CASO CONTRARIO:
14. TERMINAR PROGRAMA.
15. ASIGNARLE A TODOS LOS ELEMENTOS VALOR NULO.
16. NOMBRAR AL PRIMER VECTOR CAPA DE ENTRADAS.
17. NOMBRAR AL ULTIMO VECTOR CAPA DE SALIDAS.
18. NOMBRAR A LOS VECTORES INTERMEDIOS COMO CAPAS OCULTAS.
19. HACER MIENTRAS EXISTAN CAPAS OCULTAS.
20. RECOGER VALOR DE CAPA PREVIA.
21. CALCULAR RESULTADO DE LA FUNCIÓN DE ACTIVACIÓN.
22. MULTIPLICAR EL RESULTADO POR LOS PESOS CORRESPONDIENTES.
23. TRANSFERIR VALOR A LA CAPA SIGUIENTE.
24. SI LA CAPA ES LA ULTIMA ENTONCES:
25. TRANSFERIR VALORES A LA CAPA DE SALIDAS.
26. VOLVER A EMPEZAR.

27. LEER VALORES DE LA CAPA DE SALIDA Y ASIGNARLOS A UNA VARIABLE.
28. BUSCAR EN BASE DE DATOS DE SALIDAS LA DESCRIPCIÓN O EL VALOR A RETORNAR POR LA RED.
29. PARA CADA VALOR POSIBLE DE SALIDA.
30. REALIZAR PROGRAMA ESPECIFICO PARA ESTA OPCIÓN.
31. TERMINAR PROGRAMA.

La figura 5.1 muestra el primer nivel de un Diagrama de Flujo de Datos de un sistema ejecutor de una red neuronal.

El dispositivo de captura de datos, al cual se hace referencia puede ser tanto un operador humano (data - entry) como un mecanismo automático, como:

· Acceso directo de INTERNET.

· E – mail.

· Cámara de video.

· Scanner.

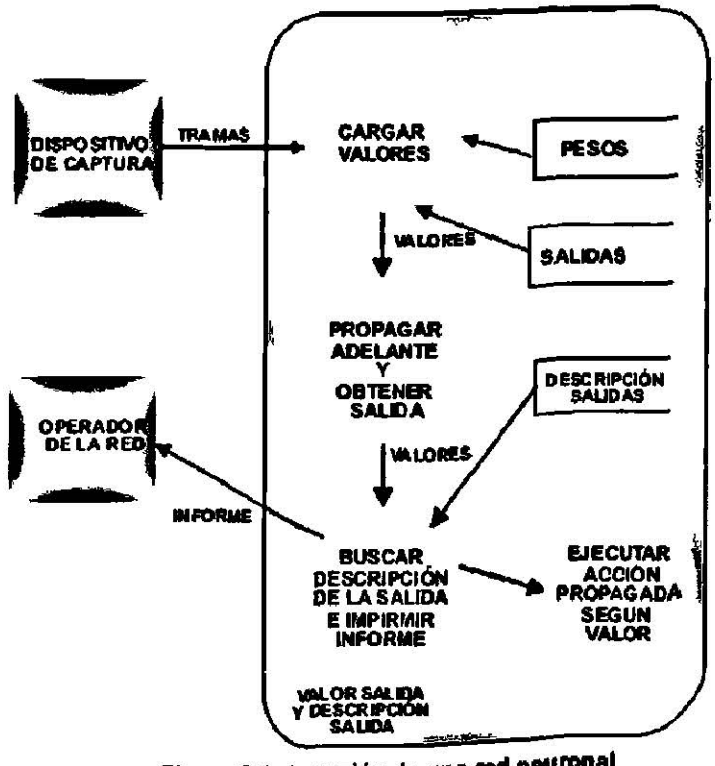


Figura 5.1 ejecución de una red neuronal

Por último hay que tener en cuenta que al implementar una red neuronal como parte de un programa o sistema informático, se pueden distinguir 3 fases básicas:

1. Diseño: En esta fase se elige el tipo de red neuronal a usar (la arquitectura o topología), el número de neuronas que la compondrán,...
2. Entrenamiento: En esta fase se le presentan a la red neuronal una serie de datos de entrada y datos de salida (resultados), para que a partir de ellos pueda aprender.
3. Uso: Se le suministran las entradas pertinentes a la red, y ésta genera las salidas en función de lo que ha aprendido en la fase de entrenamiento.

Capítulo VI

6. Arquitectura de una red neuronal.

La arquitectura de una red depende del tipo de red que se desee crear y dependiendo del tipo de la red se utilizará el mejor método de entrenamiento para la misma.

6.1. Funciones básicas de las redes neuronales.

Lo primero que tenemos que considerar acerca de las redes neuronales es su estructura, la forma en que se comunican las capas y las conexiones que puede tener.

Vamos a empezar con un modelo neuronal que se presenta y utiliza frecuentemente y a describir cada una de sus partes.

La arquitectura más usada en la actualidad de una red neuronal consistiría en:

Una primera capa de entradas, que recibe información proveniente de las fuentes externas de la red.

Una serie de capas ocultas encargadas de realizar el trabajo de la red. Estas son internas a la red y no tienen contacto directo con el entorno exterior. El número de niveles ocultos puede estar entre cero y un número elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas topologías de redes neuronales.

Una capa de salidas, que proporciona el resultado del trabajo de la red al exterior.

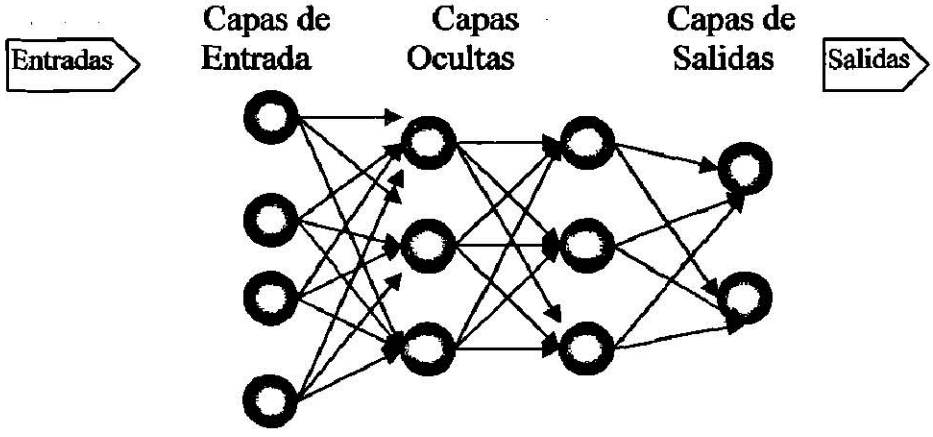


Figura 6.1Arquitectura red neuronal

Los círculos representan a las neuronas mientras que las flechas representan a las conexiones.

Un ejemplo de una neurona sobre la que convergen conexiones de diferente peso sináptico (W_i) seria el de la figura 6.2.

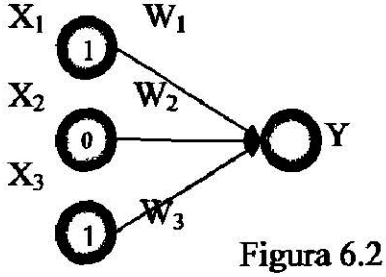


Figura 6.2

Conexiones de diferente peso
Sináptico ($w_1>w_2>w_3$) convergen
Sobre la misma neurona Y

Figura 6.2 Conexiones de diferente peso Sináptico convergen Sobre la misma neurona Y

El proceso de la información llevado acabo por la neurona Y, consiste en una función (F) que opera con los valores recibidos desde las neuronas de la capa anterior (X_i, generalmente 0 o 1), y que tiene en cuenta el peso sináptico de la conexión por la que se

recibieron dichos valores (W_i). Así, una neurona dará más importancia a la información que le llegue por una conexión de peso mayor que a aquella que le llegue por una conexión de menor peso sináptico.

Un modelo simple de la función sería:

$$F = X_1W_1 + X_2W_2 + ... + X_iW_i \qquad (?)$$

Si el resultado de la función F es mayor que el valor umbral (U), la neurona se activa y emite una señal (1) hacia las neuronas de la capa siguiente. Pero, si por el contrario, el resultado es menor que el valor umbral, la neurona permanece inactiva (0) y no envía ninguna señal.

$$X_1W_1 + X_2W_2 + ... + X_iW_i \leq U \leftrightarrow Inactivación \leftrightarrow Y = 0 \qquad (?)$$

$$X_1W_1 + X_2W_2 + ... + X_iW_i > U \leftrightarrow Activación \leftrightarrow Y = 1 \qquad (?)$$

De esta forma, definido un conjunto inicial de pesos en las conexiones, al presentar un "estímulo" (conjunto de ceros y unos que representa un dato, perfil u objeto) a la capa de entradas, cada neurona en cada capa realiza la operación descrita anteriormente, activándose o no, de manera que al final del proceso las neuronas de la capa de salidas generan un resultado (otro conjunto de ceros y unos), que puede coincidir o no con el que se desea asociar el estímulo.

6.2. Conexiones

Como pudimos apreciar en la figura anterior las flechas representan la conexiones y ésta a su vez son las que permiten que fluya la información a través de las neuronas. Cada neurona de la red es una unidad de procesamiento de información; es decir, recibe información a través de las conexiones con las neuronas de la capa anterior, procesa la información, y emite el resultado a través de sus conexiones con las neuronas de la capa siguiente, siempre y cuando dicho resultado supere un valor "*umbral*".

En una red neuronal la cual ha sido ya *entrenada*, las *conexiones* entre neuronas tienen un determinado peso, el cual se conoce como peso sináptico.

Además del número de capas que una red contenga y la manera en que se interconectan unas capas con otras se dividen en los siguiente tipos de conexiones.

Redes recurrentes (feed-back) la información puede volver a lugares por los que ya había pasado, formando ciclos, y se admiten las conexiones intracapa (laterales), incluso de una unidad consigo misma.

Redes no recurrentes o *redes en cascada* (feed-forward). La información fluye unidireccionalmente de una capa a otra (desde la capa de entrada a las capas ocultas y de éstas a la capa de salida), y además, no se admiten conexiones intracapa.

6.3. 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. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso de sí mismo (conexión auto-recurrente).

En la siguiente figura se muestran ejemplos de conexiones.

Conexiones hacia delante: para todos los modelos neuronales, los datos de las neuronas de una capa inferior son propagados hacia las neuronas de la capa superior por medio de las redes de conexiones hacia adelante.

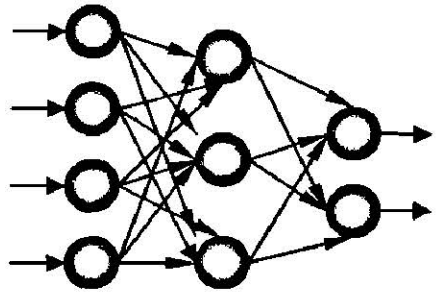


Figura 6.3

Figura 6.3 Conexiones hacia delante

Conexiones laterales: se admiten las conexiones intracapa.

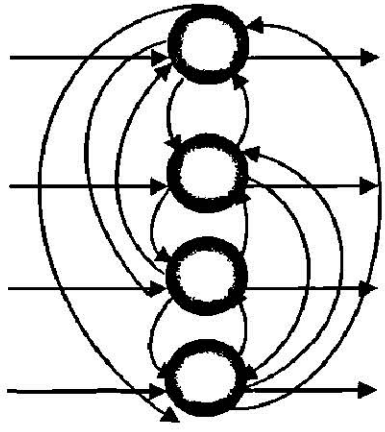


Figura 6.4

Figura 6.4 Conexiones laterales

Conexiones hacia atrás: Éstas conexiones llevan los datos de las neuronas de una capa superior a otras de la capa inferior.

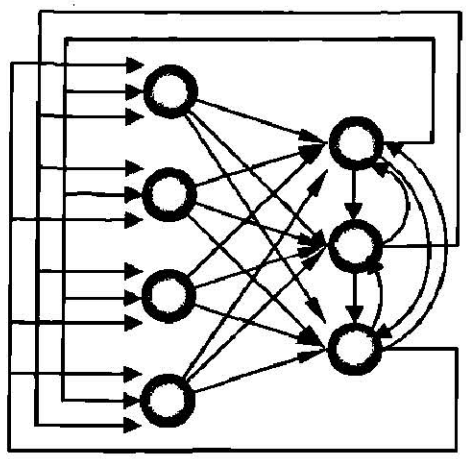


Figura 6.5

Figura 6.5 Conexiones hacia atrás (o recurrentes).

6.4. Estructuras de redes neuronales.

Para diseñar una red debemos establecer cómo estarán conectadas unas unidades con otras y determinar adecuadamente los pesos de las conexiones. Lo más usual es disponer las unidades en forma de capas, pudiéndose hablar de redes de una, de dos o de más de dos capas (redes multicapa). Aunque inicialmente se desarrollaron redes de una sola capa, lo más usual es disponer tres o más capas: la primera capa actúa como buffer de entrada, almacenando la información bruta suministrada a la red o realizando un sencillo pre-proceso de la misma, la llamamos capa de entrada.

Otra capa actúa como interfaz o buffer de salida, almacenando la respuesta de la red para que pueda ser leída, la llamamos capa de salida; y las capas intermedias, principales encargadas de extraer, procesar y memorizar la información, las denominamos capas ocultas.

Modelo de red en cascada de 3 capas:

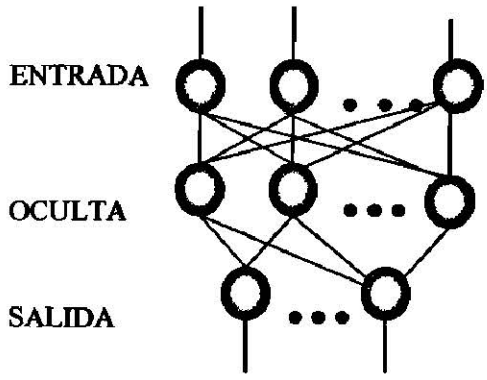


Figura 6.6

Figura 6.6 Modelo de red en cascada de 3 capas

6.5. Tipos de aprendizaje básicos de las redes neuronales.

La principal propiedad de una red neuronal artificial es la capacidad de aprender del entorno en el que opera y mejorar su funcionamiento. Dado que la noción de "aprendizaje" es muy genérica, se adopta la siguiente definición para expresar de forma más precisa, lo que se entiende por aprendizaje en el contexto de las redes neuronales artificiales: el aprendizaje es el proceso mediante el cual, los parámetros de una red neuronal artificial se adaptan como consecuencia de un proceso de estimulación llevado a cabo por el entorno en el que la red opera.

La definición anterior implica la siguiente secuencia de hechos:

- La red neuronal se encuentra estimulada por el entorno.
- La red neuronal cambia como consecuencia a dicho estimulo.

La red neuronal responde diferente al entorno a causa de los cambios que se han producido en su estructura interna.

El tipo de aprendizaje vendrá determinado por la forma en que cambie la configuración de la red.

Los tipos de aprendizaje pueden dividirse básicamente en tres, atendiendo a como esta guiado este aprendizaje:

1. Aprendizaje supervisado: En este tipo de aprendizaje se le proporciona a la Red Neuronal una serie de ejemplos consistentes en unos patrones de entrada, junto con la salida que debería dar la red. El proceso de entrenamiento consiste en el ajuste de los pesos para que la salida de la red sea lo más parecida posible a la salida deseada. Es por ello que en cada iteración se use alguna función que nos de cuenta del error o el grado de acierto que está cometiendo la red.

2. Aprendizaje de refuerzo: Se introducen valores de entrada, y lo único que se le indica a la red si las salidas que ha generado son correctas o incorrectas.

3. Aprendizaje no supervisado: No existe ningún tipo de guía. De esta manera lo único que puede hacer la red es reconocer patrones en los datos de entrada y crear categorías a partir de estos patrones. Así cuando se le entre algún dato, después del entrenamiento, la red será capaz de clasificarlo e indicará en qué categoría lo ha clasificado.

6.6. Algoritmos de aprendizaje de una RNA.

Estos algoritmos están formados por un conjunto de reglas que permiten a la red neuronal aprender (a partir de los datos que se le suministran), mediante la modificación de los pesos sinápticos de las conexiones entre las neuronas.

Existen muchos algoritmos de aprendizaje. Cada uno sirve para determinar redes neuronales. Entre los principales se tienen:

• **Aprendizaje por Corrección de Error**: Algoritmo muy conocido basado en la regla Delta, que busca minimizar la función de error usando un gradiente descendente (James A. Freeman, 1991).

• **Aprendizaje Competitivo**: En el cual dos neuronas de una capa compiten entre sí por el privilegio de permanecer activos, tal que una neurona con mayor actividad será la única que participará del proceso de aprendizaje. Es usado en mapas de Kohonen (Kohonen, 1988) y en redes ART (Gail A. Carpenter, 1992).

No hay referencia en bibliografía.

• **Aprendizaje Hebbiano**: Son dos neuronas que están simultáneamente activas a conexiones entre ellas que debe ser fortalecida caso contrario será debilitada (Hebb,1949) utilizada en el Modelo de Hopfield (Hopfield, 1982);

• **Aprendizaje de Boltzmann**: Es una regla de aprendizaje estocástico obtenido a partir de principios teóricos de información y de termodinámica. El objetivo de aprendizaje de Boltzmann es ajustar los pasos de conexión de tal forma que el estado de las unidades visibles satisfaga una distribución de probabilidades deseada en particular (D. Ackley, 1985).

No hay referencia en bibliografía

Capítulo VII

7. Redes con aprendizaje supervisado.

En esta parte se mostrará de una manera detallada el aprendizaje supervisado, la forma en que se va a desarrollar (¿Qué es lo que se va a desarrollar?) y los tipos que se derivan de la misma.

