UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

INTRODUCCIÓN A LAS REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES EN LA INGENIERÍA ELÉCTRICA-ELECTRÓNICA

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERA MECÁNICA ELECTRICISTA

PRESENTA:

MARÍA DEL CARMEN FLORES HIDALGO

ASESORES:

M. EN C. VICENTE MAGAÑA GONZÁLEZ

ING. FERNANDO PATLÁN CARDOSO

CUAUTITLÁN IZCALLI, EDO. DE MÉXICO 2010

AGRADECIMIENTOS

A DIOS:

Le agradezco por mi vida y la vida de mis seres queridos, por haberme devuelto la salud, por todo lo que me ha dado y lo que no me ha dado, por su amor, su misericordia y su fidelidad, por ser mi luz en las tinieblas, mi refugio en la tormenta, mi Padre, mi amigo fiel, mi protector, mi proveedor y mi consuelo. “Gracias”

“Mira que te mando que te esfuerces y seas valiente, no temas ni desmayes, porque Jehová, tu Dios, estará contigo donde quiera que vayas.” Josué 1-9

A MIS PADRES:

Madre te agradezco con todo mi corazón por el amor, cariño, ternura, paciencia, esfuerzo, sacrificio, comprensión, entrega y dedicación que nos has brindado a mis hermanos y a mí, gracias porque siempre has creído en mi y has impulsado cada uno de mis sueños hasta verlos hecho realidad, gracias porque a pesar de todo siempre has procurado nuestra felicidad y bienestar y has luchado contra tempestades como toda una guerrera para sacarnos siempre adelante. Eres y siempre serás mi mejor amiga. Bendigo a Dios por darme el hermoso privilegio de ser tu Hija.

Padre gracias porque a pesar de las circunstancias y los problemas siempre has procurado estar con nosotros, nos has ayudado en todo lo que has podido, invirtiendo tu tiempo, salud, trabajo, fuerzas e ingenio en busca de oportunidades para que nuestras vidas fuesen mejor, gracias por enseñarnos a trabajar y esforzarnos más cada día, gracias por todo el apoyo que me brindaste para que pudiera terminar mi formación académica. Quiero que sepas que a pesar de todas las diferencias que hemos tenido te quiero mucho, eres mi Padre y nada cambiara ese hecho. Dios te bendiga Papá.

A MIS HERMANOS:

Enrique, Guillermo, Jorge y Jesús gracias por su amor, cariño, confianza, comprensión, lealtad, complicidad, consejos y apoyo incondicional, gracias por las vivencias, las travesuras, los juegos, las alegrías y las aventuras, gracias porque en cada uno de ustedes además de un hermano tengo un amigo.

El vivir, crecer, aprender, trabajar, sufrir y reír a su lado ha sido un verdadero honor, quiero que sepan que son una Bendición de Dios en mi vida y estoy muy orgullosa de ustedes, los quiero mucho.

A mi hermana María porque formas parte de mí y me has cuidado desde donde estas. “Gracias hermanita te dedico este triunfo especialmente a ti.” (✞)

A MIS CUÑADAS Y SOBRINOS:

Guadalupe y Anahí gracias por su amistad, confianza, tolerancia, paciencia y consejos. Gracias a José Antonio, Enrique, Paulina, Ángel, Ameyallí y Josué por su amor, cariño, ternura, inocencia y alegría, porque cada uno de ustedes es una Bendición de Dios y son en mí vida fuente de inspiración.

A LA FAMILIA PINEDA GARCIA:

Gracias al Sr. Lindoro Pineda y a su esposa Ángeles García, por su apoyo y confianza, por abrirme las puertas de su casa y hacerme sentir entre familia. Gracias a Aleida, Yoshia, Ricardo, Tashai, Claudia, Angélica, Emilio y Lucia por su por su amistad, tolerancia y paciencia.

CARLOS:

Gracias por tu amor, cariño, amistad, alegría, paciencia y apoyo incondicional, por todos los hermosos momentos que hemos vivido juntos, gracias porque has llenado mi vida de sueños e ilusiones, por ayudarme a creer en mí, por darme fuerzas y aliento en los momentos que he querido renunciar, porque siempre has estado con migo sin importar las circunstancias, porque has sido para mí el mejor de los amigos, con el que he llorado, he reído, he compartido, he trabajado, he luchado y he amado. Gracias por permitirme ser parte de tu vida y ser parte de la mía. TAM

A LA UNIVERSIDAD:

Gracias a la Máxima Casa de Estudios por darme el gran privilegio de formarme como profesionista dentro de sus aulas.

A MIS AMIGOS Y COMPAÑEROS:

Gracias por la amistad que me brindaron, por el apoyo y por hacer inolvidables los momentos que compartimos a lo largo de nuestra estancia en la escuela.

A MIS PROFESORES:

Gracias a todos y cada uno de mis profesores que tuve a lo largo mi formación académica, por su tiempo, paciencia, entrega y dedicación.

ÍNDICE

CAPÍTULO III:

TOPOLOGÍAS Y ALGORITMOS DE APRENDIZAJE DE LAS REDES NEURONALES ARTIFICIALES

CAPÍTULO IV:

EL PERCEPTRON

CAPÍTULO V:

APLICACIONES DE LAS RNA EN LAINGENIERÍA ELÉCTRICA-ELECTRÓNICA Y SU IMPLEMENTACIÓN

INTRODUCCIÓN

Diseñar y construir máquinas capaces de realizar procesos con cierta inteligencia ha sido uno de los principales objetivos de los científicos a lo largo de la historia. En un principio los esfuerzos estuvieron dirigidos a la obtención de máquinas que realizaran alguna función típica de los seres humanos. Pero esto solo era el resultado del desarrollo técnico de la habilidad mecánica de los constructores de aquellas máquinas.

Actualmente se cuenta con herramientas muy sofisticadas como lo son la microelectrónica y los lenguajes de programación, de tal forma que existen diversas maneras de realizar procesos similares a los inteligentes. Sin embargo, a pesar de disponer de estas herramientas, existe un problema que limita en gran manera los resultados que se pueden obtener. Dicho problema es que estas máquinas se implementan sobre computadoras que se basan en la filosofía de funcionamiento expuesta por **Von Neumann** (computadoras basadas en lógica digital que operan ejecutando en serie las instrucciones que componen un algoritmo que se codifica en forma de programa, el cual se encuentra almacenado en memoria).

Las ***redes neuronales artificiales*** son modelos que intentan reproducir el comportamiento del cerebro, con la intención de construir sistemas de procesamiento de la información paralelos, distribuidos y adaptativos, que puedan presentar un cierto comportamiento “inteligente”. Dichos modelos se han llevado a cabo generalmente, mediante el uso de componentes electrónicos o simulación en software mediante una computadora.

No se trata de construir máquinas que compitan con los seres humanos, sino que realicen ciertas tareas de rango intelectual con que ayudarle. Por otra parte, los sistemas que se lleguen a desarrollar no van a suponer la desaparición de las computadoras tal como las conocemos hoy en día, por lo menos en aquellas tareas para las que están mejor dotadas incluso que los seres humanos.

Las aplicaciones prácticas de las ***redes neuronales artificiales***, son realmente extensas como es el caso de la visión artificial, procesado de señales e imágenes, reconocimiento del habla y de carácteres, control remoto, inspección industrial por mencionar algunos, y su principal ventaja frente a otras técnicas reside en el procesado paralelo, adaptativo y no lineal. En éste trabajo se realiza una investigación bibliográfica acerca de las ***redes neuronales artificiales*** el cual comprende un amplio panorama de éstas concluyendo con sus aplicaciones en la Ingeniería Eléctrica-Electrónica.

CAPÍTULO I

PRINCIPIOS BÁSICOS DE LAS REDES NEURONALES ARTIFICIALES

I.1.- HISTORIA

En la Segunda Guerra Mundial comienza la construcción de máquinas inteligentes, con el diseño de computadoras analógicas ideadas para controlar cañones antiaéreos o para navegación. Algunos investigadores observaron que existía una semejanza entre el funcionamiento de estos dispositivos de control y los sistemas reguladores de los seres vivos. Combinando las nuevas teorías sobre la realimentación, los avances de la Electrónica de la posguerra y los conocimientos disponibles sobre los sistemas nerviosos de los seres vivos, se construyeron máquinas capaces de responder y de aprender como los animales. Surge el término ***cibernética*** acuñado por **Norbert Wiener** para designar este estudio unificado del control y de la comunicación en los animales y las máquinas.

En el año 1937, **Alan Turing** emprende un estudio formal de la computación; introdujo una máquina ideal conocida como ***máquina de Turing*** (primer modelo teórico de lo que después sería una computadora programable). En torno a esta fecha comienza la construcción de las primeras computadoras, como ***ENIAC*** **(**Electronic Numerical Integrator And Computer) desarrollada durante la Segunda Guerra Mundial para la ejecución de cálculos de interés militar (trayectorias balísticas, desarrollo de la bomba atómica, etc.).

El matemático de origen Húngaro **John Von Neumann**, en los años cuarenta concibe una computadora basada en lógica digital que opera

ejecutando en serie (una tras otra) las instrucciones que componen un algoritmo que se codifica en forma de programa, el cual se encuentra almacenado en memoria.

La ***arquitectura Von Neumann*** es la base sobre la que se asientan la mayor parte de las computadoras digitales actuales. La computación algorítmica se basa, en resolver cada problema mediante un algoritmo, que se codifica en forma de programa y se almacena en memoria; el programa es ejecutado en una máquina secuencial. Pioneros como **Turing** o **Von Neumann**, tenían la idea de que pudiera incorporarse en una de estas máquinas la capacidad de pensar racionalmente, en forma de un complejo software.

En 1943 surge el primer modelo histórico de una ***red neuronal*** realizado por el neurofísico **Warren McCulloch**, y el matemático **Walter Pitts**, basado en la idea de que las neuronas operan mediante impulsos binarios. Dicho modelo introduce la idea de una función de paso por un umbral, que posteriormente es utilizada en otros ***modelos neuronales***.

Posteriormente **D. Hebb** desarrolló un procedimiento matemático de aprendizaje, basado en el funcionamiento de las neuronas biológicas y las condiciones clásicas de aprendizaje, de ahí que el paradigma de aprendizaje lleve su nombre: “aprendizaje de Hebb”.

En 1950 **Claude Shannon** y **Turing** diseñaron los primeros programas que permitían a una computadora digital razonar y jugar al ajedrez.

En 1957 **A. Newell**, **H. Simon** y **J. Shaw** presentaron el teórico lógico, del primer programa capaz de razonar sobre temas arbitrarios.

Marvin Minsky es el responsable de los primeros resultados prácticos ***s*** con ***redes neuronales.*** Inspirado en el trabajo de **McCulloch** y **Pitts**. Desarrolló una máquina de 40 neuronas con conexiones que se ajustaban de acuerdo a una serie de sucesos que ocurrían al realizar ciertas tareas.

En 1957**, Frank Rosenblat** generalizó el modelo de **McCulloch-Pitts**, añadiéndole la posibilidad de aprendizaje, y a este modelo le denominó ***"Perceptron"***. Esta es la más antigua ***red neuronal***, con la capacidad de generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aunque no se hubieran presentado anteriormente.

En 1959, **Bernard Widrow** y **Marcial Of**., de Stanford, desarrollaron el modelo ***Adaline*** (Adaptative Linear Elements). Siendo ésta la primera ***red neuronal*** aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas), usada desde entonces comercialmente hasta la fecha.

En 1960 **John McCarthy** acuña el término ***inteligencia artificial*** o ***IA***, para definir los métodos algorítmicos capaces de hacer pensar a las computadoras.

En 1965 **Marvin Minsky**, **Newell** y **Simon** habían creado programas de ***IA*** que demostraban teoremas de geometría. Aunque importantes, los desarrollos citados solamente eran capaces de resolver aquellos problemas para los que habían sido construidos.

En 1967, **Stephen Grossberg** (Universidad de Boston) desarrolló la red ***Avalancha***, que consistía en elementos discretos con actividad que varía con el tiempo, la cual satisface ecuaciones diferenciales continuas para resolver actividades tales como reconocimiento continuo del habla y aprendizaje del movimiento de los brazos de un robot.

En los años setenta, el rápido progreso de la ***IA*** culminó, con la introducción de los ***sistemas expertos***, que son complejos programas de computadoras en los que se codifica el conocimiento de expertos en una materia muy concreta (concesión de créditos, diagnóstico de enfermedades, etc.), en forma de reglas de decisión. Un cuarto de siglo más tarde las computadoras son miles de veces más potentes que los de la época de los pioneros de la ***IA*** y, en general, no resultan mucho más inteligentes. El problema está en que la ***arquitectura Von Neumann*** sobre el cual se basa la ***IA***, pese a su gran potencia, presenta problemas a la hora de abordar ciertas tareas, como las denominadas del mundo real, donde la información que se presenta es masiva, imprecisa y distorsionada.

En la década de los ochenta principalmente, se vuelve a retomar una serie de paradigmas de cómputo alternativos, como las ***redes neuronales***, los

sistemas difusos, ***algoritmos genéticos*** o la ***computación evolutiva***, de los cuales, quizás los dos primeros son los más relevantes y empleados.

En este intervalo de tiempo, solamente se puede destacar el trabajo de unos pocos investigadores: **Teuvo Kohonen**, **Stephen Grossberg**, **Kunihiko Fukushima,** y **John Hopfield**.

Kohonen comenzó sus investigaciones sobre ***redes neuronales*** con paradigmas de conexiones aleatorias, aunque su contribución más meritoria es los mapas autoorganizativos (también llamados redes de **Kohonen).**

Fukushima estudió modelos espaciales y espacio-temporales para sistemas de visión artificial. Los modelos desarrollados por él son el ***"Cognitron"*** (1975), y el ***"Neocognitron"*** (1980).

Hopfield en 1982, describió un método de análisis del estado estable de una red autoasociativa, introduciendo una función de energía en sus estudios sobre sistemas no lineales.