Este tipo de aprendizaje requiere tanto las entradas como las salidas esperadas de la red durante la fase entrenamiento. La red neuronal artificial procesa las entradas y compara los resultados obtenidos por las salidas reales con los valores esperados.

La diferencia entre ambos valores se propaga hacia capas intermedias haciendo que el sistema ajuste los pesos que definen el comportamiento de la red. Los datos utilizados en el proceso de aprendizaje se denominan conjunto de entrenamiento (training set). Durante el aprendizaje se procesan varias veces los mismos pares de vectores de entrenamiento y en cada paso del aprendizaje se modifican los pesos de las conexiones.

El tiempo de aprendizaje está determinado por la facilidad de la red para generar las salidas esperadas. El proceso de aprendizaje puede durar segundos o semanas, terminando cuando la red alcanza los resultados esperados. Las redes pueden no aprender, por ejemplo, en caso de que los vectores de entrada, utilizados para entrenar la misma, no contengan la información idónea a partir de la cual se deriven los vectores de salida.

Desde un punto de vista más general una red neuronal artificial puede entrenarse para que memorice un conjunto de datos. La memorización de datos es perjudicial ya que la

red deja de generalizar y empieza a reproducir. El entrenamiento excesivo es la causa de este fenómeno, si bien, puede evitarse este problema mediante la evaluación del comportamiento de la red sobre un conjunto de datos independiente o conjunto de validación (subconjunto del conjunto de datos inicial no utilizado en el entrenamiento y representativo de todo el conjunto) y así, poder actuar sobre el proceso de aprendizaje. De forma periódica, durante el entrenamiento se deben estudiar los resultados obtenidos con la red en el conjunto de validación. Para evitar la memorización de datos, el entrenamiento de la red debe concluir cuando sea capaz de producir buenos resultados en el conjunto de datos independientes. La memorización también se puede evitar utilizando un número adecuado de elementos de procesamiento (neuronas).

Si una red no puede resolver un problema, hay que revisar uno o varios de los siguientes puntos: los vectores de entrada y de salida, el número de capas, el número de neuronas en cada capa, las interconexiones entre capas, las funciones de activación utilizadas, etc.

La red neuronal artificial puede aprender inicialmente a reconocer las tendencias estadísticas de los datos utilizados en su entrenamiento, y posteriormente a extraer aspectos ocultos de los mismos. Una red entrenada se puede implementar en hardware de forma que su utilización sea más rápida. Existen redes neuronales artificiales que tienen la capacidad de aprender continuamente durante su utilización. Estas redes son capaces de absorber y adaptarse a los cambios en el medio en el que están integradas.

Las redes que requieren aprendizaje supervisado se utilizan en problemas de reconocimiento de códigos de barras, de predicción de series temporales, en problemas de mercadeo, de control, etc.

Este tipo de aprendizaje consiste en la modificación de los pesos de las conexiones en el sentido de reducir la discrepancia entre la salida obtenida y la deseada.

Los pasos generales a seguir por este algoritmo son los siguientes:

1. Inicializar con valores aleatorios (Aleatorizar) los pesos de todas las conexiones (de preferencia con valores pequeños).
2. Seleccionar un par de entrenamiento, es decir, un patrón de entrada y el patrón de salida deseado (target) correspondiente.
3. Presentar el patrón de entrada y calcular la salida de la red mediante las operaciones usuales: sumatoria de las entradas ponderadas, función de activación y transferencia a la siguiente capa, hasta llegar a la capa de salida (inicialmente se obtienen salidas aleatorias, ya que los pesos de las conexiones son aleatorios).
4. Cálculo del error o discrepancia entre la salida obtenida y la deseada. El error (función objetivo) se suele definir como la suma de los cuadrados de las diferencias entre las salidas reales obtenidas y las deseadas, promediado para todas las unidades de salida y todos los patrones de entrenamiento. Si el error es menor de cierto criterio fijado de antemano, incrementar el número de ejemplos correctos; si todos los ejemplos se han clasificado correctamente, finalizar, sino, continuar.
5. Aplicar la regla de aprendizaje, es decir, ajustar los pesos de las conexiones tratando de disminuir el error, generalmente mediante el cálculo de tasas de variación o gradientes del error, por eso hablamos de reglas de aprendizaje por gradiente descendiente. Para reducir el error habrá que modificar los pesos de las conexiones, en proporción a la tasa relativa de variación del error con respecto a la variación del peso, o sea, la derivada del error respecto al peso, EP (error respecto al peso). Una forma de calcular el EP sería perturbar levemente un

peso y observar cómo varía el error, pero no resultaría eficiente si trabajamos con muchas conexiones.

6. Volver al paso 2

De esta manera siguiendo los pasos anteriores se lleva acabo el aprendizaje supervisado.

En cuanto a los algoritmos de aprendizaje supervisado, en general se suelen considerar tres tipos, que dan lugar a los siguientes aprendizajes:

7.1 Aprendizaje por Corrección de error

El entrenamiento consiste en presentar al sistema un conjunto de pares de datos, representando la entrada y salida deseada para dicha entrada. Este conjunto recibe el nombre de conjunto de entrenamiento.

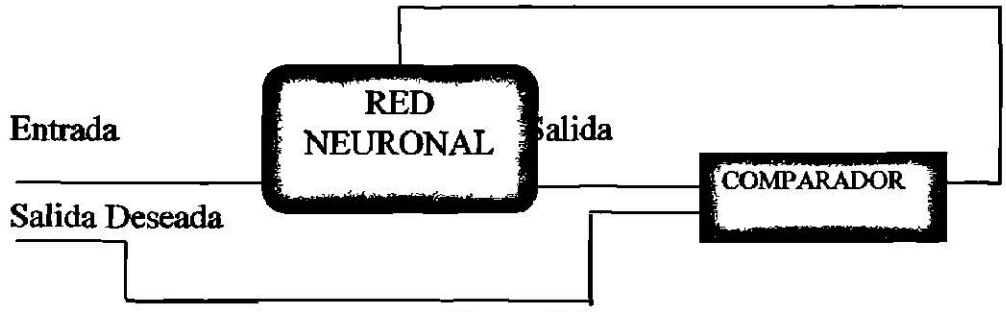


Figura 7.1

Figura 7.1 Representación del funcionamiento del aprendizaje por corrección de error

Objetivo: Minimizar el error entre la salida Deseada y la actual.

Aprendizaje Fuera de Línea (OFF line).

Cuando el aprendizaje es fuera de linea, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y un conjunto de datos de prueba, que serán utilizados en la

correspondiente fase. Además, los pesos de las conexiones permanecen fijos después 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.

Una generalización de la fórmula o regla para decir los cambios en los pesos es la siguiente:

Peso Nuevo = Peso Viejo + Cambio de Peso

Matemáticamente esto es:

wij(t+1) = wij(t) + .wij(t);

donde t hace referencia a la etapa de aprendizaje, wij(t+1) al peso nuevo y wij(t) al peso viejo.

Método:

- Inicializar aleatoriamente los pesos.
- Presentación del conjunto de entrenamiento (CE) .
- Obtención de la salida para el CE.
- Comparación de salidas deseadas actuales.
- Si se verifica el criterio de finalización ir al siguiente paso, si no ir al paso 2.
- Fin

Conjunto de entrenamiento: (X,T)

$x^i = (x1,.....,xn)$ $t^i = (t1,.....,tm)$

Salida Actual $u^i = (u1,.....,um)$

Minimizar el error $E = \sum_{i=1}^{m} (ti - ui)^2$

7.2. Aprendizaje por refuerzo.

Se trata de un aprendizaje supervisado, más lento que el aprendizaje por corrección de error. Consiste en ajustar los pesos de las conexiones, 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 que debe generar), como ocurría en el caso de supervisión por corrección del error.

7.3. Aprendizaje estocástico.

Este aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

En el aprendizaje estocástico se suele hacer una analogía en términos termodinámicos, asociando a la red neuronal con un sólido físico que tiene cierto estado energético. En el caso de la red, la energía de la misma representaría el grado de estabilidad de la red, 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 ajusta al objetivo 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 de Liapunov). 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 aceptaría el cambio en función de una determinada y preestablecida distribución de probabilidades. Lo dicho anteriormente se representa de una manera sencilla a continuación:

Simil: red Neuronal--------sólido físico(estados energéticos)

Estado mínimo de energía: Valores de pesos con los que la estructura se ajusta al objetivo deseado.

Proceso:

- Se realiza un cambio aleatorio de los pesos.
- Se determina la nueva energía de la red.
- Si la energía decrece: se acepta el cambio.
- Si no decrece: se acepta el cambio en función de una determinada y preestablecida distribución de probabilidades.

Maquina de Boltzman Cauchy

Ahora veremos algunos modelos de redes neuronales artificiales que utilizan este esquema de aprendizaje. Se pueden citar, por ejemplo:

RBF (Red con función de Base Radial).

Las redes neuronales RBF están constituidas por dos capas: una capa oculta y una capa de salida. La capa oculta esta formada por neuronas que aplican sobre sus entradas una función del tipo radial, es decir, la salida de cada neurona es una función de la distancia entra las entradas y un punto llamado centro, que caracteriza a cada neurona. Supondremos sin perder generalidad que la RNA tiene una sola salida. En dicho caso la capa de salida está constituida por una unidad que realiza una suma ponderada de las salidas de las neuronas de la capa oculta.

BAM (Memoria Asociativa Bidirectional).

La memoria asociativa bidireccional consta de dos capas de elementos de procesos que están completamente interconectados entre capas, pero no intracapas (no hay conexiones entre neuronas de la misma capa). Por ello se trata de una red recurrente. Las unidades pueden tener o no conexión consigo mismas. En la arquitectura redes BAM hay unos pesos asociados a las conexiones entre los elementos de procesos, a diferencia de otras arquitecturas estos pesos pueden ser determinados por anticipado si es posible identificar a todos los vectores de entrenamiento.

Capítulo VIII

8. Redes con aprendizaje no supervisado.

Estas redes se conocen también como aprendizaje autosupervisado, no requieren influencia externa para ajustar los pesos de las conexiones entre las 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.

El funcionamiento de estas redes se basa en la búsqueda de: Características, regularidades, correlaciones, categorías del conjunto de datos de entrada.

En este tipo de aprendizaje no se requiere presentar patrones de salida deseados. Sin embargo, el algoritmo y la regla de modificación de las conexiones producen vectores de salida consistentes, esto es, la presentación de un patrón aprendido o parecido a él, produce siempre la misma salida. Se considera que el proceso de aprendizaje es capaz de extraer ciertas propiedades estadísticas de los ejemplos de aprendizaje y agruparlos en categorías o clases de patrones similares. No sabemos *a priori* qué salida corresponderá a cada tipo o grupo de patrones de entrada, ni qué atributos usará para clasificarlos, por eso son muy útiles como generadores de categorías (clustering).

De manera general, los métodos de aprendizaje no supervisado apelan a alguna noción de la calidad de la representación. Las representaciones de calidad se caracterizan por la economía de la representación (el costo de codificación mas el costo de reconstrucción), sin perder la capacidad de reconstruir la información almacenada.

Existen varios métodos sencillos para encontrar códigos económicos que al mismo tiempo permitan una buena reconstrucción de la entrada: el aprendizaje por componentes principales, el aprendizaje competitivo y los códigos demográficos (Hinton, 1992).

En algunos casos, la salida representa el grado de familiaridad o similitud entre la información que se le está presentando en la entrada y las informaciones que se le han mostrado hasta entonces (en el pasado).

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 comparativo.**

8.1. Aprendizaje hebbiano.

Esta regla de aprendizaje es la base de muchas otras, la cual pretende medir la familiaridad o extraer características de los datos de entrada. El fundamento es una suposición bastante simple: si dos neuronas N_i y N_j toman el mismo estado simultáneamente (ambas activas o ambas inactivas), el peso de la conexión entre ambas se incrementa.

Las entradas y salidas permitidas a la neurona son: {-1, 1} o {0, 1} (neuronas binarias). Esto puede explicarse porque la regla de aprendizaje de Hebb se originó a partir de la neurona biológica clásica, que solamente puede tener dos estados: activa o inactiva.

8.2. Aprendizaje competitivo y comparativo.

Se orienta a la agrupación o clasificación de los datos de entrada. Como característica principal del aprendizaje competitivo se puede decir que, si un patrón nuevo se determina que pertenece a una clase reconocida previamente, entonces la inclusión de este nuevo patrón a esta clase matizará la representación de la misma. Si el patrón de entrada se determinó que no pertenece a ninguna de las clases reconocidas anteriormente, entonces la estructura y los pesos de la red neuronal serán ajustados para reconocer la nueva clase.

Existe una gran variedad de modelos que utilizan esta forma de aprendizaje, los más conocidos son:

8.3. SOM (Self-Organising Maps desarrollados por Kohonen).

Kohonen (1982) fue uno de los pioneros en este campo y el creador de las redes que llevan su nombre, también conocidas como SOM (Self-Organizing Maps).

Estas redes se caracterizan por su capacidad de clasificar conjuntos de datos de entrada. Poseen un solo nivel con muchas conexiones cuyos pesos necesitan inicializarse. Las neuronas compiten de forma que las triunfadoras son las únicas que modifican los pesos asociados a sus conexiones. Las redes creadas por Kohonen (1997) agrupan las neuronas en diferentes campos, de tal forma que la topología de la red define cómo proyectar un espacio de entrada en otro, sin cambiar la configuración relativa geométrica del espacio de entrada.

El aprendizaje es posible con la identificación de la estructura de los datos: figuras, correlaciones, agrupaciones, redundancia, etc. Estas redes se utilizan para interpretar datos, reducir dimensiones, preprocesar información, codificar, etc.

8.4. PCA (Principal Components Analysis).

El análisis de componentes principales, es una técnica matemática lineal que reduce la dimensionalidad de un conjunto de datos sin que se pierda demasiada información. Determina genes con características importantes sin buscar específicamente *a priori* aquellos con un patrón determinado. Es una técnica que se puede emplear en el análisis de la expresión génica en combinación con otras técnicas como K-medias o SOM.

8.5. Redes de Hopfield.

La red neuronal de Hopfield es una arquitectura formada por una sola capa que se utiliza principalmente como memoria autoasociativa, para almacenar y recuperar información. La información se almacena usando un método simple de aprendizaje no supervisado que obtiene la matriz de pesos que hace que dado cada uno de los patrones de entrenamiento (almacenamiento), la red devuelva el mismo patrón (recuperación). Posteriormente, cuando se tenga una configuración arbitraria de las neuronas como entradas, la red devolverá aquel patrón almacenado que esté más cerca de la configuración de partida en términos de la distancia de Hamming.

La distancia Hamming entre dos palabras se define como el número de bits que cambian entre una palabra código y la otra.

8.6. Learning Vector Quantization (LVQ).

LVQ es un método supervisado. Las neuronas de LVQ se actualizan de modo independiente. Es un algoritmo adaptable que encuentra un conjunto óptimo de vectores de referencia. Durante el aprendizaje, estos vectores de referencia son desplazados desde una distribución inicial aleatoria a una posición final que describe la función densidad de probabilidad del conjunto de vectores unidimensionales de entrenamiento.

8.7. ART (Adaptive Resonance Theory).

En este tipo de redes, los vectores de pesos del elemento de procesado seleccionado como ganador, sólo se actualizan con los patrones de entrada, se "resuenan" con éstos, es decir, si son "suficientemente similares". Si no son "suficientemente similares" se genera un nuevo elemento de proceso cuyos pesos son precisamente los del patrón de entrada utilizado. Evidentemente se hace necesario definir el concepto de "suficiente similaridad" que usualmente involucra un parámetro de vigilancia que lo controla.

Capítulo IX

9. Topología de redes neuronales.

La topología de la red define la forma de interconexión de las neuronas (o nodos).

En general, se pueden identificar tres tipos fundamentales de topologías de red:

9.1. Redes Monocapa.

Las redes monocapa se utilizan típicamente en tareas relacionadas con lo que se conoce como auto asociación, por ejemplo para regenerar informaciones de entrada que se presentan distorsionadas o incompletas.

Dentro de este tipo de redes se encuentran las siguientes subclases de redes.

9.2. Redes Monocapa con propagación hacia delante.

En las redes monocapa, como la red HOPFIELD y la red BRAIN-STATE-IN-A-BOX, se establecen conexiones laterales entre las neuronas que pertenecen a la única capa de la red. Además pueden existir conexiones auto recurrentes.

La forma más sencilla de organizar las neuronas de una red neuronal artificial en capas es la utilizada en las denominadas redes monocapa alimentadas hacia adelante (Single-layer Feedforward Networks).

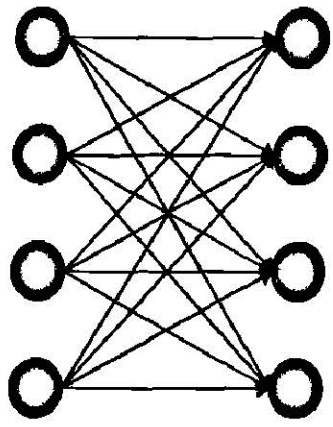


Figura 9.1

Capa de entrada Capa de Salida

Neuronas de Entrada Neuronas de salida

Figura 9.1 Red Monocapa

En esta topología, ilustrada en la Figura(9.1), se tiene una capa de entrada (formada por un conjunto de neuronas de entrada que se caracterizan por no realizar operación alguna) que es la encargada de capturar y distribuir las señales de entrada a las neuronas de la siguiente capa o capa de salida (formada por las neuronas de salida, encargadas de procesar la información y generar la salida de la red).

En este tipo de redes, se dice que la propagación de las señales a través de la red es siempre hacia delante o acíclica, en el sentido de ir de las neuronas de entrada hacia las neuronas de salida.