A finales de los ochenta y principios de los noventa se desarrollaron nuevas estructuras de sistemas conexionistas, por ejemplo las redes neuronales de base radial, memorias asociativas bidireccionales, redes basadas en ondícuas. Nuevas familias de algoritmos: regla delta-bar-delta, etc.

El surgimiento de las ***redes neuronales***, también denominados ***sistemas neuronales artificiales*** o ***ANS*** (Artificial Neural Systems), se debe en parte a la gran dificultad para resolver con la eficiencia deseada problemas como los de visión o aprendizaje. Debido a los altos requerimientos computacionales de este tipo de tareas, la ***arquitectura Von Neumann*** no es muy apropiada y resulta necesaria la introducción de los sistemas de cálculo paralelo para lograr respuestas en tiempo real. Inquietudes como estas coincidieron con el desarrollo de la alta escala de integración ***VLSI*** **(**Very Large Scale Integration**)** que permitió simular y realizar estos sistemas mostrando su potencial de aplicación práctica.

Por último se encontró la forma de entrenar un ***perceptron multicapa***, con lo cual se resolvieron los problemas atribuidos al ***perceptron simple*** y se eliminaron las antiguas objeciones a los ***ANS***.

Actualmente coexisten dos corrientes importantes dentro de la búsqueda de la ***inteligencia artificial*** que, más que alternativas, son complementarias. La ***IA*** convencional, basada en algoritmos manipuladores de información simbólica, los cuales se ejecutan sobre computadoras ***Von Neumann*** y que operan sobre la base de la lógica digital, y por otro lado las ***redes neuronales artificiales***, los ***sistemas difusos*** y otras técnicas, incluidas en la denominada ***inteligencia computacional*** que intentan imitar de cierta forma la inteligencia de los seres vivos.

I.2.- DEFINICIÓN

Existen numerosas formas de definir a las ***redes neuronales artificiales***, de las cuales destacamos las siguientes:

a) *Una nueva forma de computación, inspirada en modelos biológicos.*

b) *Modelos computacionales basados parcialmente en el funcionamiento del cerebro humano.*

c) *Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.*

d) *Un sistema de computación hecho por un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de un estado dinámico como respuesta a entradas externas.*

e) *Redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.*

I.3.- SURGIMIENTO

Las ***redes neuronales artificiales*** son ampliamente utilizadas en la actualidad, ejemplo de ello es, el apoyo al diagnóstico médico, el reconocimiento de patrones, el control de robots, el filtrado de señales, la clasificación de datos financieros, la compresión y segmentación de datos, etc.

Tareas como el reconocimiento de patrones, no pueden ser realizadas de forma eficiente ni por las mejores computadoras (una computadora convencional es una máquina que ejecuta una serie de instrucciones de forma secuencial y es capaz de resolver de forma rápida complicadas operaciones lógicas y aritméticas), mientras que el cerebro humano realiza estas tareas con gran eficiencia y relativa facilidad.

La estructura del cerebro es muy diferente a la de una computadora convencional, a diferencia de esta no se compone por un único microprocesador, sino por miles de millones de ellos (de forma simbólica), llamados ***neuronas***, las cuales realizan de modo impreciso y relativamente lento un tipo de cálculo simple.

En la tabla 1.1 se muestra una comparación entre el cerebro humano y una computadora convencional, donde se puede apreciar de forma más clara éstas diferencias.

	CEREBRO	**COMPUTADORA**
Velocidad de Proceso	$= 10^{-2}$ seg. (100Hz)	$= 10^{-9}$ seg. (1000 MHz)
Estilo de Procesamiento	Paralelo	Secuencial
Número de Procesadores	10^{11} - 10^{14}	Pocos
Conexiones	10,000 por procesador	Pocas
Almacenamiento del conocimiento	Distribuido	Direcciones Fijas
Tolerancia a fallos	Amplia	Nula
Tipo de control de proceso	Auto organizado	Centralizado

Tabla 1.1: Cerebro frente a una computadora convencional ***Von Neumann.****

El estudio del cerebro desde el punto de vista de la computación dio origen al surgimiento de los ***sistemas neuronales***, con la idea de tomar las características esenciales de la estructura neuronal de éste y la finalidad de crear sistemas que lo emularan en parte, mediante sistemas electrónicos compuestos por multitud de pequeños procesadores simples, denominados ***neuronas artificiales***.

Cabe mencionar que las computadoras neuronales cuentan con cientos de pequeños microprocesadores que trabajan en paralelo, pero también se puede emular el comportamiento de las ***redes neuronales*** en una computadora convencional mediante software, de igual manera existe gran cantidad de programas de ***redes neuronales*** que funcionan en computadoras personales.

* *Martín del Brío Sanz Molin, 2002.*

En general, el reconocimiento de patrones es la base sobre la que operan las ***redes neuronales artificiales***, adquieren, almacenan y utilizan conocimiento experimental obtenido a partir de ejemplos, el cual no es programado de forma directa, sino que se debe al ajuste de los parámetros de las neuronas mediante un algoritmo de aprendizaje.

Las ***redes neuronales artificiales*** y los ***sistemas expertos***, tienen como objetivo principal representar el conocimiento, pero son opuestos en la forma de obtenerlo. Las ***redes neuronales artificiales*** se acercan más al ***razonamiento inductivo*** (aprendizaje mediante ejemplos), mientras que los ***sistemas expertos*** al ***deductivo*** (obtener fórmulas).

I.4.- REDES NEURONALES E INTELIGENCIA ARTIFICIAL

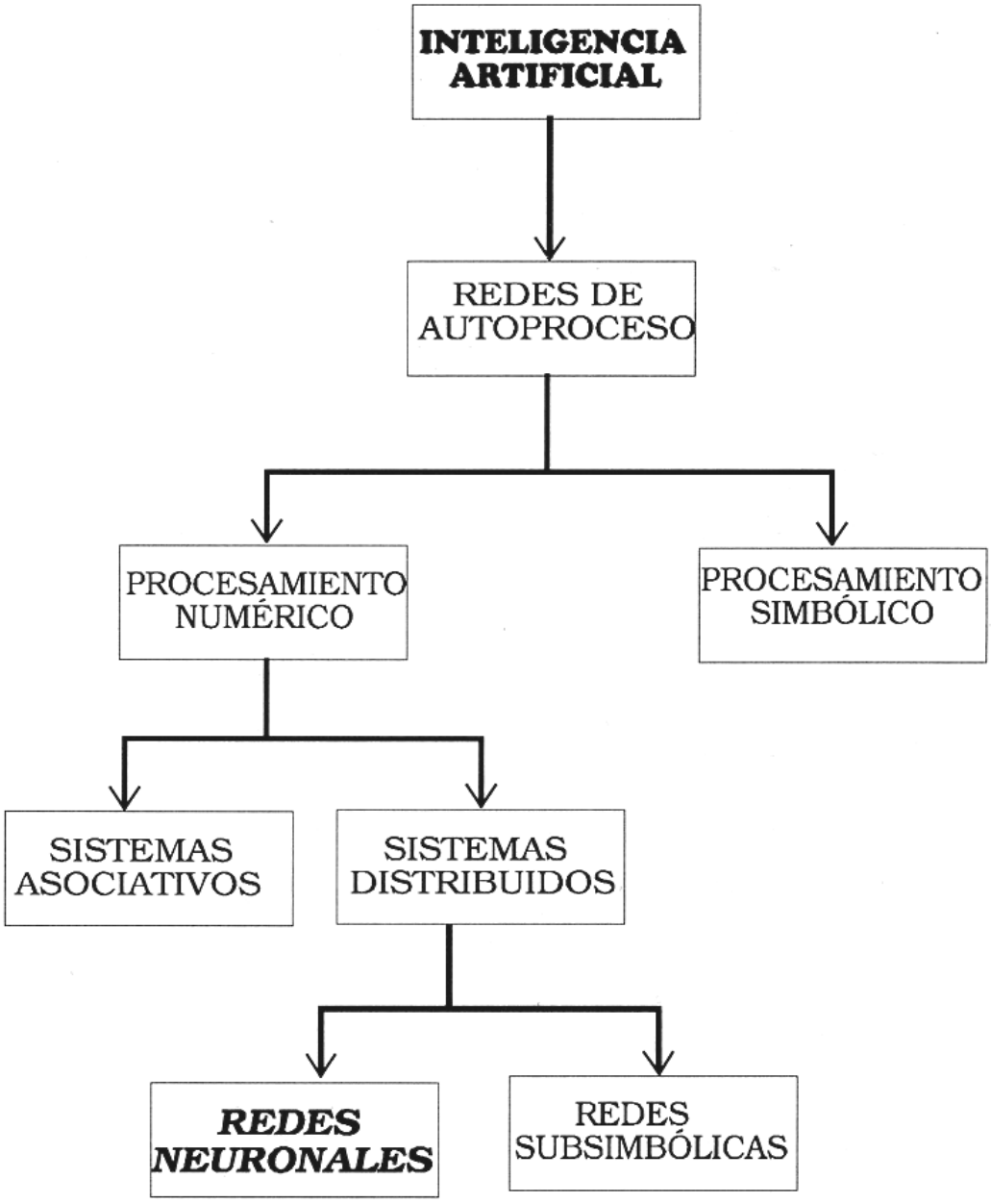


*Figura 1.1: Situación de las **redes neuronales** en el campo de la **inteligencia artificial.** (F. J. López, M. I. Acevedo y M. A. Jaramillo, 1989).*

El campo de la ***inteligencia artificial*** se divide en dos grandes ramas de autoproceso, como indica la figura 1.1, las cuales se describen brevemente a continuación.

Redes de Autoproceso:

Son formadas por nodos en los que hay elementos procesadores de información de cuyas interacciones locales depende el comportamiento del sistema.

a) Procesamiento Numérico:

Son sistemas constituidos por nodos de hardware interconectados formando una red, los cuales reciben la señal de entrada desde el exterior y operan sobre ella. Son conocidos también como ***sistemas conectivistas*** o ***conexionistas***.

Sistemas Distribuidos:

La conexión entre los nodos se realiza de forma global, bajo unas redes de composición.

Sistemas Asociativos:

La conexión se realiza agrupando el sistema de subredes. Estos sistemas

se conocen también como ***redes de redes.***

Redes Subsimbólicas:

En estas redes las agrupaciones locales de los nodos representan conceptos.

b) Procesamiento Simbólico:

Estas redes están constituidas por conceptos (nodos) y por reglas sintácticas (lazos de interconexión). Ambos forman las llamadas ***bases de conocimiento***. La simulación de estas redes es casi exclusivamente software.

En la tabla 1.2, se muestra un resumen de las ***redes neuronales artificiales*** y las diferencias existentes respecto a la computación convencional (máquinas tipo **Von Neumann**) y a la computación simbólica (***IA***), en función de las teorías en las que se basan.

	Computación Convencional	**Computación Simbólica**	**Computación Neuronal**
Basado en:	Arquitectura Von Neumann	Lógica Cognitiva	Neurobiología
Apropiada para:	Algoritmos conocidos	Heurística	Adaptación
Pero no para:	Condiciones difusas	Causalidad desconocida	Cálculos precisos
Memoria:	Precisa, estática	Bases de conocimiento	Distribuida
Construida mediante:	Diseño, programación y prueba	Representación del conocimiento + motor de inferencia	Codificación y aprendizaje
Soporte:	Computadoras secuenciales	Máquinas LISP	Procesadores paralelos

Tabla 1.2: Formas básicas de computación.[*]

[*] ***R. Rubio,*** *1990.*

I.5.- LA NEURONA BIOLÓGICA

A finales del siglo XIX se creía que el sistema nervioso estaba formado por una red continua de fibras nerviosas. Sin embargo en 1888 el científico aragonés **Santiago Ramón y Cajal,** demostró que el sistema nervioso en realidad estaba compuesto por una red de células individuales, las ***neuronas***, ampliamente interconectadas entre sí.

Existen ***neuronas*** de diferentes formas, tamaños y longitudes; siendo estos atributos importantes para determinar las funciones y utilidades de éstas. En la figura 1.2, se muestra la forma general de una ***neurona*** biológica y más adelante se describen los elementos que la conforman.

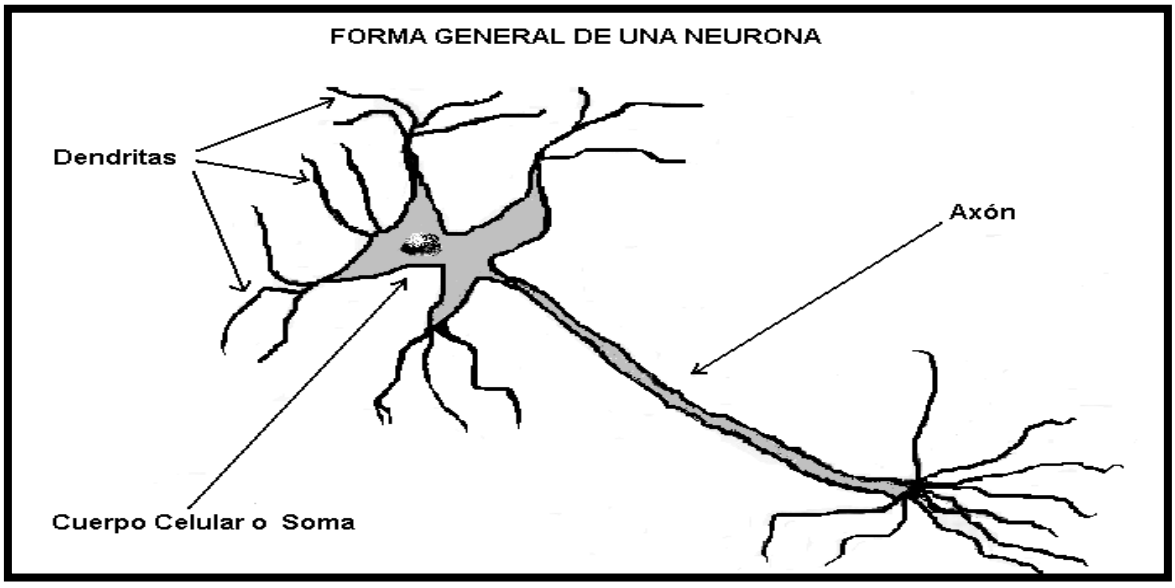


Figura 1.2: *Forma general de una* ***neurona.***

Una ***neurona*** es una célula viva y, como tal, contiene los mismos elementos que forman parte de todas las células biológicas. En general, una ***neurona*** consta de:

1) ***El cuerpo celular o soma***: El cual contiene al ***núcleo***; se encarga de todas las actividades metabólicas de la ***neurona*** y recibe la información de otras ***neuronas*** vecinas a través de las ***conexiones sinápticas*** (las cuáles se abordarán más adelante).

2) ***Las dendritas***: Son ramas de extensión que parten del ***cuerpo celular*** o ***soma*** y tienen ramificaciones. Se encargan de la recepción de señales de las otras células a través de ***conexiones sinápticas.***

3) ***El axón***: Es la rama de "salida" de la ***neurona*** y se utiliza para enviar impulsos o señales a otras células nerviosas. A su vez, el ***axón*** puede producir ramas en torno a su punto de arranque, y con frecuencia se ramifica extensamente cerca de su extremo.

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

I.6.- NATURALEZA BIOELÉCTRICA

Las señales que se utilizan, son de dos tipos distintos de naturaleza: eléctrica y química. La señal es generada por la ***neurona*** y transportada a lo largo del ***axón*** en un impulso eléctrico, mientras que la señal que se transmite entre los ***terminales axónicos*** de una ***neurona*** y las ***dendritas*** de las ***neuronas*** siguientes es de origen químico, y se realiza mediante moléculas de sustancias transmisoras (***neurotransmisores***) que fluyen a través de unos contactos especiales, llamados ***sinapsis*** que tienen la función de receptor y están localizados entre los ***terminales axónicos*** y las ***dendritas*** de la ***neurona***.

Se denomina ***sinapsis*** a la unión entre dos ***neuronas***. Las ***sinapsis*** son direccionales, es decir, la información fluye siempre en un único sentido. En el tipo de ***sinapsis*** más común no existe un contacto físico entre las ***neuronas***, sino que éstas permanecen separadas por un pequeño vacío de unas 0.2 micras. Con relación a la ***sinapsis*** se habla de dos tipos de ***neuronas***:

- Las ***neuronas presinápticas*** que son las que envían las señales
- Las ***neuronas postsinápticas*** que son las que reciben las señales

Existen muchos procesos complejos relacionados con la generación de dichas señales, sin embargo se puede resumir del siguiente modo: la ***neurona*** recibe impulsos procedentes de otras ***neuronas*** (inputs) a través de las ***dendritas***, que están conectadas a las salidas de otras ***neuronas*** por las ***sinapsis***. Las ***sinapsis*** alteran la efectividad con la que la señal es transmitida a

través de un parámetro: el ***peso***. El aprendizaje resulta de la modificación de estos ***pesos***, que unido al procesamiento de información de la ***neurona*** determina el mecanismo básico de la memoria. Algunas ***sinapsis*** permiten pasar a la señal con facilidad, mientras que otras no. El ***soma*** de la ***neurona*** recibe todos estos inputs, y emite una señal de salida (output) si la entrada total supera el valor del umbral. Esta salida se transmite a través del ***axón*** desde donde se propaga mediante diferencias de potencial a las ***dendritas*** de otras ***neuronas***.

I.6.1.- Modos de control

El control de los estados de las unidades puede ser de dos tipos: ***modo asíncrono*** y ***modo síncrono***.

a) Modo asíncrono: En el modo de control ***asíncrono***, las ***neuronas*** evalúan de forma constante su estado, de acuerdo a como va llegando la información y es de forma independiente.

b) Modo síncrono: En este modo, la información también llega de forma continua, pero a diferencia del ***modo asíncrono*** los cambios se realizan simultáneamente como si tuviera un reloj interno que le indicara cuando debe cambiar su estado.

CAPÍTULO II

ESTRUCTURA DE UN SISTEMA NEURONAL ARTIFICIAL

II.1.- ELEMENTOS DE UNA RED NEURONAL ARTIFICIAL

Las ***redes neuronales artificiales*** son modelos que intentan reproducir el comportamiento del cerebro, con la intención de construir sistemas de procesamiento de la información paralelos, distribuidos y adaptativos, que puedan presentar un cierto comportamiento "inteligente". Los tres conceptos clave, de los sistemas nerviosos que se pretenden emular son:

a) Procesamiento Paralelo: Una computadora puede realizar cálculos mucho más rápido que un ser humano, pero no es capaz de desarrollar procesos que este realiza de forma relativamente sencilla, como el reconocer un rostro, comprender una imagen, etc. Esto se debe a que el cerebro humano, para poder realizarlo utiliza simultáneamente millones de neuronas.

b) Memoria Distribuida*:* En una computadora la información ocupa posiciones de memoria bien definidas, mientras que en los ***sistemas neuronales*** se encuentra distribuida por las ***sinapsis*** de la red; en caso de resultar dañada una ***sinapsis***, no se pierde más que una muy pequeña parte de la información. Además, los ***sistemas neuronales biológicos*** son redundantes, de modo que muchas neuronas y ***sinapsis*** pueden realizar un papel similar, por lo tanto el sistema resulta tolerante a fallos.

c) Adaptabilidad al entorno: Las ***redes neuronales artificiales*** aprenden de la experiencia, pudiendo generalizar conceptos a partir de casos particulares además de adaptarse fácilmente al entorno modificando sus

sinapsis (entre otros mecanismos). Llamando a esta propiedad ***generalización a partir de ejemplos.***

Cada neurona realiza una función matemática. Las neuronas se agrupan en capas, constituyendo una ***red neuronal***. Una determinada ***red neuronal*** es creada y entrenada para llevar a cabo una labor específica. Finalmente, una o varias redes, más las interfaces con el entorno, conforman el ***sistema global***; como se puede observar en la figura 2.1.

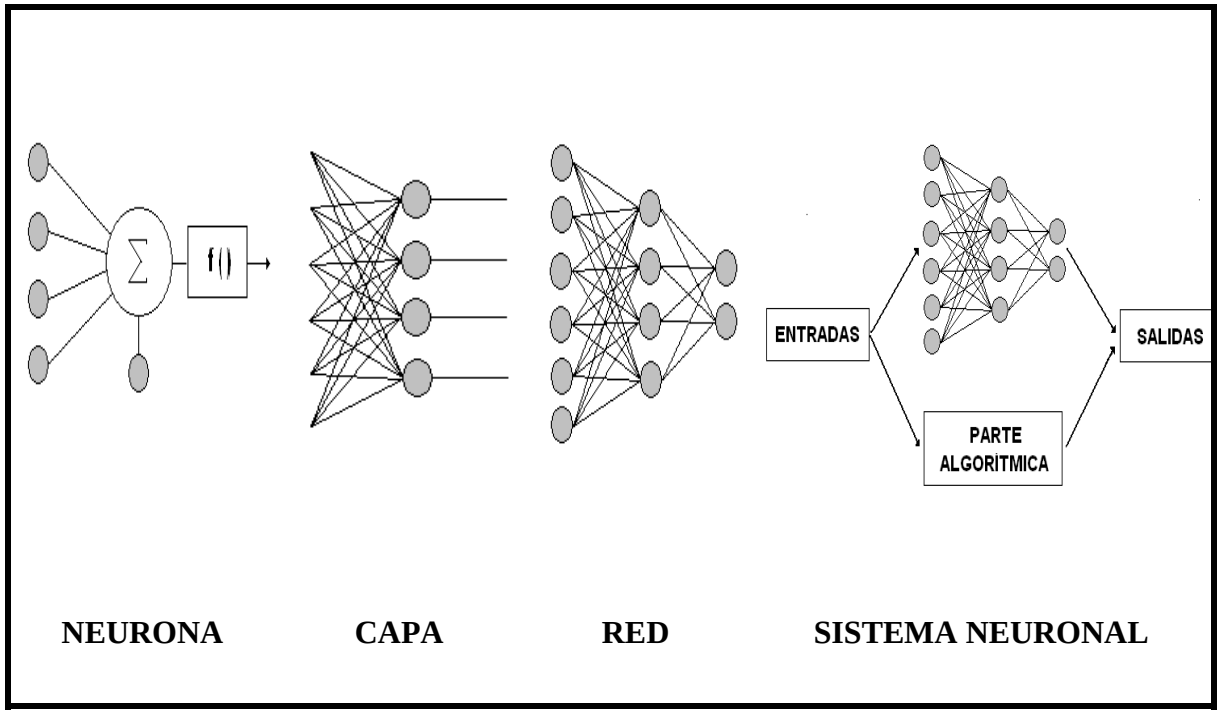


*Figura 2.1: Estructura jerárquica de un sistema basado en **redes neuronales artificiales.***

En las ***redes neuronales biológicas***, las neuronas corresponden a los elementos de proceso. Las interconexiones se realizan por medio de las ramas

de salida (***axones***) que producen un número variable de conexiones (***sinapsis***) con otras neuronas o con otras partes como músculos y glándulas. Las ***redes neuronales*** son sistemas de elementos simples de proceso muy interconectados.

La compleja operación es el resultado de abundantes lazos de realimentación junto con no linealidades de los elementos de proceso y cambios adaptativos de sus parámetros, que pueden llegar a definir fenómenos dinámicos muy complicados.

Los ***modelos neuronales*** se diferencian en la función que incorpora la neurona, su organización y forma de las conexiones. La mayoría de los ***modelos neuronales*** tienen un equivalente tradicional conformado por los siguientes elementos.

- ***Unidades de procesamiento***
- ***Estado de activación de cada neurona***
- ***Patrón de conectividad entre neuronas***
- ***Regla de propagación***
- ***Función de transferencia***
- ***Regla de activación***
- ***Regla de aprendizaje***

II.2.- UNIDAD DE PROCESAMIENTO (LA NEURONA ARTIFICIAL)

Un modelo de ***red neuronal*** consta de dispositivos elementales de proceso: las ***neuronas***. A partir de ellas se pueden generar representaciones específicas, de modo que un estado conjunto de ellas puede representar una letra, un número o cualquier objeto.

Se denomina ***procesador elemental*** *o* ***neurona artificial*** a un dispositivo simple de cálculo que tiene como función recibir las entradas procedentes del exterior o de otras neuronas, y calcular un valor de salida, el cual es enviado a todas las células restantes.

Básicamente se pueden encontrar tres tipos de neuronas:

1) Las que reciben ***estímulos externos***, es decir aquellas relacionadas con el aparato sensorial, que toman la información de entrada.

2) Esta información es transmitida a ciertos elementos internos los cuales la procesan. Las neuronas y ***sinapsis*** correspondientes a este segundo nivel son las que generan cualquier tipo de representación interna de la información y debido a que no tienen relación directa con la información de entrada, ni con la de salida se denominan ***unidades ocultas***.

3) Ya finalizado el procesado de la información, esta llega a las ***unidades de salida***, cuya función es dar la respuesta del sistema.

La ***neurona artificial*** pretende imitar las características esenciales de las neuronas biológicas. Cada neurona *i*-ésima está caracterizada en cualquier instante por un valor numérico denominado ***valor*** o ***estado de activación*** *a(t)*, asociado a cada unidad; la ***función de salida*** f_i, que es la encargada de transformar el estado actual de activación en una ***señal de salida*** y_i. Ésta señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red, la señal es modificada en estos canales de acuerdo con la ***sinapsis*** (el ***peso sináptico*** $w_{ji,}$ el cual se describe más adelante) asociada a cada uno de ellos. Las señales que han llegado a la unidad *j*-ésima se combinan, generando así la ***entrada total*** Net_j.

$$Net_j = \sum_i y_i w_{ji}$$

La ***función de activación*** *F*, determina el nuevo ***estado de activación*** $a_j(t+1)$ de la neurona, a partir de la entrada total calculada y el ***estado de activación*** anterior $a_j(t)$. La figura 2.2, muestra el modelo de una ***neurona artificial estándar***.

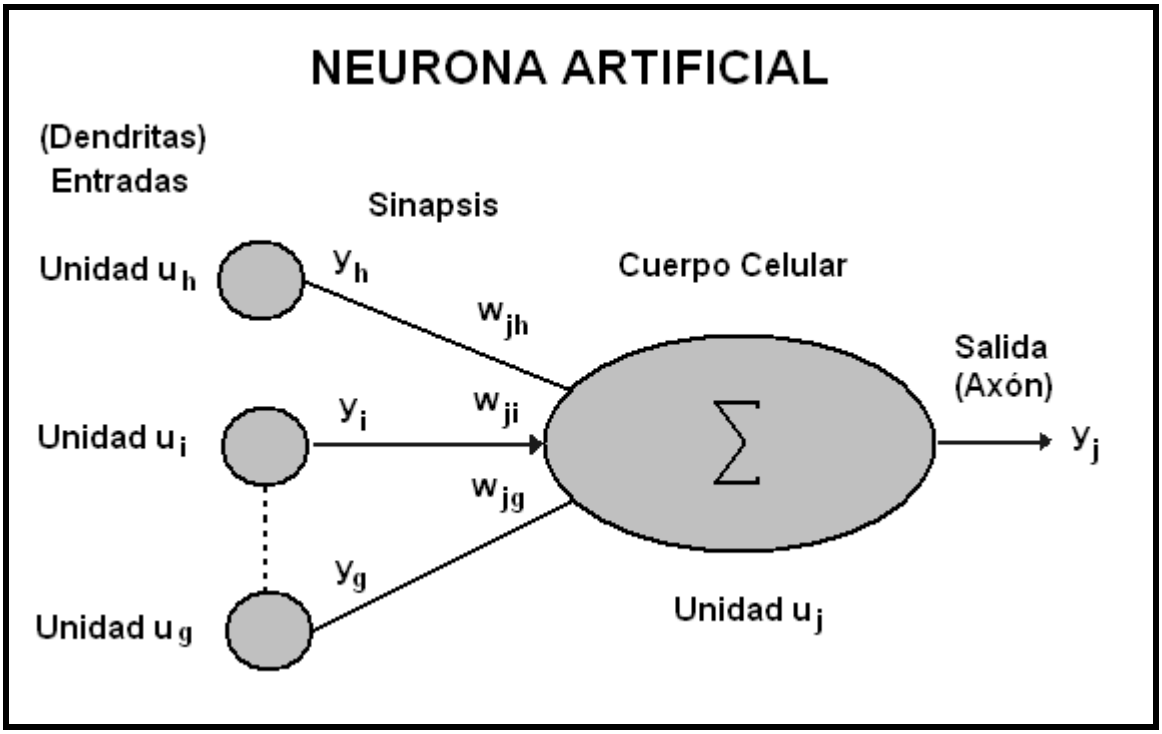


Figura 2.2: La neurona artificial.