Es habitual que cuando se adopta una distribución en capas, la capa de entrada no se contabiliza como tal al no realizar operación alguna. De esta forma, también se denomina a este tipo de redes como redes monocapa.

9.3. Redes Multicapa.

Las redes multicapas son aquellas que disponen de un conjunto de neuronas agrupadas en más de 2 niveles o capas. En estos casos, una forma para distinguir la capa a la que pertenece una neurona, consistiría en fijarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida.

Dentro las redes Multicapa se distinguen dos tipos: conexiones hacia adelante o feedforward y conexiones hacia atrás o feedback.

9.4. Redes multicapa con conexiones hacia delante.

Normalmente, todas las neuronas de una capa reciben señales de entrada desde otra capa anterior (la cual está más cerca a la entrada de la red), y envían señales de salida a una capa posterior (que está más cerca a la salida de la red).

Las neuronas de estas capas intermedias se denominan neuronas ocultas. De esta forma, la capa de entrada actúa como una capa sensorial que capta las señales de entrada externa y las propaga hacia la primera capa intermedia. Las salidas de esta primera capa intermedia se distribuyen sobre la siguiente capa intermedia y así sucesivamente, hasta llegar a la capa de salida que proporciona la respuesta de la red ante el estímulo externo.

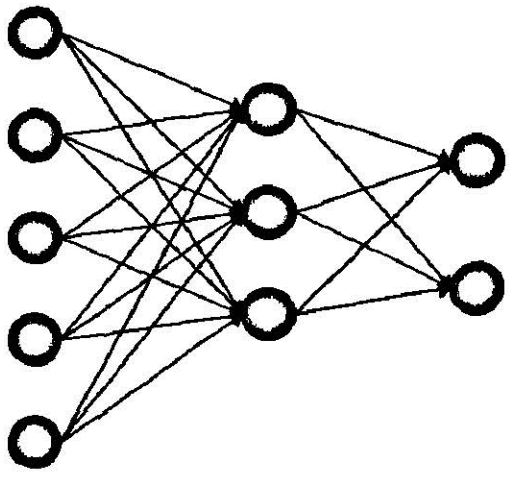


Figura 9.2

Figura 9.2 Red multicapa con propagación hacia delante.

La figura muestra una red multicapa que se referencia como red 5-3-2 (por tener 5 neuronas de entrada, 3 ocultas y 2 de salida). Además, se dice que la red está completamente conectada ya que todas las neuronas de cada una de las capas están conectadas con todas las neuronas de la capa siguiente.

De acuerdo con el Teorema de Aproximación Universal (Hornik et al., 1989; Funahashi, 1989; Cybenko, 1989; Hartman et al., 1990), el cual establece que una sola capa intermedia es suficiente para aproximar, con una precisión arbitraria, cualquier función con un número finito de discontinuidades, siempre y cuando, las funciones de activación de las neuronas ocultas sean no lineales.

La estructuración en capas, incrementa notablemente el poder representativo de las RNA (o capacidad de la red de modelar una función específica). Esta afirmación es basada en el teorema anterior.

Este teorema establece que las redes multicapa no añaden capacidad computacional a menos que la función de activación de las capas sea no lineal. La demostración intuitiva es sencilla. Para ello, se denota por W1 a la matriz de pesos de las interconexiones entre la capa de entrada y la primera capa intermedia, donde la fila k-ésima de dicha matriz se corresponde con el vector de pesos (w_k) asociado a la k-ésima neurona oculta. De esta forma, si x es el vector de entrada a la red y no hay funciones de activación no lineales, se tiene que la salida de la primera capa intermedia viene dada por el producto W1x. Para la salida de la segunda capa se tiene que es W2(W1x), y así sucesivamente. Como el producto de matrices es asociativo, la expresión anterior es equivalente a (W1W2)x y se puede concluir, por tanto, que una red bicapa sería equivalente a una red monocapa con una matriz de pesos igual a la matriz resultado de W1W2.

De especial relevancia histórica cabe mencionar el resultado obtenido por Minsky y Paper en 1969. En dicho resultado se mostraba que un perceptrón monocapa (hasta entonces el máximo exponente de las RNA) era incapaz de representar la disyunción exclusiva o XOR a menos que se añadiera una neurona oculta. Este simple ejemplo, puso de manifiesto que las redes multicapa incrementaban la clase de problemas que las redes neuronales artificiales podían solucionar. Por otro lado, planteó el problema de cómo determinar los valores adecuados para los pesos durante el proceso de

entrenamiento, cuestión no resuelta hasta la introducción del algoritmo de retropropagación (Rumelhart et al., 1986).

Las redes feedforward más conocidas son: PERCEPTRON, ADALINE, MADALINE, LINEAR ADAPTATIVE MEMORY, DRIVE-REINFORCEMENT, BACKPROPAGATION. Todas ellas son útiles en aplicaciones de reconocimiento o clasificación de patrones.

9.5 Redes multicapa con conexiones hacia atrás.

Sin embargo, en un gran número de estas redes también existe la posibilidad de conectar la salida de las neuronas de capas posteriores a la entrada de 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.

9.6. Redes recurrentes.

En la década de los 80's con el fin de estudiar procesos que involucran sistemas gobernados por ecuaciones diferenciales no lineales surge la teoría clásica de control geométrico basada en la geometría diferencial, simultáneamente renace el estudio de la redes neuronales debido al redescubrimiento del algoritmo de retropropagación, este hecho sumado al fracaso de las metodologías tradicionales aplicadas a la inteligencia

artificial y a la disponibilidad de herramientas computacionales de bajo costo, permitieron el desarrollo de las redes neuronales recurrentes cuya principal aplicación es el control e identificación de los sistemas no lineales. Este desarrollo es posible debido a que las propiedades matemáticas de las redes recurrentes están enmarcadas en las mismas propiedades que fundamentan el control geométrico. La primera red neuronal recurrente de naturaleza dinámica fue propuesta por Hopfield en 1984 bajo el contexto de memorias asociativas.

Las redes recurrentes (Recurrent Networks) se diferencian de las redes con propagación hacia delante, en el hecho de presentar al menos un ciclo o un camino cerrado dentro de la red. Por lo tanto, las señales ahora no sólo se propagan desde la entrada hacia la salida, sino que existe una realimentación de forma que la salida de una neurona en un determinado camino, es a su vez entrada para otra neurona de procesamiento anterior del mismo camino.

El ciclo más sencillo posible consiste en una autorealimentación, de forma que la salida de una neurona se utiliza simultáneamente como entrada a la misma.

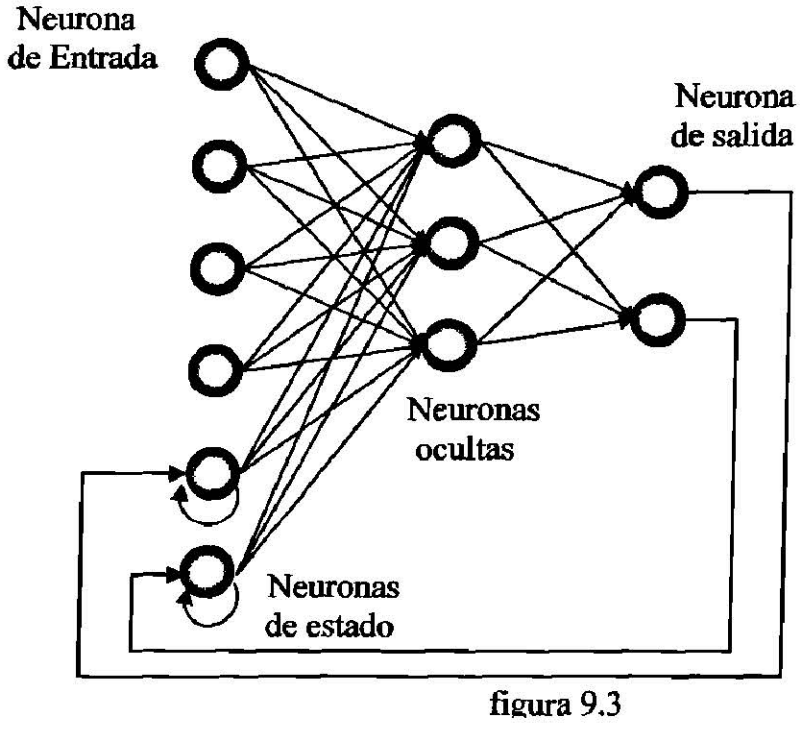


Figura 9.3 Figura Red recurrente de Jordán.

La Figura muestra un ejemplo de red recurrente, denominada red de Jordan (1986).

A diferencia de las redes con propagación hacia adelante, en este tipo de redes las propiedades dinámicas son importantes. En algunos casos, la salida de la red sufre un proceso de relajación de modo que la misma evoluciona hacia un estado estable en el que las neuronas no cambian su salida, mientras que en otros casos no está garantizada dicha estabilidad.

9.7. La Red Hopfield.

La Red Hopfield es recurrente y completamente interconectada. Funciona como una memoria asociativa no lineal, que puede almacenar internamente patrones presentados de forma incompleta o con ruido. De esta forma puede ser usada como una herramienta de optimización; también se han utilizado en aplicaciones de segmentación y restauración de imágenes y optimización combinatoria.

La Red Hopfield consta de un número de neuronas simétrica e íntegramente conectadas, como ya se mencionó anteriormente. Esto significa que si existe una conexión desde la neurona N_i a la neurona N_j, también existe la conexión desde N_j a N_i; ambas exibiendo el mismo peso ($w_{ij} = w_{ji}$). Vale aclarar que la conexión de una neurona con sí misma no está permitida.

El conjunto permitido de valores de entrada y salida es {0, 1} (o en alguna oportunidad {-1, 1}); o sea, es un conjunto binario. De esta manera todas las neuronas en una Red Hopfield son binarias, tomando solamente uno de los dos estados posibles: activo (1) o inactivo (-1 o 0).

Las Redes Hopfield se emplean para reconocer patrones. Después que el aprendizaje haya llegado a su fin, la red neuronal debe ser capaz de dar una salida correcta para cada patrón de entrada dado, aun cuando éste sea ruidoso. La clave del aprendizaje Hopfield es que si un patrón que tiene que ser aprendido se conoce, los pesos sobre cada conexión de la red neuronal pueden ser calculados. En esta circunstancia, solamente el estado de las neuronas cambia durante el proceso de aprendizaje. Este cálculo garantiza que cada patrón aprendido corresponda a un mínimo de la función energía.

Es importante entender que para este tipo de redes la definición de aprendizaje

es diferente al dado anteriormente, donde aprendizaje significaba simplemente la adaptación de los pesos. En una Red Hopfield los pesos se pueden calcular y se mantienen fijos durante el aprendizaje de los patrones. Solamente cambia el estado de las neuronas.

Para calcular el peso de una conexión cualquiera, w_{ij} (y por simetría para la conexión w_{ji}), en una Red Hopfield se utiliza la siguiente ecuación: siendo Q el número de patrones y eq_i la entrada a la neurona N_i.

Generalmente es aconsejable trabajar con esta ecuación cuando los patrones que se han de aprender no son muy semejantes unos a otros, y si el número de ceros y unos son similares para todos los patrones. Con respecto al número de ceros y unos, el umbral de cada neurona puede utilizarse para regular esto, distinguiéndose así dos casos posibles:

a) Si hay más 0's que 1's el umbral tiene que disminuirse, porque las neuronas tienen una probabilidad más alta para hacerse inactivas que para hacerse activas.

b) Si hay mas 1's que 0's el umbral tiene que incrementarse, porque las neuronas tienen una probabilidad más alta para hacerse activas que para hacerse inactivas.

Existen otros tipos de redes neuronales artificiales con no menos importancia, las cuales veremos a continuación:

9.8. Redes heteroasociativas.

Las redes heteroasociativas, al asociar informaciones de entrada con diferentes informaciones de salida, precisan al menos de dos capas, una para captar y retener la información de entrada y otra para mantener la salida con la información asociada. Si

esto no fuese así, se perdería la información inicial al obtenerse el dato asociado {3}, lo cual no debe ocurrir, ya que en el proceso de obtención de la salida se puede necesitar acceder varias veces a esta información que, por tanto, deberá permanecer en la capa de entrada.

En cuanto a su conectividad, pueden ser del tipo con conexión hacia adelante (o feedforward) o con conexión hacia atrás (feddforward/feedback), o bien con conexiones laterales.

9.9. Redes autoasociativas.

Una red autoasociativa asocia una información de entrada con el ejemplar más parecido de los almacenados conocidos por la red. Estos tipos de redes pueden implementarse con una sola capa de neuronas. Esta capa comenzará reteniendo la información inicial a la entrada, y terminará representando la información autoasociada. Si se quiere mantener la información de entrada y salida, se deberían añadir capas adicionales, sin embargo, la funcionalidad de la red puede conseguirse en una sola capa. En cuanto a su conectividad, existen de conexiones laterales y, en algunos casos, conexiones autorrecurrentes.

9.10.Redes de prealimentación o alimentación.

En este tipo de redes las conexiones son unidireccionales y no existen ciclos.

Capítulo X

10. Perceptrón Multicapa

10.1. Perceptrón

Empezaremos por analizar el modelo del Perceptrón cuya función es que intenta modelar el comportamiento de la neurona biológica. También hay que reconocer que el Perceptrón es la base de la arquitectura de las redes neuronales artificiales. Veremos los diferentes tipos que hay, nosotros centraremos nuestra atención en el Perceptrón multicapa el cual surge como una necesidad de las limitaciones del Perceptrón.

La red tipo Perceptrón fue inventada por el psicólogo Frank Rosemblatt en el año de 1957. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general sin entrar en mayores detalles con respecto a condiciones específicas y desconocidas por organismos biológicos concretos.

La primera red neuronal conocida fue desarrollada en 1943 por Warren McCulloch y Walter Pitts; ésta consistía en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la red. Si en la comparación, la suma de las entradas multiplicadas por los pesos es mayor o igual que el patrón preestablecido. la salida de la red es uno(1) en caso contrario es cero(0). Al inicio del desarrollo de los sistemas de inteligencia artificial, se encontró gran similitud entre su comportamiento y

el de los sistemas biológicos y en principio se creyó que este modelo podía computar cualquier función aritmética o lógica.

10.2. Funcionamiento

En la siguiente figura se representa una neurona "artificial", que intenta modelar el comportamiento de la neurona biológica. Aquí el cuerpo de la neurona se representa como un sumador lineal de los estímulos externos z_j, seguida de una función no lineal $y_j = f(z_j)$. La función $f(z_j)$ es llamada la función de activación, y es la función que utiliza la suma de estímulos para determinar la actividad de salida de la neurona.

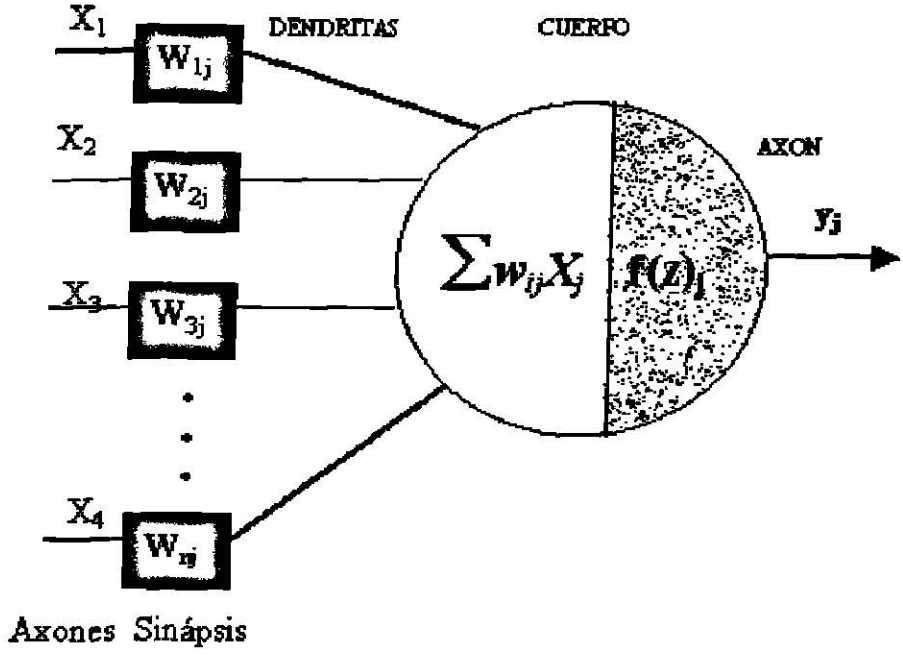


Figura 10.1Perceptrón de McCulloch-Pitts

Este modelo se conoce como Perceptrón de McCulloch-Pitts, y es la base de la mayor parte de la arquitectura de las redes neuronales artificiales que se interconectan entre sí. Las neuronas emplean funciones de activación diferentes según la aplicación, algunas veces son funciones lineales, otras funciones sigmoidales y otras funciones de umbral de

disparo. La eficiencia sináptica se representa por factores de peso de interconexión w_{ij}, desde la neurona i, hasta la neurona j.

Los pesos pueden ser positivos (excitación) o negativos (inhibición). Los pesos junto con las funciones $f(z)$ dictan la operación de la red neuronal. Normalmente las funciones no se modifican de tal forma que el estado de la red neuronal depende del valor de los factores de peso (sinápsis) que se aplica a los estímulos de la neurona.

En un Perceptrón, cada entrada es multiplicada por el peso W correspondiente, y los resultados son sumados, siendo evaluados contra el valor de umbral, si el resultado es mayor al mismo, el Perceptrón se activa.

Limitantes.

El Perceptrón es capaz tan sólo de resolver funciones definidas por un hiperplano (objeto de dimensión N-1 contenida en un espacio de dimensión N), que corte un espacio de dimensión N.