Peso sináptico:

El ***peso sináptico*** w_{ji} define en este caso la intensidad de interacción entre la ***neurona presináptica*** *j* y la ***postsináptica*** *i*. Dada una entrada positiva, si el ***peso*** es positivo tenderá a excitar a la ***neurona postsináptica***, si el ***peso*** es negativo tenderá a inhibirla. Así se habla de ***sinapsis excitadoras*** (de ***peso positivo***) e ***inhibidoras*** (de ***peso negativo***).

II.3.- ESTADO DE ACTIVACIÓN

El ***estado de activación*** del conjunto de ***unidades de procesamiento*** se especifica por un vector de *N* números reales *A(t),* el cual representa los estados del sistema en un tiempo *t*, donde cada elemento del vector representa la activación de una unidad en dicho tiempo. Por lo que la activación de una unidad U_i en el tiempo *t* se designa por $a_i(t)$, es decir:

$$A(t) = (a_1(t), a_2(t),\ldots a_i(t),\ldots, a_N(t))$$

Generalmente hay dos posibles ***estados de activación***, ***reposo*** y ***excitado***, a los cuales se les asigna un valor. Estos valores pueden ser continuos o discretos; además pueden ser limitados o ilimitados.

En el caso de ser valores discretos, se suele tomar un pequeño conjunto de valores o bien valores binarios. En notación binaria, un *1* indica un ***estado activo***, y se caracteriza por la emisión de un impulso de la neurona; un *0* indica un ***estado pasivo***, es decir que la neurona se encuentra en ***reposo***.

Para otros modelos se considera un conjunto continuo de ***estados de activación***, en vez de sólo dos estados; en tal caso es asignado un valor entre *[0,1]* o en el intervalo *[-1,1]*, por lo general siguiendo una ***función sigmoidal***.

Los ***estados de activación*** de las neuronas dependen de dos factores:

a) De los ***mecanismos de interacción*** entre las neuronas, ya que el efecto que produce una neurona sobre otra es proporcional a la fuerza, peso o magnitud de la conexión entre ambas.

b) A ***la señal*** que cada una de las neuronas envía a sus aledañas, ya que ésta dependerá de su propio ***estado de activación***.

II.4.- FUNCIÓN DE SALIDA O DE TRANSFERENCIA

Las neuronas que forman la ***red neuronal artificial*** se encuentran interconectadas unas a otras; cada neurona transmite señales a aquellas que se encuentran conectadas con su salida. Es decir a cada unidad U_i hay asociada una ***función de salida*** $f_i(a_i(t))$, que transforma el ***estado actual de activación*** $a_i(t)$ en una ***señal de salida*** $y_i(t)$, es decir:

$$y_i(t) = f_i(a_i(t))$$

El vector que contiene las salidas de todas las neuronas en un instante t esta dado por:

$$Y(t) = (f_1(a_1(t)), f_2(a_2(t))... \ f_i(a_i(t)),..., f_N(a_N(t)))$$

Para algunos modelos, esta salida es igual al ***nivel de activación*** de la unidad, en dicho caso f_i es la ***función identidad*** $f_i(a_i(t)) = a_i(t)$, generalmente es de tipo ***sigmoidal***, y suele ser la misma para todas las unidades.

Las funciones de transferencia más usuales, que determinan los distintos tipos de ***neuronas*** son:

a) Función escalón

b) Función lineal o mixta

c) Función sigmoidal

d) Función gaussiana

Estas funciones se describen a continuación:

a) Neurona de función escalón:

La ***función escalón*** se asocia a neuronas binarias en las cuales cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es *1*; si es menor, la activación es *0 (ó –1)*. Las redes formadas por este tipo de neuronas son fáciles de implementar en hardware, pero sus capacidades están limitadas. La figura 2.3 muestra dicha función.

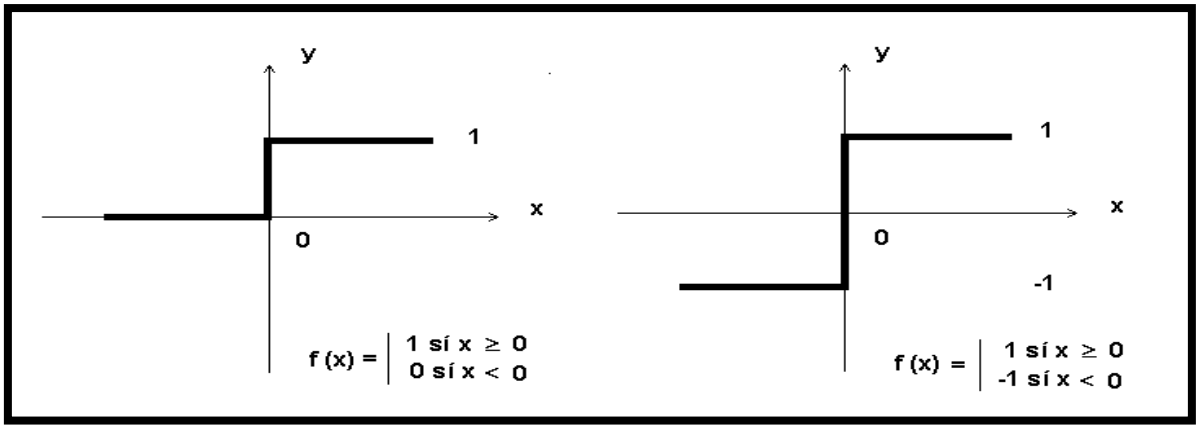


Figura 2.3: Función de transferencia escalón.

b) Neurona de función lineal o mixta:

La ***función lineal*** o ***mixta*** corresponde a la función *f(x)=x*. En las neuronas con ***función mixta***, si la suma de las señales de entrada es menor que un límite inferior, la activación se define como *0* (ó *–1*). Si dicha suma es mayor o igual que el límite superior, entonces la activación es *1*. Si la suma de

entrada está comprendida entre ambos límites, la activación se define como una función lineal de suma de las señales de entrada. En la figura 2.4. se muestran dichas funciones.

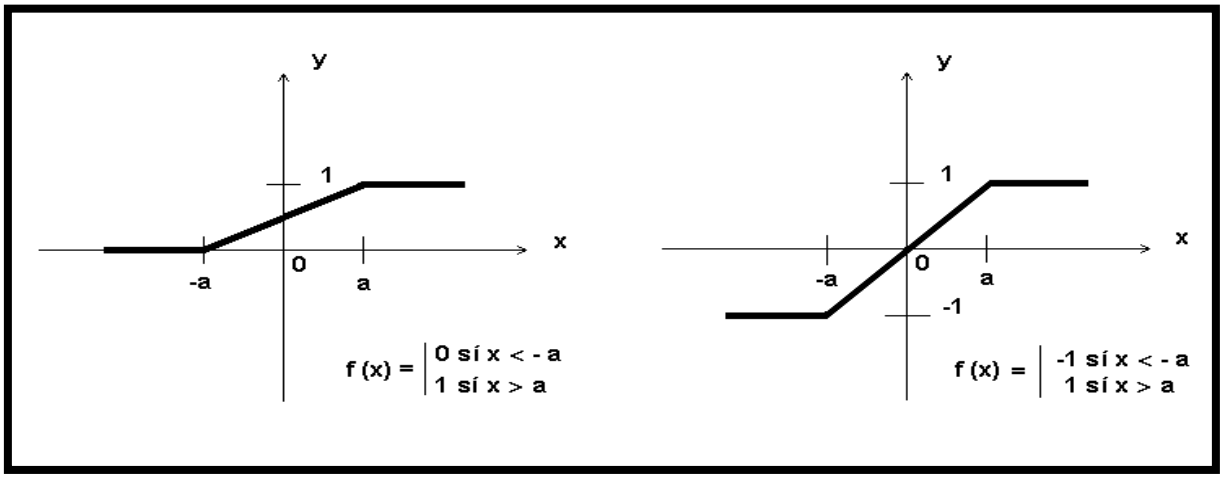


Figura 2.4: Funciones de activación mixtas.

c) Neurona de función continua (sigmoidal):

La ***función continua*** tiene como característica que su derivada es siempre positiva y cercana a cero para valores grandes positivos o negativos, además, de tomar su máximo valor cuando x es cero. Esto resulta particularmente útil para definir métodos de aprendizaje en los cuales se usan derivadas. La figura 2.5 muestra las gráficas correspondientes a dicha función.

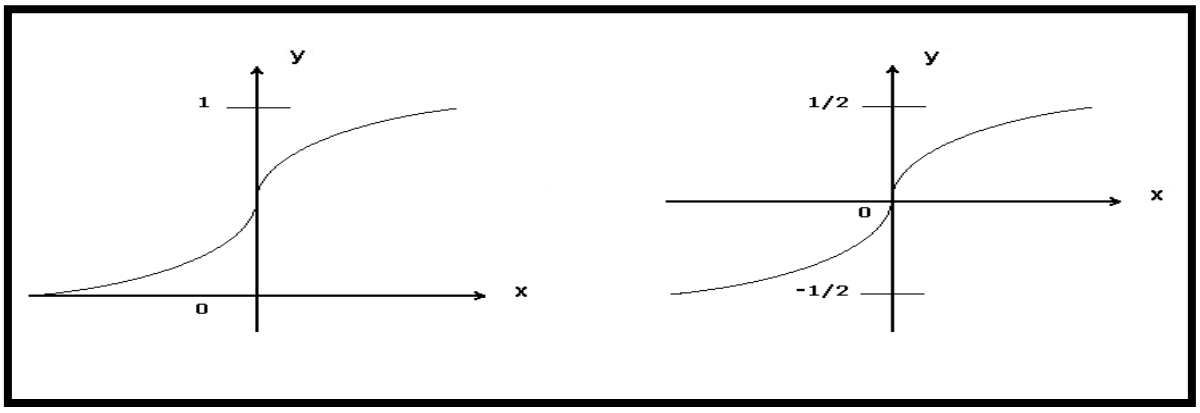


Figura 2.5: Funciones de activación continuas.

e) Neurona de función de transferencia gaussiana:

En estas funciones, los centros y anchura pueden ser modificados, lo cual las hace más adaptativas que las funciones continuas. Mapeos que suelen requerir dos niveles ocultos con neuronas de transferencia sigmoidales, algunas veces se pueden realizar con un solo nivel empleando neuronas de transferencia ***gaussiana***. La figura 2.6 muestra la gráfica de una función de transferencia ***gaussiana***.

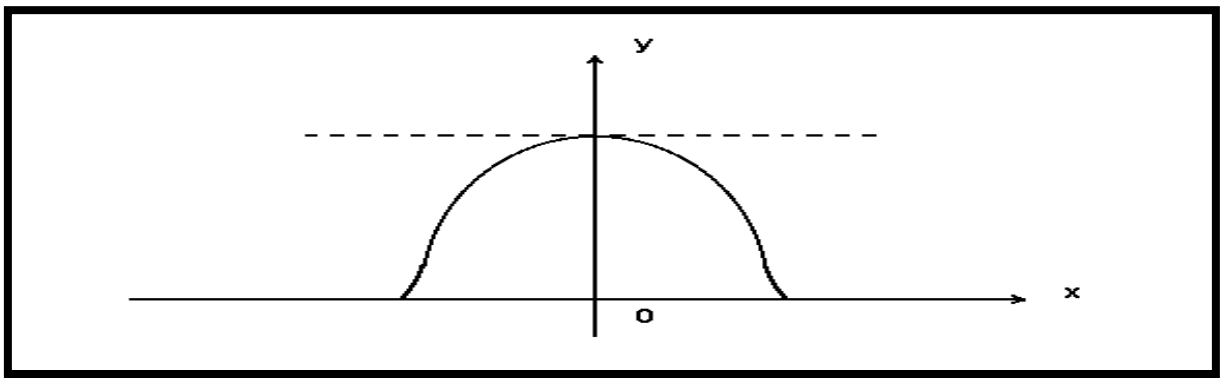


Figura 2.6: Función de transferencia gaussiana.

II.5.- CONEXIONES ENTRE NEURONAS

La ***red neuronal artificial***, adquiere conocimiento debido al ***peso*** asociado a las conexiones que unen a las neuronas que conforman la red; éste proceso se lleva a cabo de la siguiente manera:

Tomemos el valor y_i como el valor de salida de una neurona *i*, en un instante dado. Una neurona recibe un conjunto de señales, las cuales le proporcionan información del ***estado de activación*** de todas las demás neuronas con las que se encuentra conectada. La conexión (***sinapsis***) entre la neurona *i* y la neurona *j* está ponderada por un ***peso*** w_{ji}. Por simplificación se considera al efecto de cada señal aditivo, de manera que la entrada neta que recibe una neurona (***potencial postsináptico***) net_j, esta dada por el producto de cada señal individual, por el valor de la ***sinapsis*** que conecta ambas neuronas, es decir:

$$net_j = \sum_{i}^{N} w_{ji} * y_i$$

Esta ecuación es conocida como ***regla de propagación***, y muestra el procedimiento seguido en la combinación de los valores de entrada a una unidad con los ***pesos*** de las conexiones que llegan a ella.

Se suele utilizar una matriz *W* con todos los ***pesos*** w_{ji}, los cuales reflejan la influencia que tiene la neurona *j* sobre la neurona *i*. Si w_{ji} es positivo, significa que siempre que la neurona *i* esté activada, la neurona *j* recibirá una señal proveniente de *i* que tenderá a activarla. Para el caso en que w_{ji} sea negativo, la ***sinapsis*** será ***inhibidora***; sí w_{ji} es cero, significa que no existe conexión entre ambas neuronas.

II.6.- FUNCIÓN O REGLA DE ACTIVACIÓN

La ***función*** o ***regla de activación*** *F* combina las entradas con el estado actual de la neurona, produciendo un nuevo ***estado de activación***, a partir del estado *(a_i)* que existía y la combinación de las entradas con los ***pesos*** de las conexiones *(net_i)*.

Dado el ***estado de activación*** $a_i(t)$ de la neurona U_i y la entrada total que llega a ella, Net_i, el ***estado de activación*** siguiente, $a_i(t+1)$ sé obtiene aplicando la ***función de activación*** *F*:

$$a_i(t+1) = F(a_i(t), Net_i)$$

En la mayor parte de los casos, *F* es la ***función identidad***, por lo que el ***estado de activación*** de la neurona en *t+1* coincide con el *Net* de la misma en el tiempo *t*. Por lo que en este caso, la salida de la neurona *i* (y_i) será:

$$y_i(t+1) = f(Net_i) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t)\right)$$

Normalmente la ***función de activación*** no esta centrada en el origen del eje que representa el valor de la entrada neta, sino que existe cierto desplazamiento debido a las características internas de la propia neurona, y

que no es igual en todas ellas. Este valor se denota como θ_i, el cual representa el umbral de activación de la neurona *i*.

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

En el caso de las neuronas de respuesta todo-nada (*1 ó 0*), este parámetro representa el umbral de disparo de la neurona, es decir, el nivel mínimo que debe alcanzar el ***potencial postsináptico*** para que la neurona se dispare o active.

II.7.- REGLA DE APRENDIZAJE

El ***aprendizaje*** puede ser comprendido como la modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos. En el cerebro humano el conocimiento se encuentra en las ***sinapsis***, mientras que en las ***redes neuronales artificiales***, el conocimiento se encuentra representado en los ***pesos*** de las conexiones entre neuronas.

Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. Puede decirse que se aprende modificando los valores de los ***pesos*** de la red.

Un aspecto importante respecto al aprendizaje de las ***redes neuronales*** es el conocer cómo es que se modifican estos valores; es decir, cuales son los criterios seguidos para cambiar los valores asignados a las conexiones cuando se pretende que la red aprenda una nueva información.

Estos criterios determinan lo que se conoce como la ***regla de aprendizaje de la red***. (En el capítulo III se describen con más detalles estos criterios de aprendizaje).

II.8.- FORMAS DE CONEXIÓN ENTRE NEURONAS

Dentro de una red, la distribución de neuronas se realiza formando ***niveles*** o ***capas*** de un número determinado de neuronas cada una. Se dice que una red es totalmente conectada si todas las salidas desde un nivel llegan a todos y cada uno de los nodos del nivel siguiente.

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 ser una entrada de sí mismo en una conexión auto recurrente.

Sí ninguna salida de las neuronas de una capa es entrada de neuronas del mismo ***nivel*** o de ***niveles*** precedentes, se dice que la red tiene ***propagación hacia adelante***. En caso contrario se dice que la red es ***de propagación hacia atrás***. Las redes de ***propagación hacia atrás*** que tienen lazos cerrados se dice que son ***sistemas recurrentes***. (Estos conceptos se describen en el capítulo III)

CAPÍTULO III

TOPOLOGÍAS Y ALGORITMOS DE APRENDIZAJE DE LAS REDES NEURONALES ARTIFICIALES

III.1.- TOPOLOGÍA DE LAS REDES NEURONALES

Se denomina ***topología*** o ***arquitectura*** de red a la organización y disposición de las neuronas en la red formando capas más o menos alejadas de la entrada y salida de la red. Así, los parámetros fundamentales de la red son:

- ***El número de capas* (*monocapa* o *multicapa*).**

- ***El número de neuronas por capa.***

- ***El grado de conectividad y el tipo de conexiones entre neuronas* (*unidireccionales* o *recurrentes*).**

III.1.1.- Redes Neuronales Monocapa

Se utilizan típicamente en tareas relacionadas con la autoasociación; por ejemplo, para regenerar información de entrada que se presentan a la red incompleta o distorsionada.

La tabla 3.1, muestra las características topológicas de los modelos de redes monocapa más conocidos.

TIPOS DE CONEXIONES		**MODELO DE RED**
CONEXIONES LATERALES EXPLÍCITAS	CONEXIONES AUTORRECURRENTES	BRAIN-SATATE-IN-A-BOX
		ADDITIVE GROSSBERG (AG)
		SHUNTING GROSSBERG (SG)
		OPTIMAL LINEAR ASOCIATIVE MEMORY
	NO AUTORRECURRENTES	HOPFIELD
		BOLTZMANN MACHINE
		CAUCHY MACHINE
CROSBAR		LEARNING MATRIX (LM)

Tabla 3.1: Redes Neuronales monocapa.[*]

III.1.2.- Redes Neuronales Multicapa

Se conocen como ***redes multicapa*** a aquellas que disponen de conjuntos de neuronas agrupadas en varios niveles o capas. Las cuales se pueden dividir en dos tipos:

a) Las redes con conexión hacia adelante (feedforward)

Se denomina conexiones ***hacia adelante*** o ***feedforward*** cuando todas las neuronas de una capa reciben señales de entrada de otra capa anterior, más cercana a la entrada de la red, y envían señales de salida a una capa posterior, más cercana a la salida de la red. Es decir, en estas redes todas las señales neuronales se propagan ***hacia adelante***, a través de las capas de la red.

Las redes ***feedforward*** son especialmente útiles en reconocimiento o clasificación de patrones.

b) Las redes con conexiones hacia adelante y hacia atrás (feedforward/feedback)

En estas redes la información circula tanto ***hacia adelante*** como ***hacia atrás*** durante el funcionamiento de la red, debido a la existencia de conexiones ***feedforward*** y ***feedback*** entre las neuronas.

---

[*] ***José R. Hilera*** *y* ***Víctor J. Martínez,*** *1995.*

Las redes ***feedforward/feedback*** suelen ser ***bicapa*** (dos capas), lo que implica que existen dos conjuntos de pesos, los correspondientes a las conexiones ***feedforward*** de la capa de entrada hacia la de salida y los de las conexiones ***feedback*** de la salida a la entrada. No es necesario que coincidan los valores de los pesos.

La estructura ***bicapa*** tiene varias aplicaciones aún que destacan en la asociación de una información o patrón de entrada con otra información o patrón de salida.

Existen también algunas redes dentro de este género que tienen conexiones laterales entre neuronas de la misma capa. Diseñadas como conexiones excitadoras (con ***peso positivo***), permitiendo la cooperación entre neuronas, o como inhibidoras (con ***peso negativo***), estableciéndose una competición entre las neuronas correspondientes.

La tabla 3.2, muestra las características topológicas de las redes ***multicapa*** con conexiones ***hacia adelante*** y ***hacia adelante/atrás***.

No. DE CAPAS	TIPO DE CONEXIONES		MODELO DE RED
DOS CAPAS	CONEXIONES HACIA ADELANTE (FEEDFORWARD)		ADALINE/MADALINE
			PERCEPTRON
			LINEAR/ASSOC REWAR. PENALTY
			LINEAR ASSOCIATIVE MEMORY
			OPTIMAL LINEAR ASSOC. MEM.
			DRIVE-REINFORCEMENT (DR)
		CONEXIONES LATERALES IMPLÍCITAS Y AUTORRECURRENTES	LEARNING VECTOR QUANTIZER
			TOPOLOGY PRESERVING MAP (TPM)
	CONEXIONES ADELANTE/ATRÁS (FEEDFORWARD/FEEDBACK)	SIN CONEXIONES LATERALES	BIDIRECTIONAL ASSOC. MEM. (BAM)
			ADAPTIVE BAM.
			TEMPORAL ASSOC. MEMORY (TAM)
			FUZZY ASSOCIATIVE MEMORY (FAM)
		CON CONEXIONES LATERALES Y AUTORRECURRENTES	COMPETITIVE ADAPTIVE BAM
			ADAPTIVE RESONANCE THEORY (ART)
TRES	CONEXIONES HACIA ADELANTE (FEEDFORWARD)	SIN CONEXIONES LATERALES	ADAPTIVE HEURISTIC CRITIC (AHC)
			BOLTZMANN/CAUCHY MACHINE
		CON CONEXIONES LATERALES	COUNTERPROPAGATION
			BOLTZMANN/CAUCHY MACHINE
	CONEXIONES ADELANTE/ATRÁS Y LATERALES		BOLTZMANN/CAUCHY MACHINE
N	CONEXIONES HACIA DELANTE		BACK-PROPAGATION (BPN)
	FEEDFORWARD-FEEDBACK (JERARQUÍA DE NIVELES DE CAPAS BIDIMENCIONALES)		COGNITRON/NEOCOGNITRON

La tabla 3.2: Redes multicapa más conocidas. [*]

III.2.- ALGORITMOS NEURONALES

Los ***modelos neuronales*** utilizan varios algoritmos de estimación, aprendizaje o entrenamiento para encontrar los valores de los parámetros del modelo, denominados ***pesos sinápticos***.

El entrenamiento se realiza mediante patrones- ejemplo; siendo dos los tipos de aprendizaje: ***supervisado*** y ***no supervisado***.

Su diferencia fundamental radica en la existencia o no de un agente externo (***supervisor***) que controle el proceso de aprendizaje de la red

III.2.1.- Aprendizaje Supervisado

Se caracteriza porque el proceso de aprendizaje es realizado mediante un entrenamiento controlado por un agente externo (*supervisor*), el cual determina la respuesta que debería generar la red a partir de una entrada determinada. En el caso de que no coincidan las salidas, el supervisor modifica los ***pesos*** de las conexiones hasta que éstas se aproximen. Existen tres tipos de aprendizaje supervisado y son:

1) Aprendizaje por corrección de error
2) Aprendizaje por refuerzo
3) Aprendizaje estocástico

---

[*] ***José R. Hilera*** *y* ***Víctor J. Martínez,*** *1995.*

1) Aprendizaje por corrección de error:

Se lleva a cabo ajustando los ***pesos*** de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida; es decir, en función del error cometido.

Un ejemplo podría ser el siguiente algoritmo simple de aprendizaje por corrección de error:

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

Donde:

Δw_{ji}: Es la variación en el ***peso*** de la conexión entre las neuronas *i* y *j* ($\Delta w_{ji} = \Delta w_{ji}^{actual} - \Delta w_{ji}^{anterior}$).

y_i: Es el valor de salida de la neurona *i*.

d_j: Es el valor de salida deseado para la neurona *j*.

y_j: Es el valor de salida obtenido en la neurona *j*.

α: Es el factor de aprendizaje $(0 < \alpha \leq 1)$.

La figura 3.1, muestra de forma simbólica la expresión anterior.

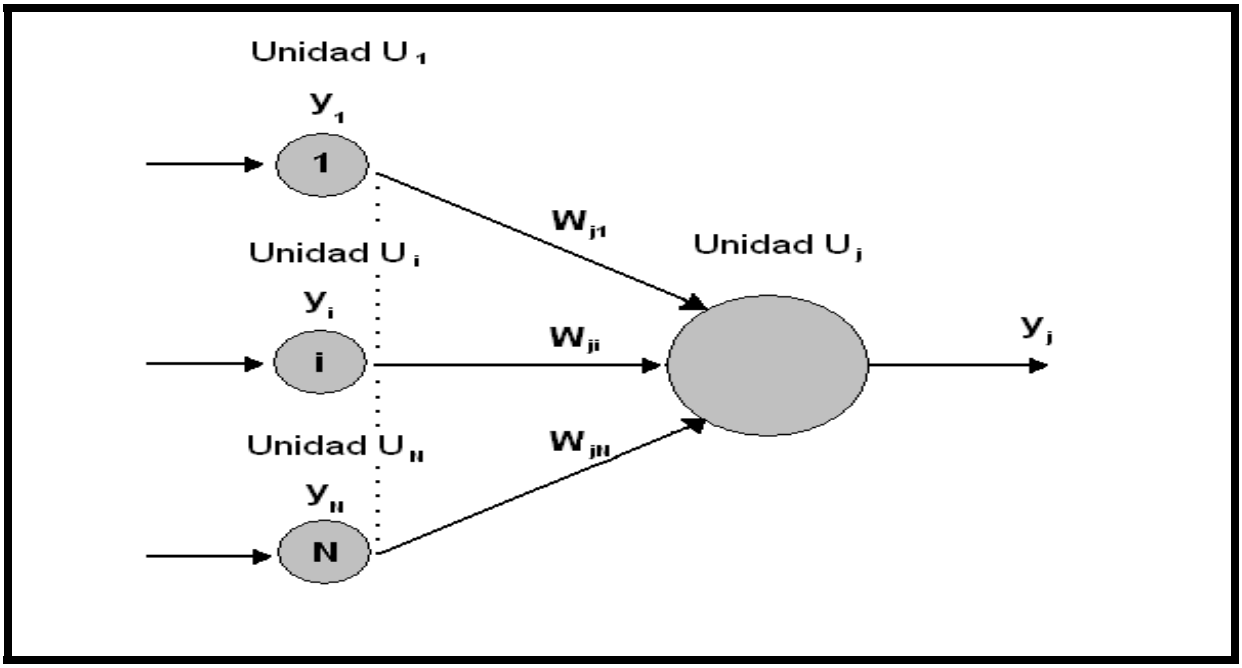


Figura 3.1. Representación simbólica de la expresión $\Delta w_{ji} = \alpha y_i (d_j - y_j)$.