Una explicación más sencilla de un hiperplano sería, hablando en un plano de dos dimensiones, una línea que separa a los elementos existentes en dos grupos. El Perceptrón sólo puede resolver una función, si todos los posibles resultados del problema pueden separarse de ésta forma (en dos secciones) es decir, que no se combinen entre sí.

10.3. Tipos de Perceptrón

Perceptrón Unicapa.

Un Perceptrón unicapa no es más que un conjunto de neuronas no unidas entre sí, de manera que cada una de las entradas del sistema se conectan a cada neurona, produciendo cada una de ellas su salida individual:

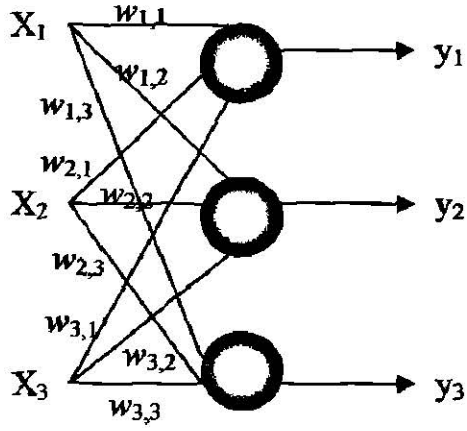


Figura 10.2 Perceptrón unicapa

El Perceptrón básico de dos capas sólo pude establecer dos regiones separadas por una frontera lineal en el espacio de patrones de entrada, donde se tendría un hiperplano.

Un Perceptrón con tres niveles de neuronas puede formar cualquier región convexa en este espacio. Las regiones convexas se forman mediante la intelección (Intelección. Acción y efecto de entender) entre las regiones formadas por cada neurona de la segunda capa, cada uno de estos elementos se comporta como un Perceptrón simple, activándose su salida para los patrones de un lado del hiperplano.

Un Perceptrón con cuatro capas puede generar regiones de decisión arbitrariamente complejas. Puede resolver una gran variedad de problemas cuyas entradas sean analógicas; la salida sea digital y sea linealmente separable.

Hemos visto que las neuronas necesitan de algún tipo de entrenamiento para la resolución de problemas. Para el Perceptrón existen tres tipos de entrenamiento los cuales se mencionan a continuación: **Supervisado, Por Refuerzo y No Supervisado.**

Aprendizaje Supervisado. Se presentan al Perceptrón unas entradas con las correspondientes salidas que queremos que aprenda. De esta manera la red primeramente calcula la salida que da ella para esas entradas y luego, conociendo el error que está cometiendo, ajusta sus pesos proporcionalmente al error que ha cometido (si la diferencia entre salida calculada y salida deseada es nula, no se varían los pesos).

Aprendizaje No Supervisado, sólo se presentan al Perceptrón las entradas y, para esas entradas, la red debe dar una salida parecida.

Aprendizaje Por Refuerzo se combinan los dos anteriores, y de cuando en cuando se presenta a la red una valoración global de como lo está haciendo.

10.4. Aplicaciones del Perceptrón

El Perceptrón solo es el ejemplo más elemental de una red neuronal, de hecho, no puede siquiera ser considerado una "red", puesto que no intervienen otros elementos. Si se combinan varios perceptrones en una "capa", y los estímulos de entrada después se suman tendremos ya una red neuronal.

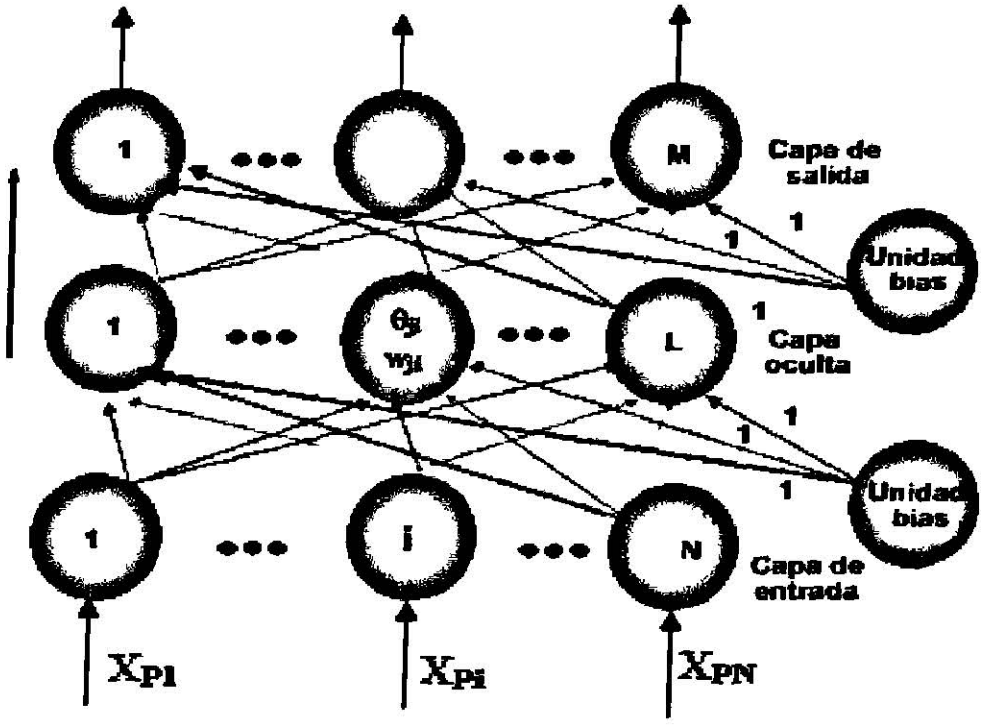


Figura 10.3 Ejemplo esquemático de la arquitectura de los perceptrones

En esta red, se interconectan varias unidades de procesamiento en capas, las neuronas de cada capa no se interconectan entre sí. Sin embargo, cada neurona de una capa proporciona una entrada a cada una de las neuronas de la siguiente capa, esto es, cada

neurona transmitirá su señal de salida a cada neurona de la capa siguiente. La figura muestra un ejemplo esquemático de la arquitectura de este tipo de redes neuronales.

10.5 .El Perceptrón multicapa o MLP (Multi-Layer Perceptron)

Esta estructura nació con la intención de dar solución a las limitaciones del Perceptrón clásico o unicapa, y supuso el resurgimiento del movimiento conexionista. Como su nombre lo indica, se trata de unos cuantos (dos o tres) perceptrones unicapa conectados en cascada, como en la siguiente figura:

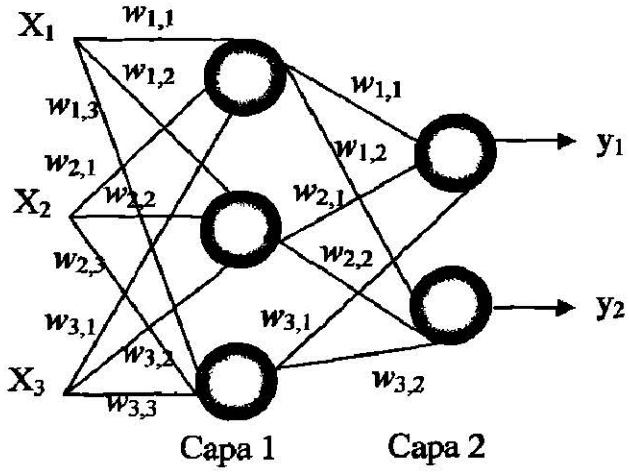


Figura 10.4 Perceptrón multicapa

El Perceptrón multicapa es una red con alimentación hacia delante, compuesta de varias capas de neuronas entre la entrada y la salida de la misma, esta red permite establecer regiones de decisión mucho mas complejas que la de dos semiplanos, como lo hace el Perceptrón de un solo nivel.

La popularidad de la arquitectura MLP se debe al hecho de que un MLP con una única capa oculta puede aproximar cualquier función continua en un intervalo hasta el nivel deseado, cuestión demostrada por Funahaski (1989) y que proporciona una base sólida al campo de las redes neuronales, aunque el resultado no informa sobre el número de nodos ocultos necesarios para llevar a cabo la aproximación.

Para poder hacer aprender cosas a un Perceptrón de este tipo, se implementó el algoritmo de BackPropagation, que tal como su nombre lo indica tiene la función de ir propagando los errores producidos en la capa de salida hacia atrás.

El algoritmo de aprendizaje *backpropagation* (Rumelhart, Hinton y Williams, 1986) aplicado a un perceptrón multicapa se basa en el cálculo de la derivada parcial del error con respecto a los pesos para averiguar qué dirección tomar para modificar los pesos con el fin de reducir el error de forma iterativa.

En el siguiente capítulo se menciona ampliamente cómo se lleva a cabo el entrenamiento de este tipo de redes.

Capítulo XI

11. Retropropagación (Backpropagation).

La retropropagación consiste en propagar el error hacia atrás, es decir, de la capa de salida hacia la capa de entrada, pasando por las capas ocultas intermedias y ajustando los pesos de las conexiones con el fin de reducir dicho error. Hay distintas versiones o reglas del algoritmo de retropropagación y distintas arquitecturas conexionistas a las que puede ser aplicado.

Al principio no se contaba con algoritmos de entrenamiento para redes multicapa, las redes de una sola capa estaban limitadas en su capacidad de representación y por un tiempo las redes neuronales artificiales se quedaron estancadas. La invención y perfeccionamiento del algoritmo de retropropagación dió un gran impulso al desarrollo de este campo. Tiene un buen fundamento matemático y a pesar de sus limitaciones ha expandido enormemente el rango de problemas donde se aplican las redes neuronales artificiales.

11.1. Origen

El algoritmo fue ideado a principios de los 70 por Werbos, y redescubierto a principios de los 80's por Parker y Rumelhart independientemente, sin embargo, no se hizo popular hasta 1986, cuando Rumerlhart, Hinton y Williams presentaron una descripción clara y concisa del mismo.

Desde la fecha clave de 1986 han surgido nuevas versiones que han tratado de aumentar la velocidad de convergencia del algoritmo y han tratado de superar algunos de sus inconvenientes, como la tendencia a alcanzar mínimos locales y no globales.

Empezaremos por ver cómo se representa la retropropagación, el algoritmo de entrenamiento en el cual se basa y cómo se va desarrollando cada paso para su funcionamiento.

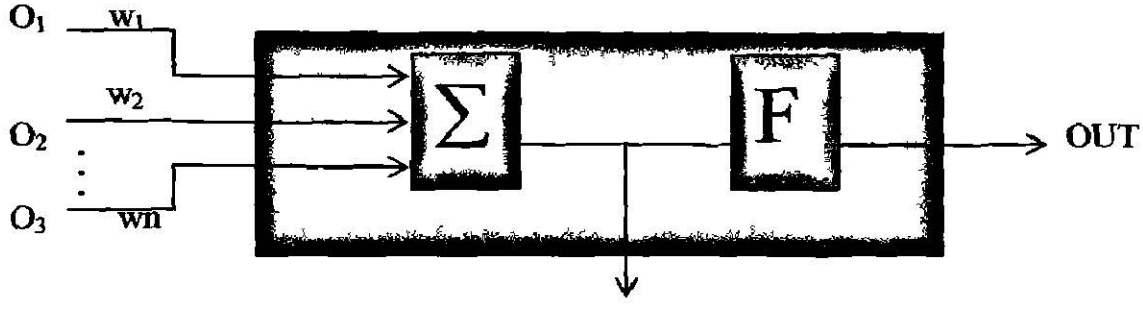
Tomaremos una definición acerca de la retropropagación ya que la mayoría coinciden en lo mismo.

Definición. El funcionamiento de la red de retropropagación (BPN por su siglas en inglés) consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo: Primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, éstos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo de error aproximado a la neurona intermedia a la salida original. Este

proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida sea la más cercana a la deseada.

La siguiente figura muestra la neurona usada como el bloque fundamental para las redes del retropropagación. Un juego de entrada es aplicado, ya sea del exterior o de una capa anterior. Cada uno de éstos es multiplicado por un peso, y los productos se suman. Esta suma de productos es él termino NET y debe calcularse para cada neurona en la red. Después de que NET es calculado, una función de activación F se aplica para modificarlo, lo cual produce la señal salida.

Muestra la función de activación normalmente usada para la backpropagation.



$$NET = o_1 w_1 + o_2 w_2 + \ldots + o_n w_n = \sum_{i=1}^{n} o_i w_i \qquad OUT = F(NET)$$

Figura 11.1 Neurona artificial con función de activación

Dentro de los avances logrados con la retropropagación es que esta red aprovecha la naturaleza paralela de las redes neuronales para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre patrones dados. Además el tiempo de desarrollo de cualquier sistema que sé este tratando de analizar se puede reducir como consecuencia de que la red puede aprender el algoritmo correcto sin que alguien tenga que deducir por anticipado el algoritmo en cuestión.

La importancia de la red radica en que tiene la capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones de entrada y sus salidas correspondientes. Es importante la capacidad de generalización, facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le dan entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas.

Analizando la retropropagación (propagación del error hacia atrás), basado en la generalización de la regla delta.

11.1 La Regla Delta Generalizada

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

Este algoritmo utiliza una función de error asociada a la red, buscando el estado estable de mínima energía o de mínimo error a través del camino descendente de la superficie del error. Por 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.

11.2. Funcionamiento del algoritmo

El método que sigue la regla delta generalizada para ajustar los pesos es exactamente el mismo que el de la regla utilizada en ADALINE; los pesos se actualizan de forma proporcional a la delta, o diferencia entre la salida deseada y la obtenida (δ = salida deseada – salida obtenida).

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

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

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

En redes multinivel en principio no se puede conocer la salida deseada de las neuronas ocultas para poder determinar los pesos en la función de error cometido. Inicialmente podemos conocer la salida deseada de las neuronas de salida. Según esto, si consideramos la unidad U_j de salida, entonces definimos:

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

Donde δ_{pj} es la salida deseada de la neurona j para el patrón p y net_j es la entrada neta que recibe la neurona j.

Esta fórmula es como la de la regla delta, excepto a los se refiere a la derivada de la función de transferencia. Este término representa la modificación que hay que realizar en la entrada que recibe la neurona j. En caso de que dicha neurona no sea de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona. Esto es lo que se denomina como procedimiento de propagación de error hacia atrás.

Según esto, en el caso de que U_j no sea una neurona de salida, el error que se produce está en función del error que se comete en las neuronas que reciben como entrada la salida de U_j:

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

Donde el rango de ***k*** cubre a todas las neuronas a las que está conectada la salida de U_j. De esta forma el error que se produce en una neurona oculta es la suma de los errores que se producen en las neuronas a las que está conectada la salida de ésta, multiplicado cada uno de ellos por el peso de la conexión.

11.4. Adición de un momento en la regla delta generalizada.

El método de retropopagación del error requiere un importante número de cálculos para lograr el ajuste de los pesos de la red. En la implementación del algoritmo, se toma una amplitud de pasos que vienen dada por la tasa de aprendizaje. A mayor tasa de aprendizaje, 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. Rumelhart, Hinton y Williams sugirieron que para filtrar estas oscilaciones se añada en la expresión de incremento de los pesos un término (momento), β, de manera que dicha expresión quede:

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

Donde β es una constante que determina el efecto $t+1$ 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 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 significa que se ha pasado por un mínimo y que los pesos deben ser menores para poder alcanzarlo.

Resumiendo, el algoritmo Backpropagation queda finalmente:

$$w_{ji}(t+1) = w_{ji}(t) + [\Delta w_{ji}(t+1)]$$

$$w_{ji}(t+1) = w_{ji}(t) + [\alpha \delta_{pj} y_{pi} + \beta(w_{ji}(t)]$$

donde:

$$\delta_{pj} = (d_{pj} - y_{pj}) \cdot f'(net_j)$$ Si U_j es una neurona de salida.

$$\delta_{pj} = (\sum_k \delta_{pk} w_{kj}) \cdot f'(net_j)$$ Si U_j no es una neurona de salida.

11.5. Estructura y aprendizaje de la red de retropropagación.

En una red de retropropagación 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 (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás feedback ni laterales entre las neuronas de la misma capa.

La aplicación del algoritmo tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida de la red, se inicia la segunda fase, comparándose éstos valores con la salida esperada para obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropropagación del error, ajustando los pesos y continuando con este proceso hasta llegar a la primer capa. De esta manera se han modificado los pesos de las

conexiones de la red para cada patrón de aprendizaje del problema, del que conocíamos su valor de entrada y la salida deseada que debería generar la red ante dicho patrón.

La técnica de retropropagación requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal.

11.6. Pasos para aplicar el algoritmo de entrenamiento.

Paso 1

Inicializar los pesos de la red con valores pequeños aleatorios.

Paso 2

Presentar un patrón de entrada y especificar la salida deseada que debe generar la red.

Paso 3

Calcular la salida actual de la red. Para ello presentamos las entradas 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. 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:

$$net_{pj}^{h} = \sum_{i-1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h}$$

En donde el índice ***h*** se refiere a magnitudes de la capa oculta; el subíndice ***p***, al *p*-ésimo vector de entrenamiento, y ***j*** a la *j*-ésima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

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:

$$net_{pk}^o = \sum_{j-1}^{L} w_{kj}^o y_{pj} + \theta_k^o$$

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

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 la delta es:

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

La función f debe ser derivable. En general disponemos de dos formas de función de salida:

La función lineal: $f_k(net_{jk}) = net_{jk}$

La función sigmoidal: $f_k(net_{jk}) = \dfrac{1}{1 + e^{-net_{jk}}}$

La selección de la función depende de la forma que se decida representar la salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, en otros casos, la lineal.