2) Aprendizaje por refuerzo:

Se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado; es decir, de no indicar durante el entrenamiento de forma exacta la salida que se desea que proporcione la red ante una determinada entrada.

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 los ***pesos*** se ajustan basándose en un mecanismo de probabilidades.

3) Aprendizaje estocástico:

El aprendizaje estocástico consiste básicamente en realizar cambios aleatorios en los valores de los ***pesos*** de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

III.2.2.- Aprendizaje no Supervisado

Si el entrenamiento es ***no supervisado***, únicamente se debe suministrar a la red los datos de entrada para que esta extraiga los rasgos característicos esenciales; es decir, ya que 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, deben encontrar las características, regularidades, o categorías que se puedan establecer entre los datos que se presentan en su entrada. En estas redes, la respuesta generada ante una entrada concreta depende de varios factores, como son su estructura y el algoritmo de aprendizaje empleado.

Por ejemplo 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. Otro caso, sería el realizar un establecimiento de categorías, indicando la red a la salida a que categoría pertenece la información presentada a la entrada, a partir de correlaciones entre las informaciones presentadas.

En general se suelen considerar dos tipos de algoritmos de aprendizaje ***no supervisado***:

1) ***Aprendizaje hebbiano***
2) ***Aprendizaje competitivo y cooperativo***

Estos aprendizajes se describen a continuación con mayor detalle:

1) Aprendizaje hebbiano:

El ***aprendizaje hebbiano*** consiste básicamente en el ajuste de los ***pesos*** de las conexiones de acuerdo con la correlación (multiplicación en el caso de los valores binarios *+1* y *–1*) de los valores de activación (salidas) de las dos neuronas conectadas. La siguiente expresión es un ejemplo de regla de ***aprendizaje no supervisado*** en el que si las dos unidades son activadas (positivas), se produce un reforzamiento de la conexión. Por lo contrario, cuando una es ***activa*** y la otra ***pasiva*** (negativa), se produce un debilitamiento de la conexión.

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

***2) Aprendizaje competitivo y cooperativo*:**

Este tipo de aprendizaje tiene por objetivo categorizar los datos que se introducen en la red de la siguiente forma:

Cuando se presenta a la red cierta información de entrada, sólo una de las neuronas de salida de la red, o una por cierto grupo de neuronas, se activa (alcanza su valor de respuesta máximo). De manera que, las neuronas compiten por activarse, teniendo finalmente una, o una por grupo, como neurona vencedora, quedando anuladas el resto, que son forzadas a sus valores de respuesta mínimos.

CAPÍTULO IV

EL PERCEPTRON

IV.1.- ANTECEDENTES

Resulta muy difícil realizar una clasificación completa de todos los modelos neuronales desarrollados por la comunidad científica hasta el momento. Es por ello que a efectos de centrar el tema, sólo se describe el modelo de ***red neuronal perceptron.***

El ***perceptron*** fue el primer modelo de ***red neuronal artificial***, desarrollado por el psicólogo **Frank Rosenblatt** en 1958. Su intención era ilustrar algunas de las propiedades fundamentales de los sistemas inteligentes en general, sin entrar en mayores detalles con respecto a ciertas condiciones especiales, y en muchas veces desconocidas, válidas para organismos biológicos concretos.

Rosenblatt creía que la conectividad que se desarrolla en las redes biológicas tiene un elevado porcentaje de aleatoriedad, por ello opinaba que la herramienta más apropiada para su análisis era la teoría de probabilidades; más adelante desarrolló una teoría de separabilidad estadística, la cual utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias.

Mediante ciertas investigaciones se pudo demostrar que el ***perceptron*** era capaz de clasificar patrones correctamente, además de responder de manera congruente a patrones aleatorios, pero a medida que aumentaba el número de patrones que intentaba aprender su precisión iba disminuyendo.

Se crearon grandes expectativas sobre las aplicaciones del ***perceptron***, que posteriormente se tornaron en gran decepción cuando en 1969 **Marvin Minsky** y **Seymour Papert** publicaron su libro: ***"Perceptrons: An introduction to Computational Geometry"***. En él, se presentaba un análisis detallado del ***perceptron***, en términos de sus capacidades y limitaciones, en especial en cuanto a las restricciones que existen para los problemas que una red tipo ***perceptron*** puede resolver; como su incapacidad para solucionar problemas que no sean linealmente separables. A pesar de estas limitaciones, el ***perceptron*** sigue siendo en la actualidad una red de gran importancia, pues con base a su estructura se han desarrollado otros modelos de ***red neuronal*** como la ***red Adaline*** y las ***redes multicapa***.

IV.2.- ESTRUCTURA

El ***perceptron*** es un dispositivo determinista constituido por un conjunto de ***unidades de procesamiento*** (***neuronas***) organizadas en dos capas.
La capa de entrada esta formada por varias neuronas lineales, las cuales actúan como sensores del ambiente y capturan los patrones de entrada a la red, mientras que la capa de salida se encuentra constituida por una única neurona. Un ***perceptron***, con estas características tiene la capacidad de decidir a cual de las dos clases *(A o B)*, que es capaz de reconocer, pertenece una entrada presentada a la red, de la siguiente forma:

La neurona de salida del ***perceptron*** realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo ***escalón***. La ***regla de decisión*** consiste en responder *+1* si el patrón presentado pertenece a la clase *A*, o *–1* si el patrón pertenece a la clase *B*. La salida dependerá de la entrada neta (suma de las entrada x_i ponderadas) y del valor umbral θ.

En la figura 4.1, se muestra el modelo de un ***perceptron*** con *N* variables de entrada.

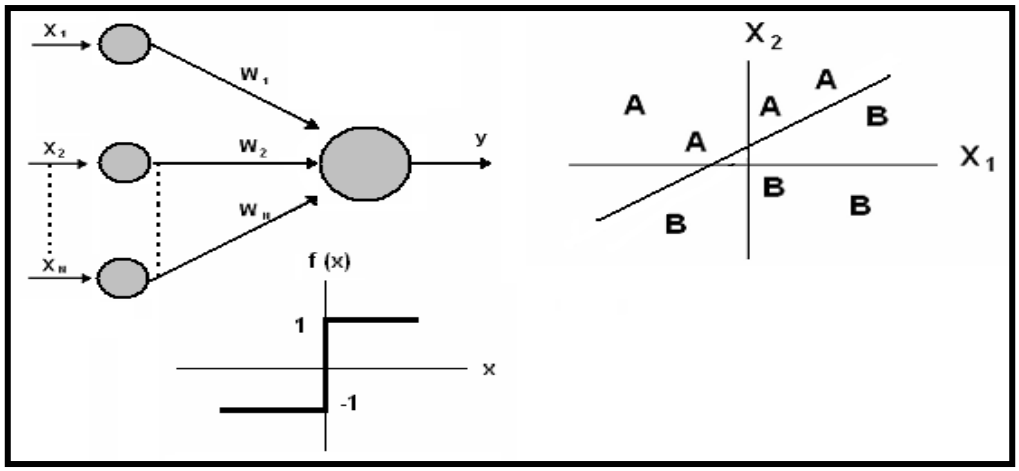


Figura 4.1: El ***Perceptron.***

Para analizar el comportamiento de redes como el ***perceptron***, se emplea una técnica que consiste en representar en un mapa las regiones de decisión creadas en el espacio multidimensional de entradas a la red. En estas regiones se visualizan los patrones que pertenecen a cada clase. El ***perceptron*** separa las regiones por un hiperplano cuya ecuación queda determinada por los ***pesos*** de las conexiones y el valor umbral de la ***función de activación*** de la neurona. En este caso, los valores de los ***pesos*** pueden fijarse o adaptarse utilizando diferentes algoritmos de entrenamiento de la red.

La separabilidad lineal, limita a la red a la resolución de problemas en los cuales el conjunto de puntos (correspondientes a los valores de entrada) sea separable geométricamente. En el caso de dos entradas, la separación se lleva a cabo mediante una línea recta. Para tres entradas, se realiza mediante un plano en el espacio tridimensional, y así sucesivamente hasta el caso de *N* entradas, en el cuál el espacio *N*-dimensional es dividido en un hiperplano.

IV.3.- REGLA DE APRENDIZAJE

El algoritmo de aprendizaje del ***perceptron*** es de tipo ***supervisado***, por lo que requiere que sus resultados sean evaluados y en caso de ser necesario se realicen las modificaciones oportunas al sistema (el algoritmo original de convergencia del ***perceptron*** fue desarrollado por **Rosenblatt)**.

El funcionamiento de la red se puede determinar por los valores de los ***pesos***, ya que estos se pueden fijar o adaptar utilizando diferentes algoritmos de entrenamiento de la red.

El ***perceptron*** no puede aprender a realizar todo tipo de clasificaciones, esa limitación se debe a que usa un separador lineal como célula de decisión, con la cual solo le es posible realizar una separación lineal (por medio de un hiperplano). A continuación se soluciona como ejemplo de su funcionamiento el problema de la ***función OR*** con dos entradas.

Ejemplo: ***Función OR***

Para resolver el problema de la ***función OR***, la ***red perceptron*** debe tener la capacidad de devolver, a partir de los patrones de entrada *00, 01,10 y 11*, a qué clase pertenece cada uno. Es decir, al patrón de entrada *00* le corresponde la clase *0* y a los restantes la clase *1,* como se muestra en la tabla 4.1.

Patrones de Entrada	Función OR	Clase
00	0	0
01	1	1
10	1	1
11	1	1

Tabla 4.1: Función OR con dos entradas.

En este caso, las entradas son dos valores binarios, la salida que produce sin tener en cuenta el valor umbral es:

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

Donde:

x_1, x_2: Son las entradas a la neurona.

w_1, w_2: Son los pesos entre las neuronas de la capa de entrada y la de salida.

f: Es la función de salida o transferencia.

Cuando $w_1 x_1 + w_2 x_2$ es mayor que *0*, la salida es *1*, y para el caso contrario *–1* (función escalón unitario). La sumatoria que pasa como parámetro (entrada total) a la función *f* (función de salida o transferencia) es la expresión matemática de una recta, donde w_1 y w_2 son variables y x_1 y x_2 son constantes.

Resumiendo, para la función ***OR*** se deben separar los valores *01*, *10* y *11* del valor *00*. Al no existir término independiente en la ecuación debido a que el umbral θ es cero, las posibles rectas pasarán por el origen de coordenadas, por lo que la entrada *00* quedará sobre la propia recta.

En la figura 4.2, se puede observar la ***red perceptron*** que realiza esta tarea y la gráfica característica.

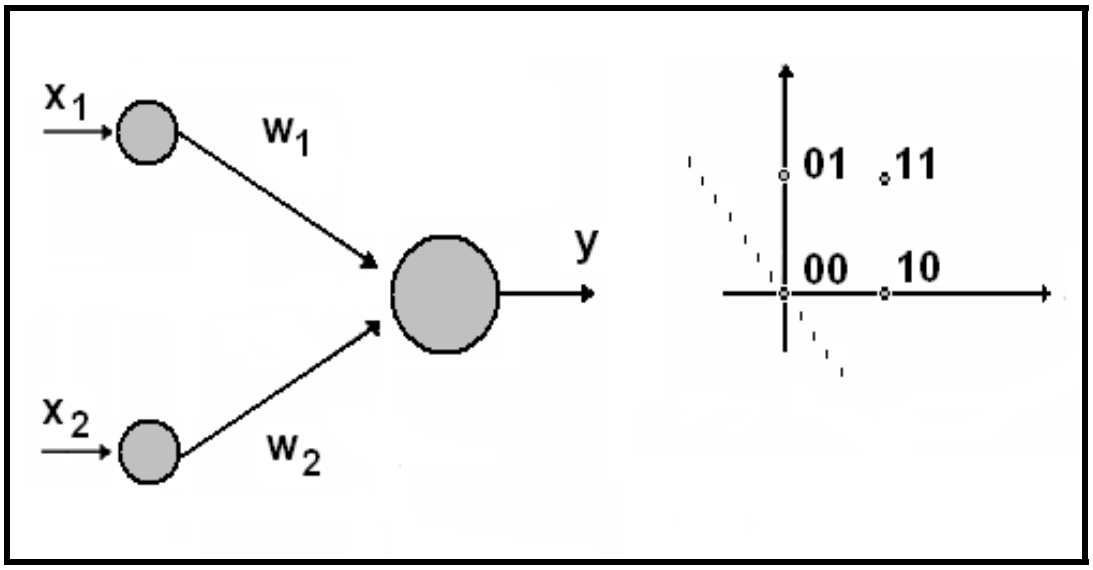


Figura 4.2: Perceptron aplicado a la solución de la función OR.

IV.4.- ALGORITMO DE ENTRENAMIENTO

El algoritmo de entrenamiento de un ***perceptron*** con *N* elementos procesales de entrada y un único elemento procesal de salida se puede resumir en los siguientes pasos:

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

2) Se presenta un nuevo patrón de entrada $X_p = (x_1, x_2, ..., x_N)$ y la salida esperada d(t).

3) Se calcula la salida actual

$$y(t) = f\,[\sum_i w_i(t)x_i(t) - \theta]$$

Siendo *f(x)* la función de transferencia de tipo escalón.