Para una 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 los términos de error para las neuronas de salida quedan:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) \text{ Para la salida lineal.}$$

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) y_{pk} (1 - y_{pk}) \text{ Para la salida sigmoidal.}$$

Si la neurona j no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente, por tanto se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados.

La expresión obtenida en este caso es:

$$\delta_{pj}^{h} = f_j^{h\prime}(net_{pj}^{h}) \sum_k \delta_{pk}^{o} w_{kj}^{o}$$

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 propagación hacia atrás.

Paso 5.

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

Para los pesos de las neuronas de la capa de salida:

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

$$\Delta w_{kj}^{o}(t+1) = \alpha \delta_{pk}^{o} y_{pj}$$

Para los pesos de las neuronas de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \Delta w_{ji}^{h}(t+1)]$$

$$\Delta w_{ji}^{h}(t+1) = \alpha \delta_{pj}^{h} x_{pi}$$

En ambos casos, para acelerar el proceso de aprendizaje se puede añadir un término momento.

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.

11.7. Consideraciones sobre el algoritmo de aprendizaje

El algoritmo encuentra un valor mínimo de error (local o global) mediante una aplicación de pasos (gradiente) descendentes. Cada punto de la superficie de la función 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.

Uno de los problemas del algoritmo es que en busca de minimizar la función de error, puede caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función de error. Sin embargo, no tiene porqué

alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

11.8. Aplicaciones de la red de Retropropagación.

Actualmente, este tipo de redes se están aplicando en distintas clases de problemas debido al la naturaleza general del proceso de aprendizaje. Algunos de los campos generales de aplicación son:

Codificación de Información.

Traducción de texto a lenguaje hablado.

Reconocimiento de lenguaje hablado.

Reconocimiento óptico de caracteres (OCR).

11.9. Ejemplo del caso de prueba

La información aquí presentada fue proporcionada la Sociedad Financiera de Objetivo Limitado.

Objetivos del negocio

Predecir si una persona de acuerdo con la información presentada podrá caer en un estado moroso después de que se le otorgo un crédito (Cartera vencida).

Inventario de recursos

Herramienta Tiberius Versión 1.3.83 (demo): con límite de 5 neuronas de entrada, 250 patrones y 4 neuronas ocultas.

Datos reales de 250 personas, después de varias combinaciones lo siguientes datos fueron los que proporcionaron un mejor resultado.

Monto crédito, Ingresos mensuales, Monto inversiones, Monto egresos, Monto deudas largo plazo, Meses, y la posibilidad de que caiga en un estado de deudor (Cartera Vencida)

Restricciones sobre el caso de estudio

Es un proyecto académico, por lo cual, el personal a realizar el mismo, no es experto en el área.

En cuanto a las herramientas a utilizar son de evaluación, por lo que no brindan todas las funcionalidades y tiene capacidad de procesamiento restringida.

Beneficios potenciales

Poder predecir si una persona de acuerdo a la información presentada podrá caer en un estado de deudor o no con un 98.3% de certeza

Descripción de los datos

Colección inicial de datos

Datos reales de 250 personas; Como es un software de prueba solo permite el entrenamiento de 250 datos. En la información proporcionada, se encuentran los

siguientes datos: Monto crédito, Ingresos mensuales, Monto inversiones, Monto egresos, Monto deudas largo plazo, Meses, Sexo, Edad, Estado Civil, Clases edades, Estado, Ciudad Ocupación, Clases Ingresos, Tipo Vivienda, Clases Inversiones, Clases Egresos , Clases Deudas, Valor Vivienda y la posibilidad de que caiga en un estado de deudor (Cartera Vencida).

Los datos fueron presentados con el siguiente formato:

Contratoid	Monto Credito	Cartera Vencida	Sexo	Edad	Estado Civil	Depen	Nivel Estudios	Ocupación	Ingresos Mensuales	Monto Inversiones	Monto Egresos	Monto Deudas Largo Plazo	Meses
6161	230382.5915	0	H	2	CBS	3	5	RESPONSABLE DE VENTAS	120000	0	73000	0	0
362	61500	1	H	1	CBM	2	5	EMPLEADO	12316	59575	2350	0	0.3815
327	61500	1	M	1	CBM	0	3	SECRETARIA	3084.8	59575	623.84	0	4.3727
5275	125000	1	H	2	CBM	0	5	JEFATURA DE ADQUISICION	12302	0	2696	5500	4.266
313	89000	0	H	2	CBM	0	5	JEFE DE MECANICOS (TARDE)	5634.4	86230	2506.34	0	3.1849
5270	135720.01	1	H	3	CBS	0	4	MECANICO DE GASOLINA	4816	0	1500	0	2.1885
323	94500	0	H	2	CBM	0	3	TECNICO METROLOGO	5250	91550	500.9	0	2.1502
321	71000	0	H	2	CBM	0	5	AUXILIAR DE INSPECTOR	2862	68790	700	0	2.1317
5258	171295	0	H	2	CBM	0	5	CONSULTOR	9597	0	3200	5164	1.3584
1140	76500	0	H	2	CBM	0	1	*OPERADOR GENERAL B1*	*5501*	*0*	*300*	*0*	*1*
7208	211067.8198	0	H	2	CBS	0	5	INGENIERO DE PLANTA EXTERNA	21926	0	7670.25	58650	0.5275
330	80000	0	M	4	S	0	4	PROPIETARIO	4000	77485	1751	0	0
1489	76500	0	H	2	CBM	1	6	PROPIETARIO	3002	0	500	0	0
322	71000	0	H	1	CBM	0	1	MECANICO AUTOMOTRIZ	2690	68790	1402.93	0	0
334	69000	0	H	1	S	0	3	JEFE DE PANIFICADORA	3271	66856	1098.72	0	0
325	69000	0	H	4	CBM	2	3	ANALISTA RESP.	2289.2	66850	558.06	0	0
316	69000	0	M	2	CBM	0	1	CONTADOR	6237	66850	2170	0	0
332	63400	0	H	1	CBM	0	3	DIBUJANTE	3946.5	61418	780	0	0
331	63400	0	H	2	CBM	0	1	LIQUIDADOR	2610	61418	600	0	0
337	61500	0	M	2	S	0	3	ANALISTA DE CUENTA	3706	59575	1482	0	0
274	61500	0	H	2	S	0	5	RESPONSABLE AREA INFORMATICA	4104	59575	696	0	0
317	61500	0	H	2	CBM	0	3	CAJERO DE VENTAS	2283	59575	880	0	0
326	61500	0	H	3	CBM	0	5	ASISTENTE	3040	59575	1347	0	0

Siendo:

- Monto crédito La cantidad de crédito otorgada
- Ingresos mensuales. Los ingresos que percibe la persona al mes
- Monto inversiones. Si tiene inversiones cuanto es lo que gana.
- Monto egresos. La cantidad de gastos al mes
- Monto deudas largo plazo En caso de deudas la cantidad
- Meses Tiempo que tiene para ir pagando

- **Sexo:** Sexo de la persona.

Posibles Valores	
1	MASCULINO
2	FEMENINO

- **Edad:** Edad de la persona.

Posibles Valores	
1	22-30 años
2	31-40 años
3	41-50 años
4	51 62 años

- Estado Civil

1	Casado Bienes Mancomunados
2	Casado Bienes separados
3	Soltero
4	Viudo
5	Unión libre
6	Divorciado

- Nivel de estudios

1	Sin estudio(S)

2	B/P
3	Carrera Técnica(Ct)
4	Profesional
5	Licenciatura

y la posibilidad de que caiga en un estado de deudor (Cartera Vencida)

1	Cartera Vencida
2	No cartera Vencida

Para esta practica se considerará como valor aceptable siempre y cuando el porcentaje total de aciertos se encuentre por encima del 80% , se elige este valor debido a que se pretende encontrar las variables que permitan generar un modelo optimo, cabe aclarar que se espera encontrar un modelo que sobrepase el 90%.

Lo primero que se realizó fué cargar la base de datos con el software Tiberius.

Como la versión de Tiberius utilizada solo trabaja con 5 neuronas de fue necesario realizar diversas combinaciones para evaluar cual genera un mejor resultado, Tiberius permite dar entradas tanto numéricas como de texto así iniciamos tomando una combinación de variables numéricas y de texto

Tomamos las siguientes variables para la primera combinación: Monto crédito, Ingresos mensuales, Monto egresos, Tipo de empleo, Clases Egresos, y la posibilidad de que caiga en un estado de deudor (Cartera Vencida).

Donde:

- % True representa el porcentaje de aciertos de que la persona caiga en cartera vencida
- % False representa el porcentaje de aciertos de que la persona no caiga en cartera vencida
- % Total es el porcentaje total de aciertos.

Vamos a empezar formando diferentes modelos con diferentes tasas de aprendizaje y compararemos esos resultados para buscar la mejor opción, los modelos generados se muestran a continuación:

Las variables de tipo texto no son representadas en la salida de la red, entrenamos con solo una neurona en la figura 11.9 muestra el resultado después de darle una tasa de aprendizaje de 0.7 y un tiempo de entrenamiento de 30 minutos , como se puede apreciar la salida no es muy satisfactoria, debido a que presenta un 13.7 del total de porcentaje de aciertos, para que el valor sea aceptado como favorable se espera se encuentre por encima de 80%, aunado a eso el % total de acierto no varia, en el lapso de entrenamiento, la red no es capaz de predecir si se generara o no algún cambio

Se mostrará que sucede cuando se agrega una segunda neurona, con una tasa de aprendizaje de 0.7

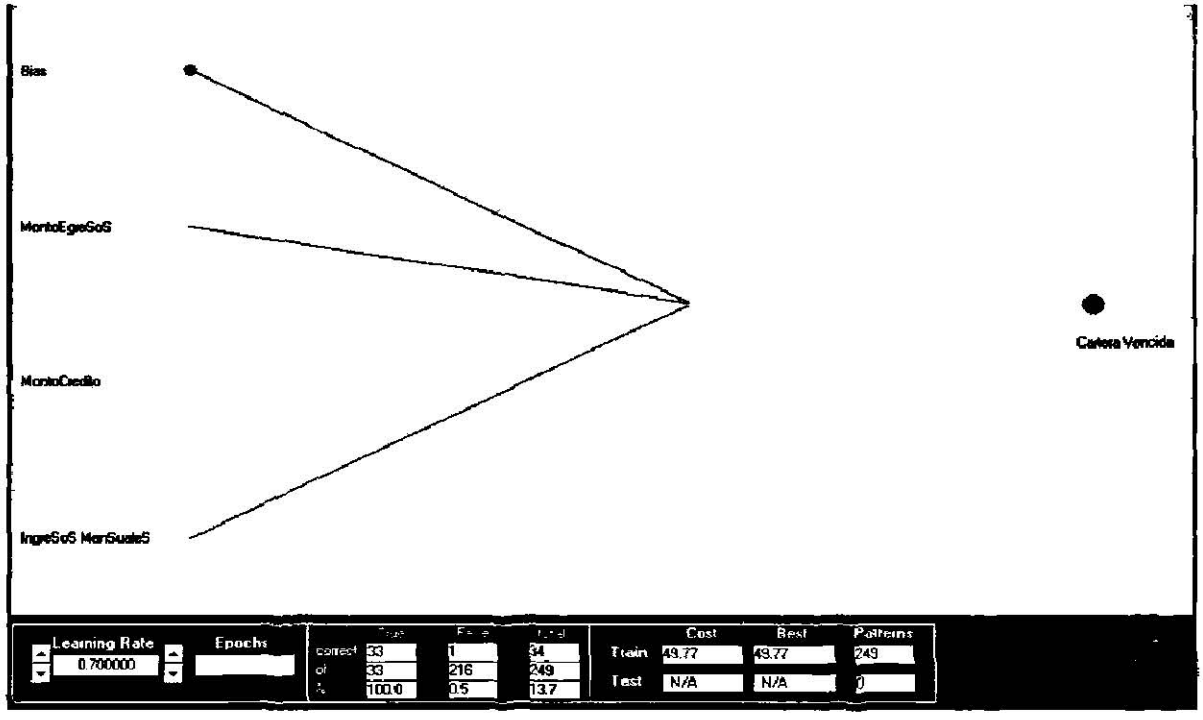


Figura 11.9.1 Modelo de una red neuronal con 1 sola neurona y una tasa de aprendizaje de 0.7

Con dos neuronas ocultas nos genera el siguiente resultado, comparando los dos resultados vemos que mejoro la salida, pero aun así no es un resultado aceptable, se puede ver en la figura 11.9.2 que el porcentaje total de aciertos esta por encima del 80%, pero al tener dos neuronas limita a que esta red pueda aprender, este mismo modelo fue dejado mas de un día de entrenamiento y el porcentaje de acierto no varió al momento de utilizar esto datos como base, lo resultados no fueron satisfactorios ya que la salida generada por el porcentaje de aciertos era menor que la presentada en el modelo original se seguirá utilizando una tasa de aprendizaje del 0.70

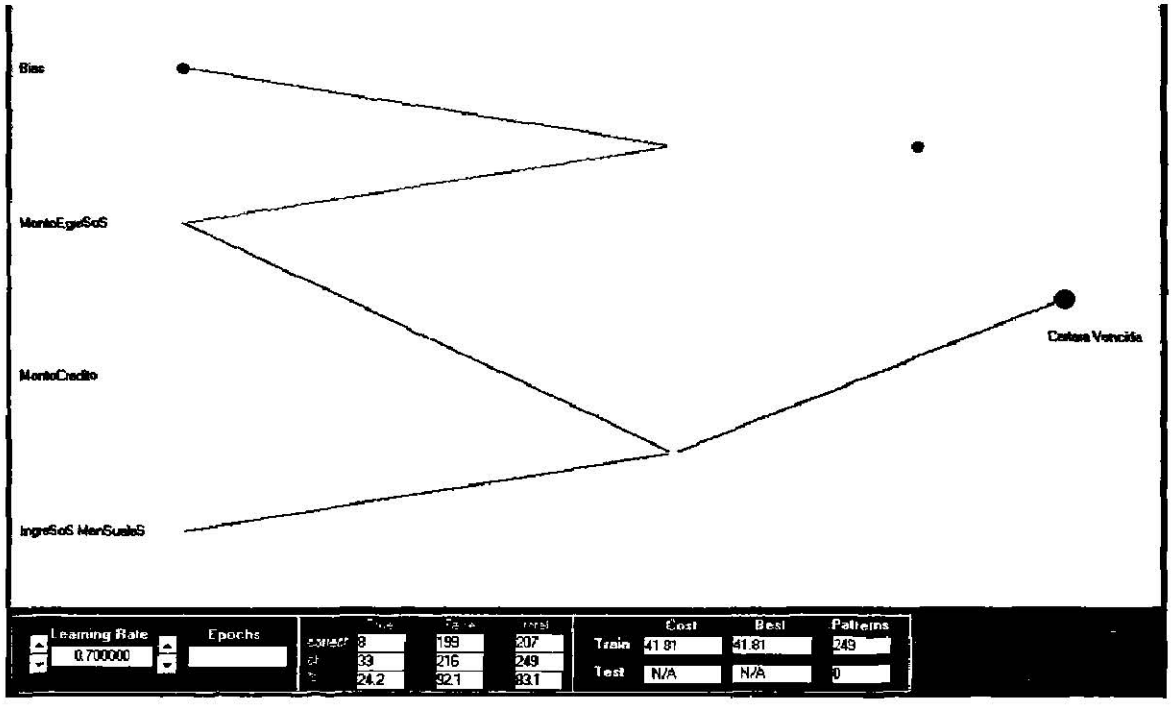


Figura 11.9.2 Modelo de una red neuronal con 2 neuronas y una tasa de aprendizaje de 0.7

Con tres neuronas ocultas y una tasa de aprendizaje del 0.7 se espera que mejore la salida. Sin embargo sucedió lo contrario, generó una salida como si tuviera una sola neurona, hay que recordar que los datos de entrada son muy significativos, ya que de eso depende que la red pueda aprender o no:

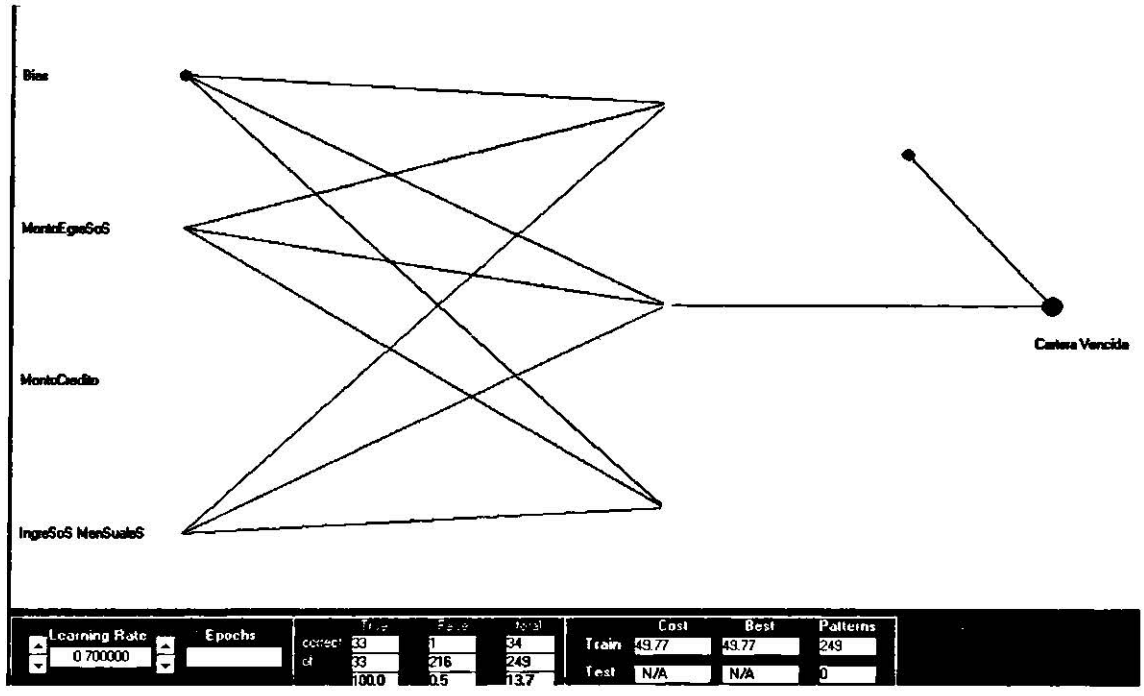


Figura 11.9.3 Modelo de una red neuronal con 3 neuronas y una tasa de aprendizaje de 0.7

Al evaluar con 4 neuronas y una tasa de aprendizaje del 0.7, mejoró la salida pero no es aceptable ya que el porcentaje total de acierto es de 13.7 % lo cual es relativamente bajo

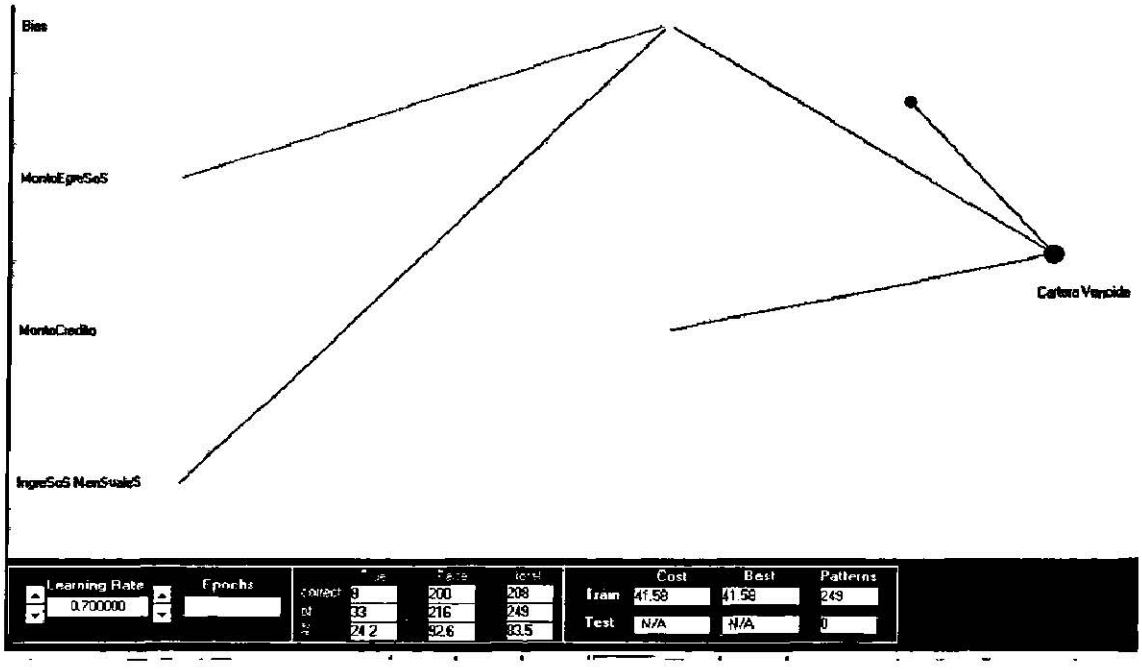


Figura 11.9.4 Modelo de una red neuronal con 4 neuronas y una tasa de aprendizaje de 0.7

Al mismo modelo se le cambió la tasa de aprendizaje a 0.001, con la tasa de aprendizaje se especifica que tan rápido aprenderá la red.

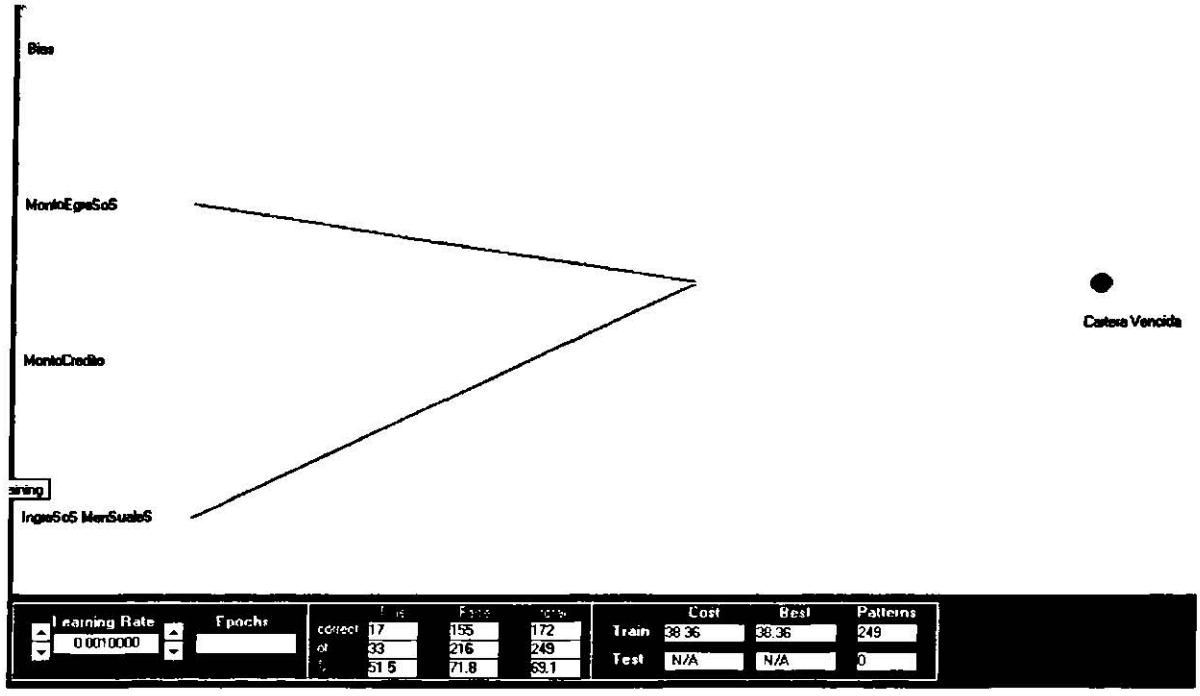


Figura 11.9.5 Modelo de una red neuronal con 1 una neurona y una tasa de aprendizaje de 0.001

Al comparar los resultado de los modelos con el mismo numero de neuronas pero diferentes tasas de aprendizaje y ver cual genera un mejor resultado, recordando que los primero modelos fueron creados con una combinación de variables numéricas y de texto y los segundos con variable numéricas.

Comparando el modelo de la Figura 11.9.1 con el de Figura 11.9.5 es posible observar que mejoró pero todavía no es un buen resultado. Esto se puede ver en los resultado obtenidos en el porcentaje total de aciertos, la figura 11.9.1 es del 13.7% y el obtenido en la figura 11.9.5 es del 69.1%, los cuales están muy por debajo del mínimo establecido. Se siguió probando para evaluar posible mejora en la salida.

Con dos neuronas ocultas y una Tasa de aprendizaje del 0.001

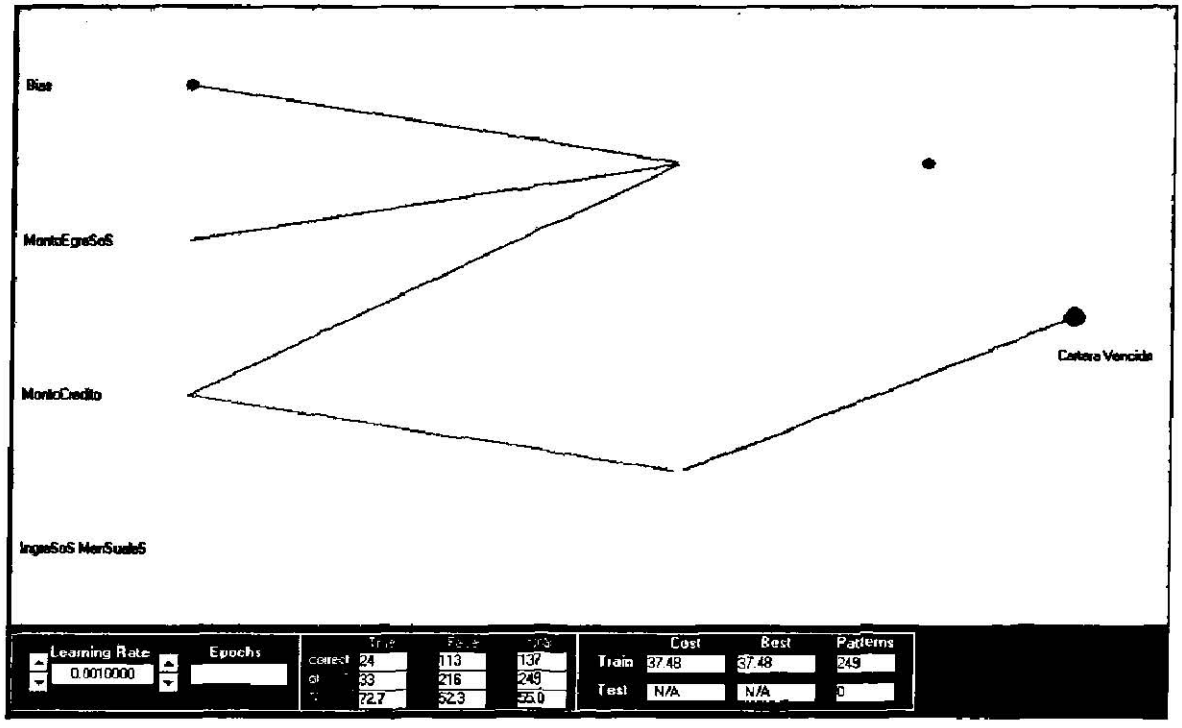


Figura 11.9.6 Modelo de una red neuronal con 2 neuronas y una tasa de aprendizaje de 0.001

Comparando el modelo de la Figura 11.9.2 con el de Figura 11.9.6 se puede apreciar que es mucho mejor la salida de la Figura 11.9.2 , esto se debe a que el porcentaje total de aciertos es del 83.5% y del segundo modelo es del 52.3%, pero recordemos que como solo utiliza dos neuronas es difícil que la red pueda aprender y mejorar los resultados.

Con tres neuronas y una Tasa de aprendizaje del 0.001

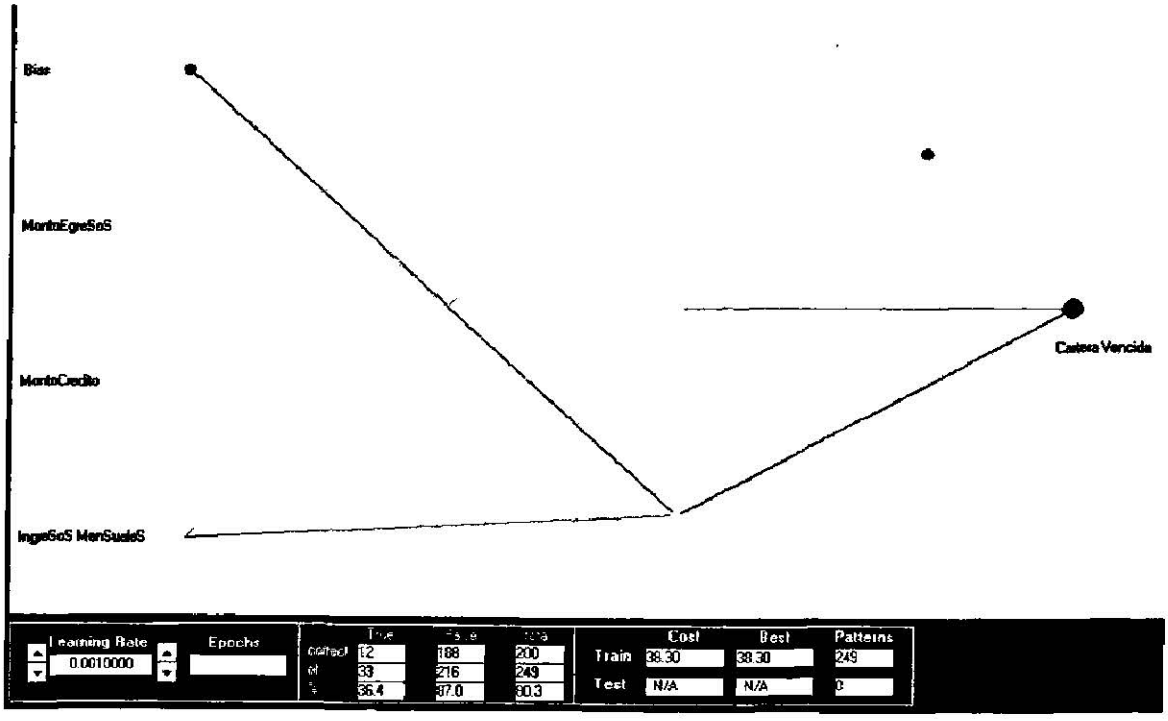


Figura 11.9.7 Modelo de una red neuronal con 3 neuronas y una tasa de aprendizaje de 0.001

Al comparar el modelo de la Figura 11.9.3 con el de Figura 11.9.7 se puede apreciar que es mucho mejor la salida de la Figura 11.9.7 , en la figura 11.9.3 ,el % total de aciertos es de 13.7% y en el de la figura 11.9.7 es de 80.3%,los resultados obtenidos no son aceptables debido a que uno de ellos apenas alcanza el mínimo establecido y el otro esta muy por debajo, por tal motivo el modelo no puede ser considerado como aceptable.

Con cuatro neuronas y una Tasa de aprendizaje del 0.001

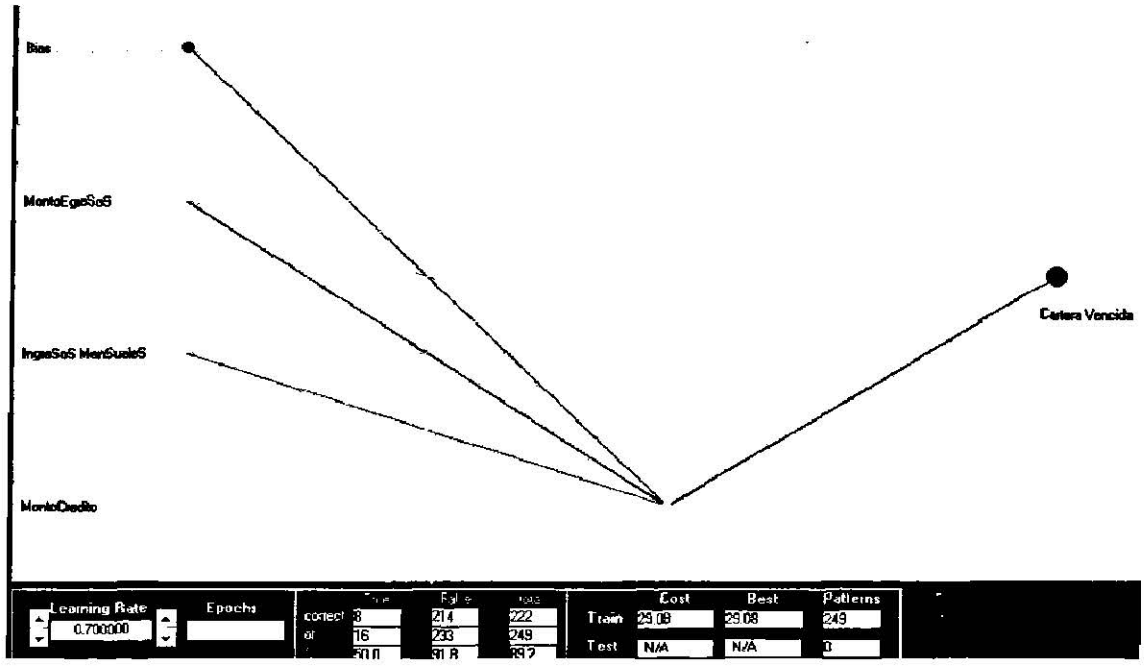


Figura 11.9.8 Modelo de una red neuronal con 4 neuronas y una tasa de aprendizaje de 0.001

Comparando el modelo de la Figura 11.9.4 con el de Figura 11.98 se puede apreciar que es mucho mejor la salida de la Figura 11.9.8,el porcentaje de aciertos es de 89.2% y el de la figura 11.9.4 es de 83.5 , esta dos ultimas comparaciones fueron muy buenas se podría tomar como valores aceptables, pero lo que se espera es encontrar un modelo, el cual genere el mínimo de error posible.

De las comparaciones se puede concluir que el modelo con una de aprendizaje de 0.001 fue el que obtuvo un mejor resultado pero los valores de salidas son descartados debido a que se espera encontrar un modelo que genere una salida en un rango 90-99%.

Analizando la salida de los modelos anteriores

Estas tablas resumen las salidas de los dos modelos anteriores con su respectiva tasa de aprendizaje, como se pueden observar las salidas no son muy favorables, esto se debe a que se tomo como mínimo el % total con valor de 80% y el máximo valor obtenido hasta el momento fue de 89.2 % si utilizáramos alguno de estos modelos es resultado no sería muy convincente, así que seguiremos buscando un mejor modelo para poder así tener una mejor solución

11.9.1	0,700000	1	100	0.5	13.7
11.9.2	0,001000	1	51.5	71.8	69.1
11.9.3	0,001000	2	24.2	92.1	83.1
11.9.4	0,700000	2	24.2	92.6	83.1
11.9.5	0,700000	3	100	0.5	13.7
11.9.6	0,001000	3	36.4	87.0	80.3
11.9.7	0,700000	4	24.2	92.6	83.1
11.9.8	0,001000	4	50.0	91.8	89.2

Tabla 11.9.6 Representación de los valores utilizando variables de texto y numéricas con diferentes numero de neuronas y diferentes tasas de aprendizaje

Posteriormente se tomaron solo valores numéricos y se analizaron cada una de las salidas.