4) Adaptación de los pesos

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

$$(0 \leq i \leq N)$$

Donde *d(t)* es la salida deseada, y será *1* si el patrón pertenece a la clase *A*, y *–1* si es de la clase *B*. En estas ecuaciones, α es un factor de ganancia en el rango [*0.0* a *1.0*]. Este factor se debe ajustar de tal manera que satisfaga tanto los requerimientos de aprendizaje rápido como la estabilidad de las estimaciones de los pesos. Este proceso se repite hasta que el error que se produce para cada uno de los patrones (diferencia entre el valor de salida deseado y obtenido) es cero o bien menor que un valor preestablecido. Si la red toma la decisión correcta, los pesos ya no se cambian.

5) Volver al paso 2

Para ilustrar estos cinco pasos del algoritmo de entrenamiento del ***Perceptron*** se muestra a continuación un ejemplo del ajuste de los pesos de las conexiones de una red que debe realizar la ***función OR***, se va a utilizar un umbral distinto de cero con la finalidad de aumentar el número de posibles soluciones del problema.

Ejemplo:

1) Valores elegidos aleatoriamente:

$w_0 = 1.5$
$w_1 = 0.5$
$w_2 = 1.5$

2) Se toman uno por uno de los cuatro patrones de entrada y se aplica el método expuesto como se muestra a continuación:

2.1) Patrón de entrada 00

Entradas: $x_1 = 0,\ x_2 = 0,\ x_0 = 1$

Pesos: $w_1(t) = 0.5$, $w_2(t) = 1.5$, $w_0(t) = 1.5$

Net_i: $[(0 * (0.5)) + (0 * (1.5)) + (1 * (1.5))] = 1.5$

Salida obtenida: $1(Net_i >= 0)$

Salida deseada: 0

Error: $(Salida\ deseada - Salida\ obtenida) = (0 - 1) = -1$

Pesos modificados: $w_1(t+1) = [0.5 + ((-1) * 0)] = 0.5$

$w_2(t+1) = [1.5 + ((-1) * 0)] = 1.5$

$w_0(t+1) = [1.5 + ((-1) * 1)] = 0.5$

Cálculos en forma matricial:

$$Entrada * Pesos = net \longrightarrow Salida = f(net)$$

$$[1\ 0\ 0] * \begin{bmatrix} 1.5 \\ 0.5 \\ 1.5 \end{bmatrix} = 1.5 \qquad \Rightarrow \qquad S = f(1.5) = 1$$

$$Error = (salida\ deseada - salida\ obtenida) = -1$$

$$Pesos\ (t+1) = Pesos\ (t) + (Error * Entrada)$$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 0.5 \\ 1.5 \end{bmatrix} + (-1) * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 1.5 \end{bmatrix} \quad \Rightarrow \quad \textit{Nuevos pesos}$$

2.2) Patrón de entrada 01

Entradas: $x_1 = 0,\ x_2 = 1,\ x_0 = 1$

Pesos: $w_1(t) = 0.5$, $w_2(t) = 1.5$, $w_0(t) = 0.5$

Net_i*:[(0 * (0.5)) + (1 * (1.5)) + (1 * (0.5))] = 2*

Salida obtenida: *1*

Salida deseada: *1*

Error: *0*

Los pesos no se modifican: $w_i(t+1) = w_i(t)$

2.3) Para las entradas *10* y *11* la salida obtenida es igual que la deseada, por lo que los pesos no varía. En el caso de que no fuese así, se aplicaría el método antes empleado.

Como para el patrón de entrada *00*, el error cometido no es cero, se realiza de nuevo el procedimiento a partir del punto *2*.

3) ***Se toman de nuevo los cuatro patrones de entrada:***

3.1) Patrón de entrada 00

Entradas: $x_1 = 0,\ x_2 = 0,\ 1$

Pesos: $w_1(t) = 0.5$, $w_2(t) = 1.5$, $w_0(t) = 0.5$

Net_i: $[(0 * (0.5)) + (0 * (1.5)) + (1 * (0.5))] = 0.5$

Salida obtenida: *1*

Salida deseada: *0*

Error: *-1*

Pesos modificados: $w_1(t+1) = [0.5 + ((-1) * 0)] = 0.5$

$w_2(t+1) = [1.5 + ((-1) * 0)] = 1.5$

$w_0(t+1) = [0.5 + ((-1) * 1)] = -0.5$

En forma matricial:

*Entrada * Pesos = net* $\longrightarrow$ *Salida = f (net)*

$$[1\ 0\ 0] * \begin{bmatrix} 0.5 \\ 0.5 \\ 1.5 \end{bmatrix} = 1.5 \qquad \Rightarrow \qquad S = f(0.5) = 1$$

Error = (Deseada – Obtenida) = 0 – 1 = -1

*Pesos (t+1) = Pesos (t) + (Error * Entrada)*

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 1.5 \end{bmatrix} + (-1) * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 1.5 \end{bmatrix} \quad \Rightarrow \quad \textit{Nuevos pesos}$$

3.2) Patrón de entrada 01

Entradas: $x_1 = 0,\ x_2 = 1,\ 1$

Pesos: $w_1(t) = 0.5$, $w_2(t) = 1.5$, $w_0(t) = -0.5$

Net_i: $[(0 * (0.5)) + (1 * (1.5)) + (1 * (1.5))] = 1$

Salida obtenida: *1*

Salida deseada: *1*

Error: $1 - 1 = 0$

Los pesos no se modifican

3.3) Para las entradas 10 y 11, los pesos no varían.

Siguen habiendo una entrada cuyo error ha sido diferente de cero.

6) Nuevamente se toman los cuatro patrones de entrada:

4.1) Patrón 00

Entradas: $x_1 = 0,\ x_2 = 0,\ 1$

Pesos: $w_1(t) = 0.5$, $w_2(t) = 1.5$, $w_0(t) = -0.5$

Net_i: $[(0 * (0.5)) + (0 * (1.5)) + (1 * (-0.5))] = -0.5$

Salida obtenida: *0*

Salida deseada: *0*

Error: *0*

Los pesos no se modifican

4.2) Si no han variado los pesos, entonces para el resto de las entradas el error cometido es cero.

Con estos nuevos pesos, al calcular la salida que se obtiene para cualquiera de los cuatro patrones de entrada ya no se comete ningún error, por lo que la etapa de aprendizaje concluye.

IV.5.- EL PERCEPTRON MULTICAPA

El ***perceptron multicapa*** es una red de tipo ***feedforward*** (con conexiones hacia delante), formada por varias capas de neuronas entre la entrada y la salida de la misma, esta red permite establecer regiones de decisión mucho más complejas que las de dos semiplanos, como lo hace el ***perceptron*** simple.

En la figura 4.3 se muestra una ***red perceptron multicapa*** en la cual se pueden observar las conexiones existentes entre los nodos de la red y los tres tipos de capas que la conforman.

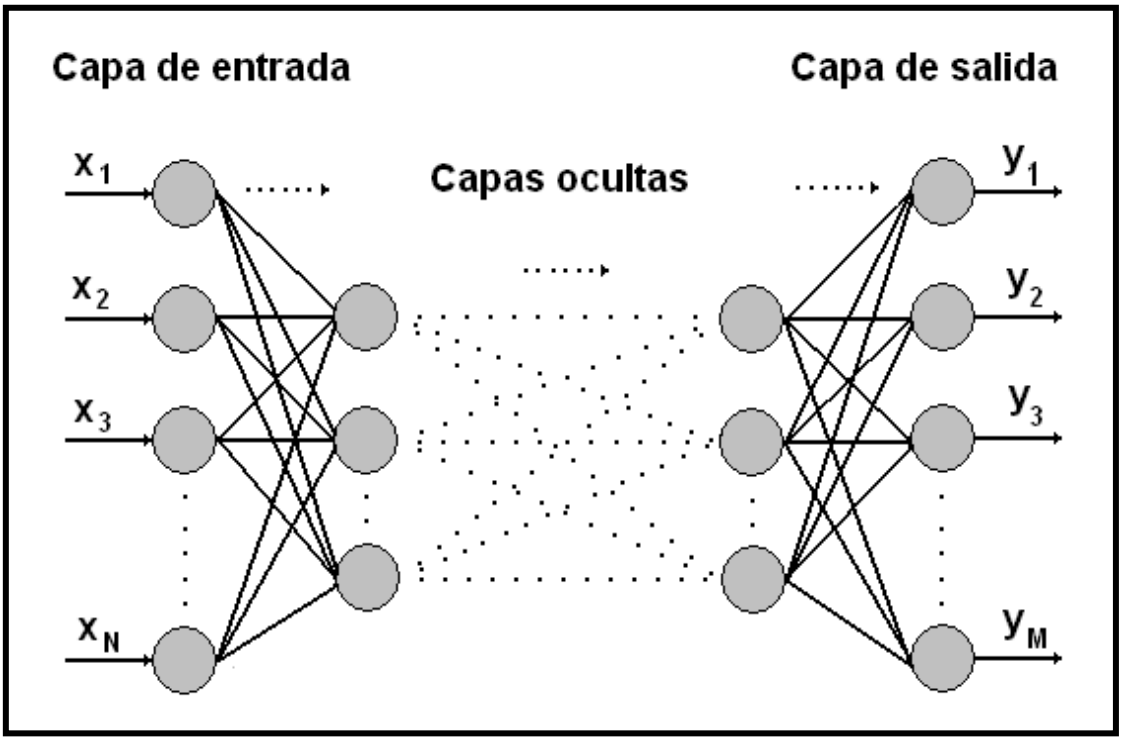


Figura 4.3: Perceptron multicapa

Un ***perceptron*** simple de dos capas (la de entrada y la de salida), sólo puede establecer dos regiones separadas por una frontera lineal en el espacio de patrones de entrada, mientras que un ***perceptron multicapa*** puede formar cualquier región convexa en este espacio. Estas regiones se forman mediante la intersección de regiones compuestas por cada neurona de la segunda capa, donde cada uno de estos elementos se comporta como un ***perceptron*** simple, activándose su salida para los patrones de un lado del hiperplano. Si el valor de los pesos de las conexiones entre las neuronas de la segunda capa N_2 y una neurona del nivel de salida son todos igual a 1, y el ***umbral*** de la salida es $(N_2 - a)$, donde $0 < a < 1$, entonces la salida de la red se activará sólo si las salidas de todos los nodos de la segunda capa están activos, lo cual equivale a ejecutar la función lógica ***AND*** en el nodo de salida, resultando una región de decisión intersección de todos los semiplanos formados en el nivel anterior. La región de decisión resultante de la intersección será una región convexa con un número de lados igual al número de neuronas de la segunda capa.

A partir de este análisis surge el interrogante respecto a los criterios de selección para las neuronas de las capas ocultas de una ***red multicapa***, este número en general debe ser lo suficientemente grande como para que se forme una región compleja que pueda resolver el problema, aunque tampoco es conveniente que el número de nodos sea tan grande que la estimación de los pesos no sea confiable para el conjunto de patrones de entrada disponibles.

En la práctica no se requieren más de cuatro capas en una red de tipo ***perceptron multicapa***, debido a que una red con estas características puede generar regiones de decisión arbitrariamente complejas.

Para algunos problemas se puede simplificar el aprendizaje mediante el aumento del número de neuronas ocultas. Sin embargo, la tendencia es el aumento de la extensión de la ***función de activación***, en lugar del aumento de la complejidad de la red. Lo cual nos lleva nuevamente al problema del número de neuronas que debemos seleccionar para un ***perceptron multicapa***, debido a que un número de neuronas excesivo en cualquier capa puede generar ruido. Aunque si existe un número de neuronas redundantes se obtiene mayor tolerancia a fallos.

Hasta el momento no hay un criterio establecido para determinar la configuración de la red ya que esto depende más bien de la experiencia del diseñador.

En la figura 4.4, se muestran las capacidades de una ***red perceptron*** con dos, tres y cuatro capas y con una única neurona en el nivel de salida, el tipo de región de decisión que puede formar con cada una de las configuraciones, las regiones formadas para resolver el problema de clases con regiones mezcladas y las formas de regiones más generales para cada uno de los casos.

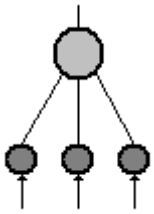
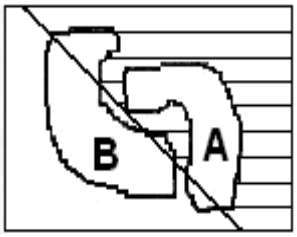
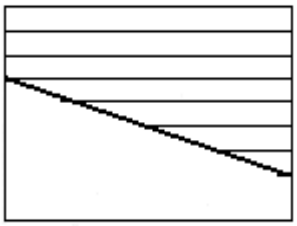
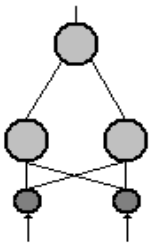
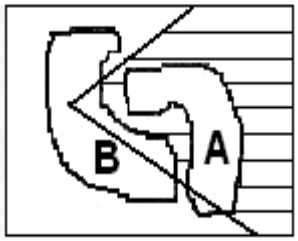
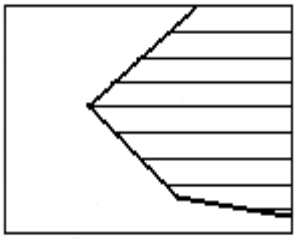
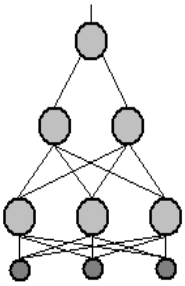
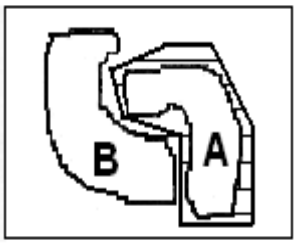
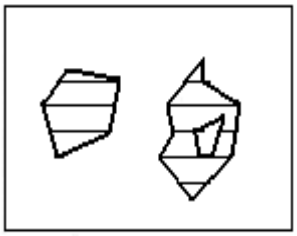
ESTRUCTURA	REGIONES DE DECISIÓN	CLASES CON REGIONES MEZCLADAS	FORMAS DE REGIONES MÁS GENERALES
	Medio plano limitado por un hiperplano		
	Regiones cerradas o convexas		
	Complejidad arbitraria limitada por el número de neuronas		

Figura 4.4: Formas distintas de las regiones generadas por un perceptron multinivel.