Al tomar solo una neurona y una tasa de aprendizaje de 0.7. Los datos que se tomaron son: Monto de inversiones, Dependientes, Monto de egresos, Monto de crédito, ingresos mensuales.

1	0,700000	1	100	0.5	13.7
2	0,004000	1	36.4	87.5	80.7
3	10.00000	1	100	0.5	13.3
4	0,700000	2	100	8.8	20.9
5	0,004000	2	48.5	75.5	71.7
6	10.00000	2	100	0.5	13.7
7	0,700000	3	100	11.6	13.7
8	0,004000	3	60.6	67.6	66.7
9	10.00000	3	97.6	6.5	18.5
10	0,700000	4	100	0.9	14.1
11	0,004000	4	54.5	73.6	71.1
12	10.00000	4	100	0.5	13.7

Tabla 11.9.7 Representación de los valores utilizando numéricas

El resultado mostrado en la Tabla 11.9.7 I no fue muy buena debido a que el valor máximo alcanzado fue del 80% y este valor es considerado como mínimo.

Después de generar diversas combinaciones la que aportó un resultado favorable fue la siguiente:

- Monto crédito
- Ingresos mensuales
- Monto inversiones,
- Monto egresos
- Meses

Parámetros

Se realizaron varias configuraciones de redes para resolver éste caso de estudio, y luego de analizar los porcentajes de aciertos en los resultados se seleccionó uno.

1	0,700000	1	93.8	93.2	93.2
2	0,004000	1	93.8	91.1	91.2
3	3,000000	1	0	100	93.6
4	0,700000	2	9.38	94.4	94.4
5	0,004000	2	93.8	91.5	91.6
6	3,000000	2	6.3	100	94.0
7	0,700000	3	93.8	94.4	94.4
8	0,004000	3	93.8	92.7	92.8
9	3,000000	3	0	100	93.6
10	0,700000	4	93.8	98.3	98.3
11	0,004000	4	93.8	91.5	91.6
12	3,000000	4	100	0	6.4

Tabla 11.9.7 Valores obtenidos del ultimo modelo

La mejor solución fue

- Tasa de aprendizaje: 0,700000
- Cantidad de neuronas ocultas: 4.

Tiempo de entrenamiento de datos

En esta parte tomamos el mismo modelo de la tabla 11.9.7 al cual le vamos a dar diferentes tiempos y analizaremos si el resultado se mantiene o varía.

1	24 horas	0,700000	4	93.8	98.3	98.3
1	1 hora	0,700000	4	93.8	98.3	98.0
1	30 min.	0,700000	4	93.8	97.4	97.2
1	15 min.	0,700000	4	93.8	97.4	97.2
1	5 min.	0,700000	4	93.8	97.4	97.2

Tabla 11.9.8 Representación de los valores obtenidos después de cierto tiempo de entrenamiento

Después del entrenamiento se puede apreciar en la tabla 11.9.8 que después de 24 horas de entrenamiento mejoro considerablemente el resultado como se puede apreciar en la tabla se alcanzo un valor de 98.3% el cual resulta muy bueno.

Ejecución del Modelo

Gráficamente, el modelo se ve de la siguiente manera:

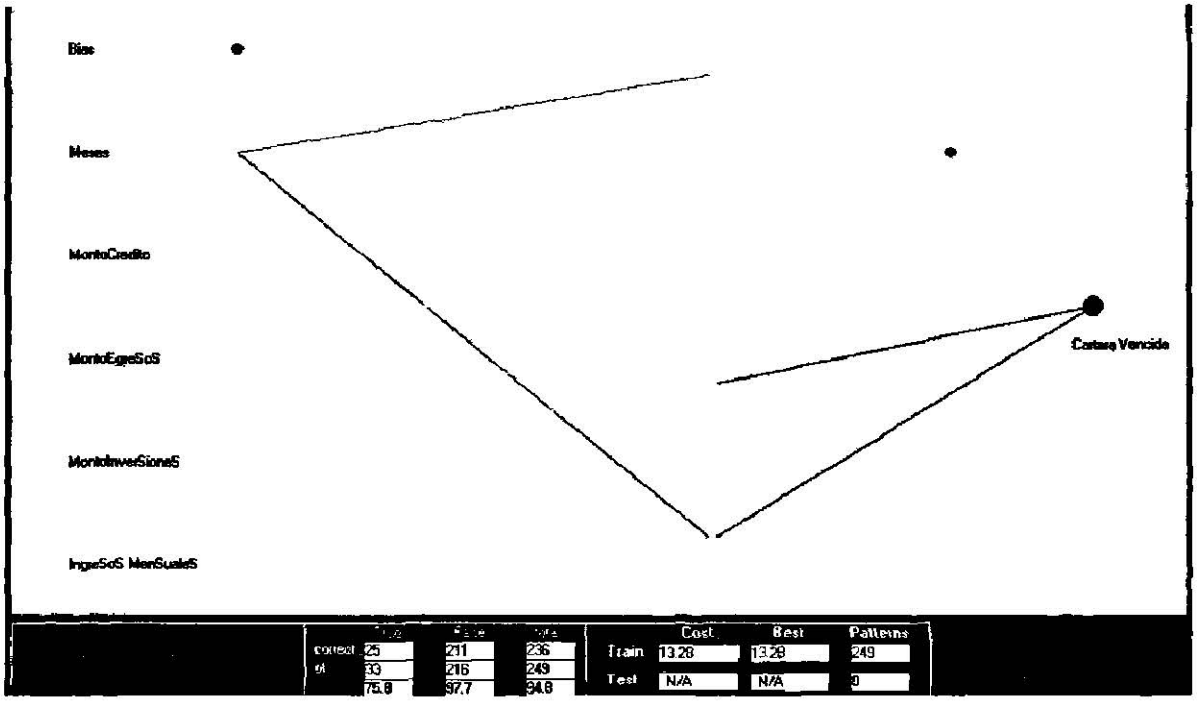


Figura 11.9.10 Representación de una red creada partir de un modelo ya existente

Esta red se probó de modo que se dejó una hora. El resultado fue muy bueno pero se decidió dejar un día para evaluar el resultado que mucho mejor, como se vio en los ejemplos anteriores, los resultados mejoraron considerablemente cuando se utilizaron 4 neuronas. Así que se pasó de lleno a entrenar con 4 neuronas y una tasa de aprendizaje del 0.7 ya que resultó óptima.

Como la red generó una salida muy aceptable, se espera que al cargar nuevos datos (ver apéndice A) el resultado sea óptimo, la figura representa un porcentaje total de 94.8

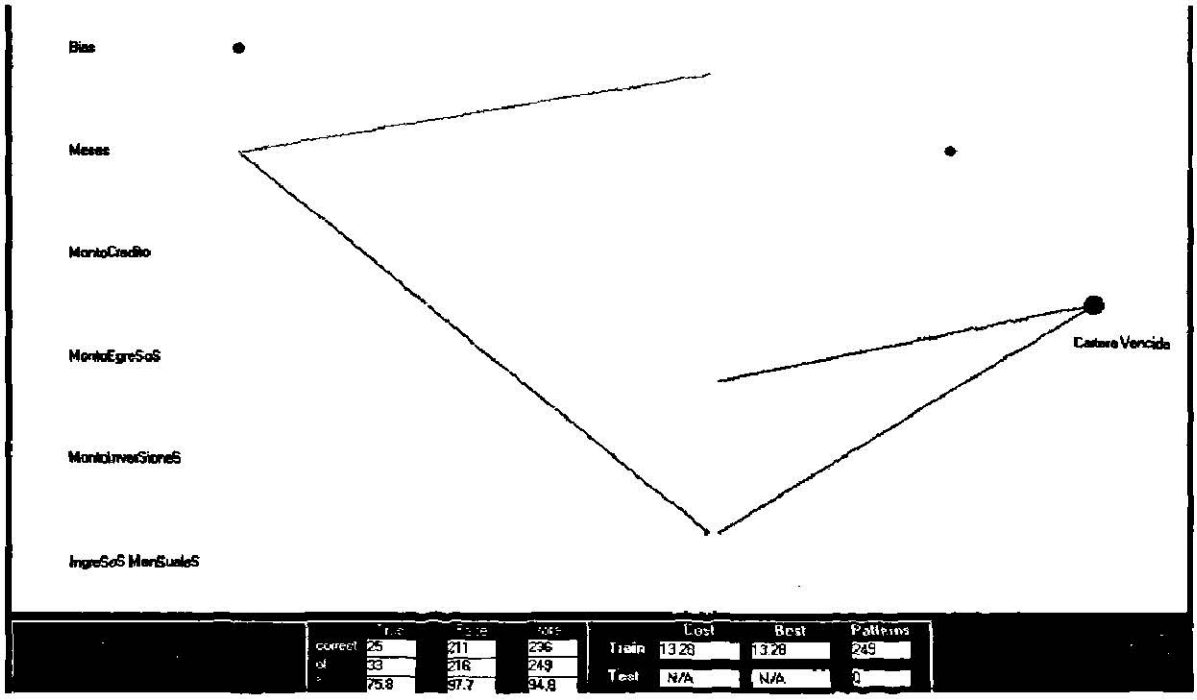


Figura 11.9.10 Representación de una red creada partir de un modelo ya existente

Comparando el resultado obtenido al cargar diferentes combinaciones a partir de un modelo ya existente se obtienen los resultados que se muestran en la siguiente tabla

1	0,700000	4	93.8	98.3	98.3
2	0,700000	4	75.8	97.7	94.8
3	0,700000	4	93.8	97.4	97.2
4	0,700000	4	62.5	97.9	95.6
5	0,700000	4	93.8	91.5	91.6

La información que fue proporcionada en una tabla de Excel consistió en total de 6500 registros de los cuales se tomaron 250 de manera aleatoria para crear las diferentes salidas, el resultado fue satisfactorio ya que ninguno de los porcentajes totales fue menor al 90 %.

Conclusiones

El software Tiberius resultó ser una herramienta sencilla para el manejo de información, se observó que es necesario realizar un análisis detallado de los datos para poder encontrar un modelo adecuado para llevar acabo el entrenamiento. Una vez que se tiene el modelo adecuado y se ha entrenado, al utilizar el mismo modelo con nuevos datos, la salida se genera rápidamente. Esto lo podemos notar al momento de cargar un modelo ya existente y darle nuevos valores, se analiza la salida que se genera y se decide si es un buen resultado. En la práctica se pudo apreciar que el resultado después de partir de un modelo ya entrenado generó salidas aceptables. Cabe aclarar que se puede especificar un rango aceptable de valores y basarse en él para decir si la salida se acepta o no. Algo que es muy importante recordar es que si no se dan los valores adecuados para la creación del modelo, por más tiempo que se deje a la red entrenando, esta no será capaz de aprender y de generar resultados óptimos, ya que lo que se pretende al trabajar con información es un *resultado preciso y en poco tiempo.*

Esta práctica pretende considerar la minería de datos como una alternativa útil en la resolución de problemas de aplicación de las redes neuronales ya que la minería de datos se emplea para mejorar el rendimiento de procesos de negocio o industriales en los que se manejan grandes volúmenes de información estructurada y almacenada en bases de datos.

Conclusión de la tesis

Cuando empecé a trabajar en esta tesis me encontraba muy emocionada la realizar este trabajo de investigación. Seguramente será de utilidad a las personas que se interesen en esta área. Las redes neuronales, podría pensarse que es un tema complejo, que emplea matemáticas complejas y conceptos que no nos son comunes, pero realmente es una área fantástica y los invito a ustedes lectores que se den la oportunidad de conocer un poco de las redes neuronales.

Vimos como empezó el estudio de las redes neuronales, cuando los científicos intentaron encontrar una respuesta de cómo crear entes artificiales capaces de aprender y se preguntaron cómo esto pudiera darse. Hubo un tiempo que esto se vio obstaculizado ya que el perceptrón normal no podía dar las soluciones que se necesitaban y es aquí cuando surge el perceptrón multicapa. El perceptrón multicapa surge para dar una solución a la limitante que se tenía con el perceptrón ya que en ese entonces el perceptron solo podía resolver problemas que fueran linealmente separables y por esa misma causa sus aplicaciones eran limitadas. Desde que se empezó a tomar el concepto de perceptrones multicapas se presentaron soluciones mas precisas y se logró hacer análisis de nuevos modelos. Una vez que tenía que modelo de perceptron se utilizaría, se buscó el mejor método de entrenamiento para presentar la aplicación en este caso fue la retropropagación.

En cuanto se cuanta con un método de entrenamiento adecuado, es necesario hacer un análisis detallado de la información proporcionada. Durante el análisis de la solución se observó que para empezar el método de aprendizaje hay que seguir una serie de pasos

para obtener la mejor solución del problema y durante esta fase se debe analizar si este tipo de entrenamiento proporcionara la mejor solución. Se pudo apreciar que se realizó un análisis detallado de la información, se generaron diversas combinaciones, con diferentes tazas de aprendizaje, diferente número de neuronas y se hicieron las comparaciones necesarias, parra elegir el modelo óptimo.

Cuando se entrena una red neuronal y se pretende obtener una solución certera, se debe tener en cuenta que aunque se utilice una determinada red neuronal no es seguro obtener un buen resultado. Si los datos de entrenamiento no son los adecuados, no se obtendrá una solución esperada.

La utilización del software Tiberius fue, a mi punto de vista, muy buena. La única desventaja es que está restringido a solo 250 patrones por ser software de evaluación. Si se hubiera podido realizar la práctica con más datos se habría obtenido una solución mucho mejor. Ya que la red hubiera tenido la oportunidad de aprender de muchos patrones más. La información proporcionada fue de 6500 patrones.

Las personas aun en este tiempo son renuentes al uso de nuevas tecnologías que le ayuden a mejorar la optimización de sus procesos , prefieren la utilización de métodos rudimentarios en los que se pueden tardar semanas o días para obtener un resultado deseado, ya que al oír la palabra cambio y tecnología sienten que serán desplazados por la tecnología actual, por eso es importante que conozcan y sepan de la ventajas de la utilización de esta tecnologías, ya que uno es quien controla la tecnología y no la tecnología lo controla a uno .Las utilización de las redes neuronales nos brindan la ventajas de obtener mejores resultados y esto se debe a que las redes neuronales son capaces de aprender, se les puede entrenar y permiten resolver un problema mediante el empleo de problemas resueltos en el pasado similares al actual. (ya no se que poner) El

objetivo de la práctica fue mostrar la utilización de las redes neuronales en el tratamiento de información. La aplicación de tecnología como lo es las redes neuronales se están empleando cada vez mas en la solución de problemas, ya sea ciencia, seguridad, medicina, administración, traducción de caracteres entre otros. Tenemos las herramientas para facilitarnos la resolución de este tipo de problemas, utilizar esta tecnología y darle el uso adecuado depende solo de nosotros."Podemos trascender creando o destruyendo, la cuestión es como quiere uno trascender" (Erich Fromm)

Apéndice A

SOFWARE TIBERIUS

Se presenta una introducción breve acerca del software Tiberius cabe aclarar que solo se describen las funciones que fueron utilizadas para la aplicación

Tiberius es desarrollado por Phil Brierley y Anderw Lewis y modela una red backpropagation, con límite de 5 neuronas de entrada, 250 patrones y 4 neuronas ocultas.

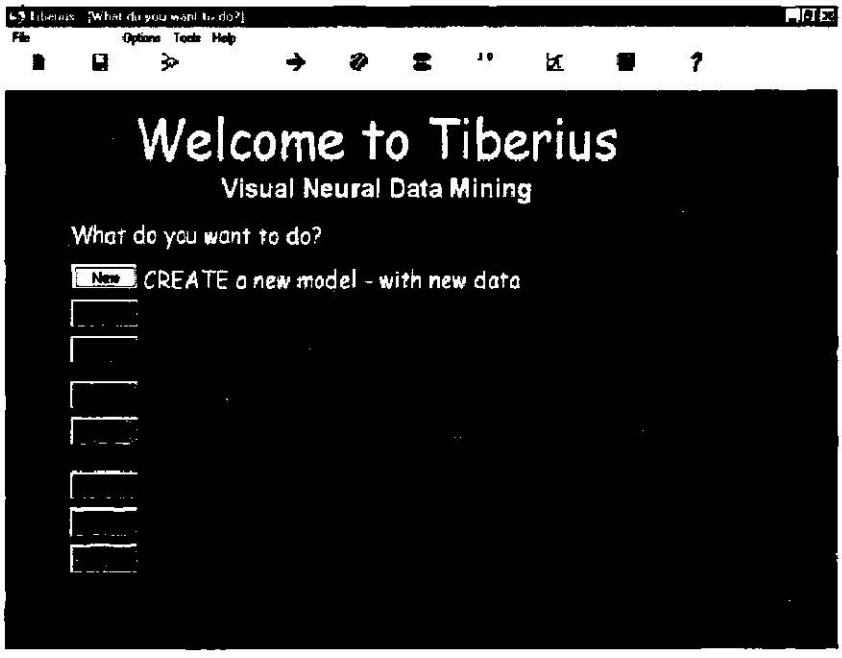


Figura I. Pantalla Principal

En la pantalla principal que se muestra en la figura I, Se hace clic en New y de manera terminada se abre la carpeta de Demos de Tiberius para seleccionar la base de datos en

caso de que el archivo no se encuentre en esta carpeta hay que dar la ubicación del archivo

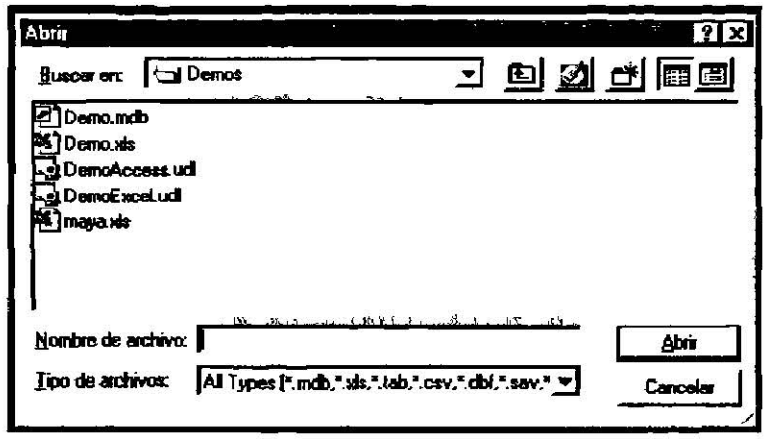


Figura II .Origen de los datos

Formato de datos

Para que la herramienta seleccionada pueda trabajar con los datos, los mismos debieron ser cargados en una tabla, en una base de datos Access.

Se renombraron las columnas por nombres más descriptivos, y se modificaron en los valores de la columna de la salida, para clarificar más el modelo, siendo 0 cuando la persona esta interesada en comprar y 0 cuando no tiene interés. Como se muestra en la siguiente Tabla

EDAD	NSE	SEXO	AUTOMOVI	INSTRUCC	ROL	INTCOM
38	1	1	1	6	1	0
39	1	1	1	7	1	0
28	1	2	2	7	4	0
30	1	1	1	9	1	1
34	1	2	2	7	1	0
39	1	1	2	9	4	1
22	1	1	1	6	4	0

Construcción del modelo

El modelo consiste en un conjunto de datos (tabla con los datos de entrenamiento), una herramienta de análisis y un conjunto de parámetros que serán citados más adelante.

Para crear el modelo en Tiberius, se debe especificar:

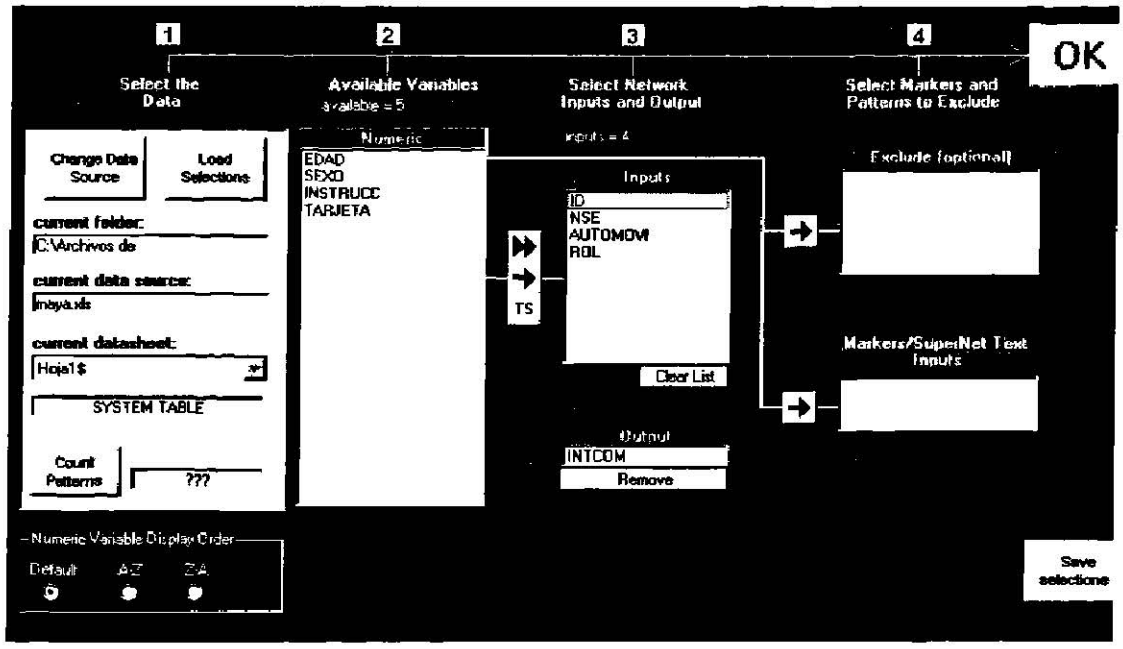


Figura III. Datos que conformaran la red

1. La base de datos que contienen los patrones de entrada a ser "aprendidos" por la red.
2. Se debe indicar cuáles son las entradas a la red y cuál es la salida.
3. El caso que se quieran excluir patrones, se debe indicar, y en que condiciones se quieren excluir los mismos.

Se mostrarán brevemente las funciones que se utilizaron para la realización del análisis de los datos.

Train: Permite empezar con el entrenamiento de la red en base a los dato guardados o si se desea hacer el análisis con datos aleatorios.

En esta parte se puede cambiar o restaurar los valores originales. También permite agregar y remover las neuronas teniendo como máximo 5 neuronas ocultas

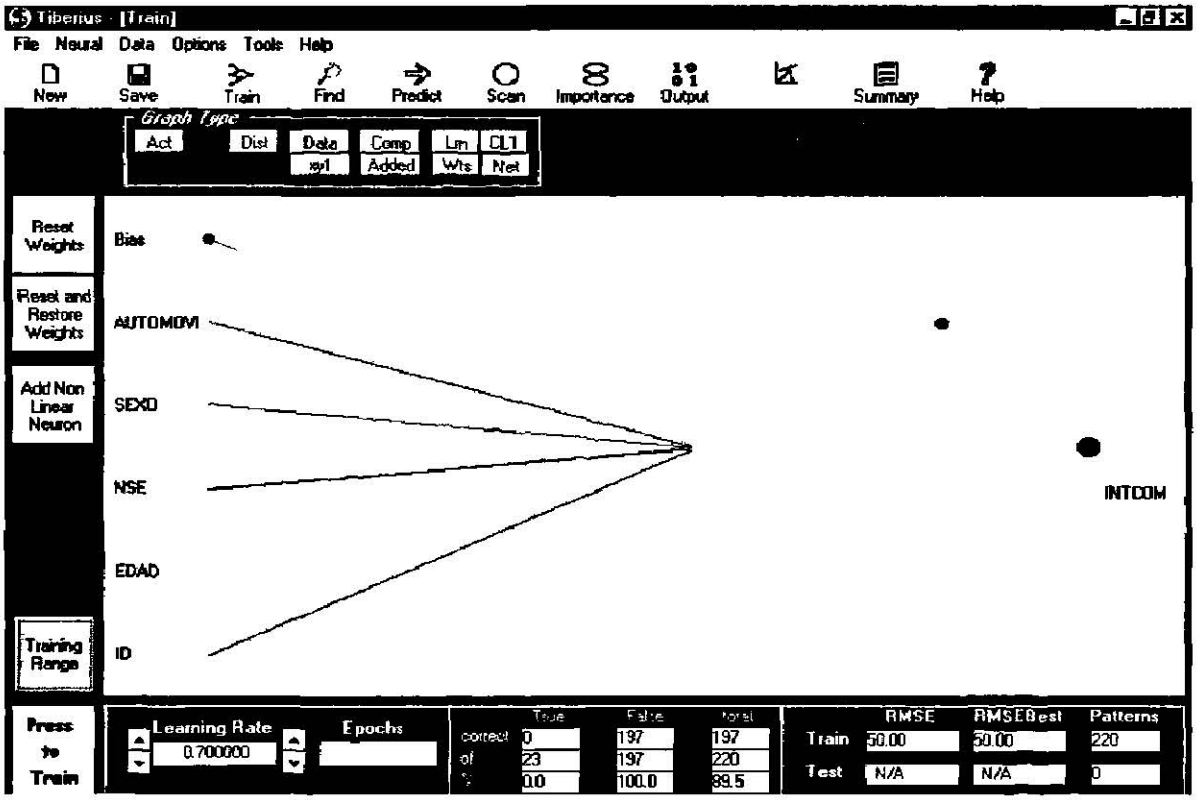


Figura IV. Representación de la red neuronal con una sola neurona

En la figura I se muestran los siguientes parámetros:

- % True representa el porcentaje de aciertos de los datos suministrados
- % False representa el porcentaje de aciertos de los datos suministrados
- % Total es el porcentaje total de aciertos.
- Learning Rate es la tasa de aprendizaje

Predict. Esta parte nos muestra todos los datos con su respectivo resultado después del entrenamiento

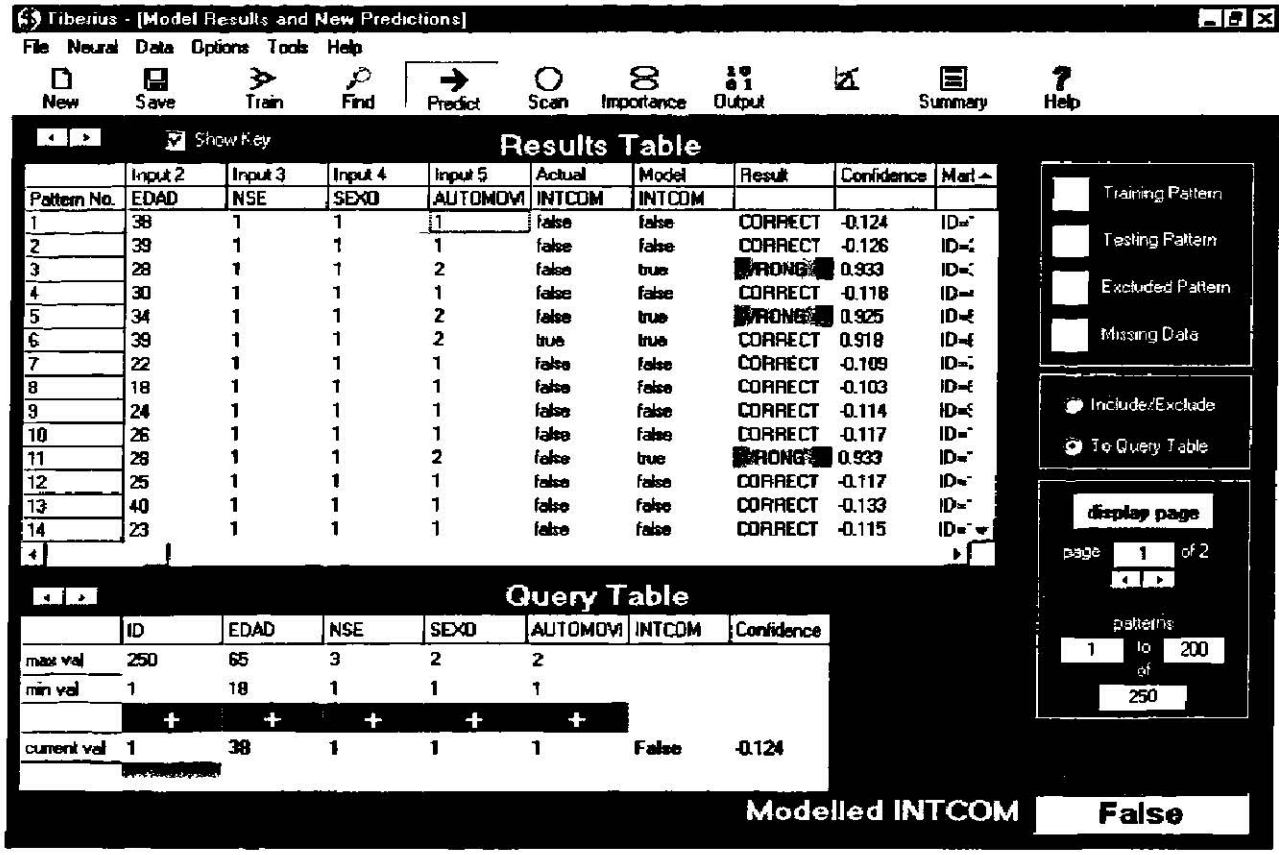


Figura VI. Representación de los valores obtenidos después del entrenamiento

Aquí se puede considerar un rango aceptable si se encuentra (0.7-0.9) en caso contrario sería desfavorable

Una vez que ya se tiene creado un modelo se siguen los siguientes pasos para cargar nuevos datos

Primero seleccionamos la opción Predict que se muestra en la Figura I, se nos despliega la siguiente pantalla figura en la cual vamos a seleccionar el origen de la información que se quiere entrenar, cabe recalcar que en la información las variables de entradas deben coincidir con la información a cargar ya que de lo contrario generara error, en la pantalla se puede apreciar las variables del modelo original y las variables que se requieren para la predicción

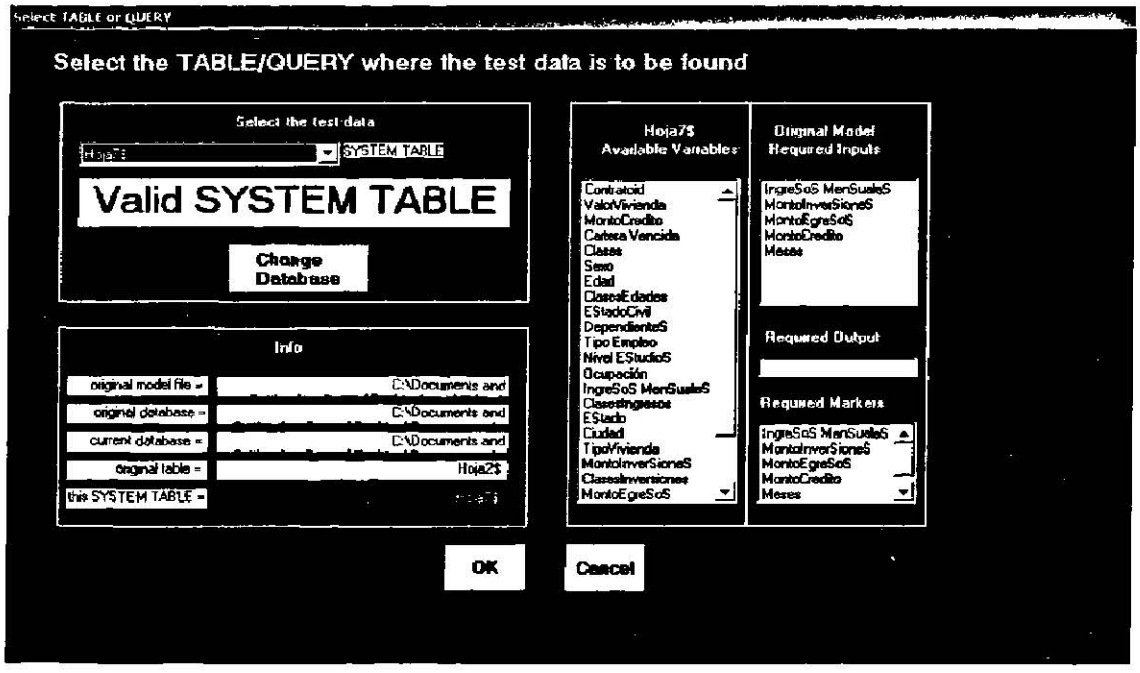


Figura VII. Pantalla para cargar nuevos valores de un modelo ya existente

Una vez que cumplimos con el requisito, cargamos los valores y automáticamente se genera la salida

Bibliografía

Inteligencia Artificial.
Elaine Rich. Knight Kevin.
Segunda Edición.
Mc Graw Hill.
México 1994.

Inteligencia Artificial un Enfoque Moderno.
Stuart Rusell. Norving Meter.
Printice Hall.
México 1996.

Inteligencia Artificial: Conceptos, Técnicas y aplicaciones.
Mompin P. José.
Marcomobo S.A Ediciones.
España 1987.

Inteligencia artificial y Minirobots
Alberto Delgado
Primera Edición
Editorial ECOE, Bogotá - Colombia.
Febrero de 1998

La inteligencia Artificial
John Haugeland
Primera Edición
Siglo veintiuno editores s.a de C.V.
México 1988

Memoria natural y artificial
Laura Viana Catrillon
Primera Edición
Fondo de cultura Económico
México 1990

Nebendah Dieter.
Sistemas Expertos. Ingeniería y Comunicación.
Editores Marcombo.
Barcelona 1988.

Principios de Inteligencia Artificial y Sistemas Expertos.
Rolston W. David.
Mc Graw Hill.
México 1992.

Redes neuronales: Algoritmos, Aplicaciones y técnica de programación
Freeman James Y Skapura.
Addison-Wesley Iberoamericana y Ediciones Díaz de Santos
1993

Mente y cerebro
Fischbach Gerald D, Semir Zeki, Eric R. Kandel y Robert D. Hawkins
Investigación y Ciencia.
Noviembre 1992

Redes neuronales que aprenden de la experiencia
Hinton Geoffrey
Investigación y Ciencia.
Noviembre 1992

Introduction to artificial intelligence.
Philip C. Jackson, Jr.
Dover Publications, Inc. New York.
Second Edition

Self Organization and Associative Memory. Kohonen.
Springer-Verlag
3rd Edition
Berlín.1988

Neural computing : theory and practice / Philip D. Wasserman.
p. cm.
Includes bibliography and index.
ISBN 0-442-20743-3
1. Neural computers. I. Title.
QA76.5.W353 1989 88-34842
006.3-dc19 CIP

Neural networks and natural intelligence.
Includes index.
1. Neural circuitry. 2. Neural computers. 3. Artificial intelligence. 4. Intellect.
Grossberg, Stephen, 1939-
QP363.3.N44 1988 612'.81 88-2942
ISBN 0-262-07107-X


ADQUISICION