CAPÍTULO V

APLICACIONES DE LAS RNA EN LA INGENIERÍA ELÉCTRICA-ELECTRÓNICA Y SU IMPLEMENTACIÓN

V.1.- APLICACIONES

Desde hace unos años las ***redes neuronales*** vienen alcanzando excelentes resultados en diversas aplicaciones. Son utilizadas en la resolución de problemas prácticos concretos, gracias a su capacidad de aprendizaje, robustez, no linealidad y tolerancia a la imprecisión e incerteza del entorno.

En la actualidad, son empleadas por muchas compañías de modo rutinario en la resolución de numerosos problemas. Los más frecuentes son los relacionados con clasificación, estimación funcional y optimización, en general, el reconocimiento de patrones. Las ***redes neuronales*** pueden realizar tareas concretas mejor que otras tecnologías convencionales y se pueden desarrollar en un periodo de tiempo razonable. Cuando son implementadas mediante hardware (***redes neuronales*** en chips ***VLSI***), presentan una gran tolerancia a los errores del sistema y proporcionan un grado muy alto de paralelismo en el proceso de datos, lo cual hace posible insertar ***redes neuronales*** de bajo costo a sistemas ya existentes y a los recientemente desarrollados.

Existen diferentes tipos de ***redes neuronales***, cada uno de los cuales tiene una aplicación particular más apropiada. Se pueden señalar, entre otras, las siguientes áreas: Biología, Economía, Finanzas, Medicina, Medio Ambiente, Manufacturación, Militares, etc. Pero para este trabajo sólo nos centraremos en sus aplicaciones en la ***Ingeniería Eléctrica-Electrónica***.

A continuación se mencionan de forma muy general las áreas de aplicación más importantes dentro de la ***Ingeniería Eléctrica-Electrónica*** para las cuales las ***redes neuronales artificiales***, pueden ser utilizadas actualmente.

a) Reconocimiento de patrones

El ***reconocimiento de patrones*** inicialmente se refería a la detección de formas simples, aunque siempre ha tenido por meta implementar la ***percepción artificial***, es decir, imitar las funciones de los sistemas sensoriales biológicos, lo cual resulta muy complicado, ya que para ello es necesario replicar al cerebro con todas sus capacidades de pensamiento. Por ejemplo para el análisis de imágenes, existen requerimientos muy difíciles de alcanzar como son la invariabilidad de la detección con respecto a la traslación, rotación, escala y perspectiva de los objetos, sobre todo en condiciones de iluminación variables. En ingeniería existe un gran número aplicaciones para las cuales las soluciones artificiales pueden ser inclusive más eficaces debido a que generalmente los problemas son simplificados, no obstante estos métodos tienen muy poco en común con los principios de operación de las funciones sensoriales biológicas, quedando abierto el problema básico de la percepción en ***computación neuronal***.

Las siguientes son algunas aplicaciones del ***reconocimiento de patrones*** en la ingeniería.

- Reconocimiento de caracteres escritos
- Reconocimiento del habla
- Procesamiento y restauración de imágenes con ruido
- Segmentación y clasificación de regiones de imágenes
- Visión en computadoras industriales (en especial robots)

b) Control de Robots

Para muchas de las aplicaciones del control de procesos, el control adaptativo en tiempo real es esencial, tal es el caso del guiado del movimiento de un robot. Existen dos categorías importantes de robots, los de ***trayectoria programada*** (en estos se almacena en memoria una secuencia de coordenadas y comandos para controlar sus movimientos y acciones en la forma deseada) y los ***robots inteligentes*** (los cuales se supone que pueden planear sus acciones). La inteligencia en estos últimos se ha implementado mayormente por programas de ***IA***, aunque con frecuencia se requiere tener un nivel más alto de aprendizaje que permita por ejemplo aprender a moverse en un entorno desconocido, la coordinación de las funciones sensoriales con las funciones motoras, etc., las cuales son tareas que no se puede resolverse de forma analítica propiciando el uso de los ***sistemas neuronales***. Actualmente se han realizado simulaciones por computadora que demuestran tales capacidades de aprendizaje autónomo.

c) Filtrado de Señales

En la eliminación de ruidos y desórdenes en señales, o para la reconstrucción de patrones a partir de datos parciales, se están empleando ***redes neuronales*** como filtros, ejemplo de ello son la red ***Backpropagation*** que a menudo es utilizada para obtener una versión de ruido-reducido a partir de una señal de entrada, y la red ***Madaline*** la cual se utiliza comercialmente para mejorar la transmisión telefónica.

d) Segmentación y comprensión de datos

La mayoría de los algoritmos de segmentación de datos o imágenes, no proporcionan completamente los resultados deseados, una causa de ello es que se ven muy afectados por varios factores, por ejemplo en la segmentación de imágenes las líneas que limitan las regiones generalmente están mezcladas con ruido u otras interferencias de pequeña escala. Con la finalidad de solucionar este problema actualmente se han implementado numerosas aproximaciones de ***redes neuronales***, de las cuales la mayoría presenta una capacidad superior en comparación con algunos de los algoritmos de segmentación más complejos. Por otro lado, en la comprensión de datos se están probando diferentes tipos de ***redes neuronales*** con resultados muy prometedores.

V.2.- REALIZACIÓN

Existen básicamente tres formas para la realización de ***redes neuronales artificiales***, las cuales son las siguientes:

- **Simulación mediante software**
- **Neurocomputadoras**
- **Chips neuronales**

A continuación se describen brevemente cada una de estas alternativas.

V.2.1.- Simulación mediante software

La ***simulación mediante software*** es el procedimiento más simple, económico e insustituible hasta el momento para realizar el entrenamiento y evaluación de las ***redes neuronales artificiales***, consiste en simular la red en una computadora convencional mediante un software específico. Su desventaja radica en que se intentan simular redes con un alto grado de paralelismo sobre máquinas que trabajan de forma secuencial.

Actualmente existe un gran número de productos y software para el desarrollo de aplicaciones con ***redes neuronales artificiales***. Por lo que es importante tomar en cuenta ciertos factores antes de elegir uno. Por ejemplo: el tipo y número de arquitecturas de red que soporta (backpropagation, Hopfield, Kohonen, etc.), la velocidad de procesamiento (número de conexiones por

unidad de tiempo), la interfaz gráfica, etc. En la figura 5.1 se muestran algunos ejemplos de software neuronal:

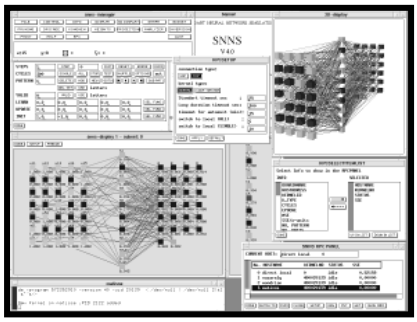
STUTTGART NEURAL NETWORK SIMULATOR 	**SNNS (STUTTGARTER NEURAL NETWORK SIMULATOR) 4.1** Simulador desarrollado por la ***Universidad de Stuttgart Alemania***, para múltiples tipos de redes con interfase X11: topología gráfica 2D y 3D, visualización de entrenamiento, múltiples patrones, etc., soporta arquitecturas de redes tipo *backpropagation*, *counterpropagation*, *quickprop*, *redes de funciones de base radial*, *correlación en cascada*, *Hopfield*, *Jordan y Elman*, *mapas auto-organizados*, *etc.* Trabaja sobre plataformas *SunOS*, *Solaris*, *IRIX*, *ltrix*, *OSF*, *AIX*, *HP/UX*, *NextStep y Linux.*
ALYUDA FORECASTER	**ALYUDA FORECASTER** Software de ***RNA*** para predicciones, análisis de datos y clasificación. Fue diseñado para ayudar a gerentes e ingenieros en la solución de las predicciones y problemas de estimación. Posee una interfaz Wizard, análisis automático de datos, y selección automática de parámetros. Trabaja sobre plataformas *Windows 98/Me/2000/XP*.
ALYUDA NEUROINTELLIGENCE	**ALYUDA NEUROINTELLIGENCE** Software de ***RNA*** para expertos, diseñado para el soporte inteligente en la aplicación de redes neuronales para resolver problemas de predicción, clasificación, y de aproximación del "mundo real". Utiliza características inteligentes para preprocesar hojas de datos, encontrar la arquitectura más eficiente para el tipo de problema y aplicar la red neuronal a los nuevos datos.

Figura 5.1: Ejemplos de productos software neuronal.

V.2.2.- Neurocomputadoras

Las ***neurocomputadoras*** son una alternativa eficaz para solucionar problemas prácticos en tiempo real. Esta característica hace que los elementos procesadores sean más rápidos que las computadoras digitales. Debido a que no es necesario buscar datos e instrucciones en una memoria y así los tiempos de retardo solo dependen de la tecnología electrónica utilizada en la realización de la ***neurocomputadora.***

En la actualidad ya existe una serie de ***neurocomputadoras*** comerciales destinadas a la realización de ***redes neuronales***; podemos citar los aceleradores BrainMaker, el acelerador para ***PC IBM*** ZISC/ISA, los sistemas de ***redes neuronales*** - NT5000 & NT6000, etc.

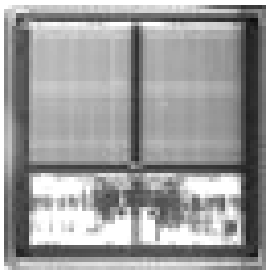
V.2.3.- Chips neuronales

La realización de ***redes neuronales*** mediante su implementación por uno o varios circuitos integrados específicos tiene como finalidad construir un elemento o conjunto de elementos que se comporten lo más cercano posible a una ***red neuronal.*** Estos elementos son los llamados ***chips neuronales***, en ellos las neuronas y las conexiones se emulan con dispositivos específicos, de tal forma que la estructura del circuito integrado refleja la arquitectura de la red, obteniendo con ello una mayor velocidad, la cual permite en varias

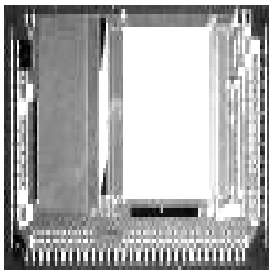
ocasiones un proceso en tiempo real. Su desventaja es una notable pérdida de la flexibilidad.

En la actualidad, estos chips se utilizan para aplicaciones en las que la ***red neuronal*** ya se ha diseñado y probado por otros métodos y lo que se requiere es una mayor velocidad para poder trabajar en tiempo real. En la figura 5.2 se muestran algunos ejemplos de ***chips neuronales.***

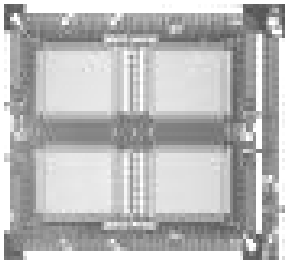
CHIPS NEURONALES



SPIKING NEURAL NETWORK CHIP (SPIKEY, 2005)



VISION ENHANCEMENT FOR A NEURAL NETWORK ARCHITECTURE (VIENNA, 2003)



HEIDELBERG ANALOG EVOLVABLE NETWORK (HAGEN, 2001)

Figura 5.2: Ejemplos de chips neuronales.

V.3.- EJEMPLO PRÁCTICO DE UNA RED NEURONAL ARTIFICIAL: APRENDIZAJE DE LA *FUNCIÓN OR-EXCLUSIVA*

Con la finalidad de ilustrar el funcionamiento de una ***red neuronal artificial*** aplicada a la Ingeniería Eléctrica-Electrónica, se dará solución al problema de la ***función OR-EXCLUSIVA*** mediante una red tipo ***perceptron multicapa.***

V.3.1.- Descripción de problema:

Para solucionar el problema de la ***función OR-EXCLUSIVA*** se requiere que para los valores de entrada *00* y *11* se devuelva la clase *0*, y para los valores *01* y *10* la clase *1*. En la tabla 5.1 se muestra la tabla de verdad de la ***función OR-EXCLUSIVA.***

TABLA DE VERDAD FUNCIÓN OR-EXCLUSIVA		
X	**Y**	**SALIDA**
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 5.1: Tabla de verdad de la función OR-EXCLUSIVA.

V.3.2.- Justificación del tipo de red:

Como se puede apreciar en la figura 5.1, los valores de salida de la ***función OR-EXCLUSIVA*** no son separables linealmente es decir no existe ninguna recta que separe los patrones de una clase de los de la otra, es por ello que el problema no puede ser resuelto mediante un ***perceptron*** sencillo, por tal motivo se implementa una red tipo ***perceptron multicapa***.

Antes de dar solución al problema de la ***función OR-EXCLUSIVA*** se describe brevemente el método de aprendizaje del ***perceptron multicapa*** y algunos parámetros importantes que determinan el aprendizaje de dicha red.

Método de aprendizaje del perceptron multicapa:

Al ***perceptron multicapa*** también se le denomina ***red backpropagation*** *(red de retropropagación)*, debido a que en su entrenamiento utiliza el método conocido como ***backpropagation error*** (propagación del error hacia atrás). Este método de aprendizaje básicamente se realiza en dos etapas:

1) Los valores de entrada se presentan a la capa de entrada de la red. Esta información se utiliza para realizar varias operaciones a lo largo de las distintas capas hasta la capa de salida, en la que se genera un resultado.

2) El resultado que proporciona la red en la capa de salida es comparado con el resultado esperado para cada uno de los patrones de entrenamiento. Si no coinciden el error se utiliza para modificar los pesos de las capas ocultas.

Tasa de aprendizaje:

El valor de la ***tasa de aprendizaje η*** tiene un papel crucial en el proceso de entrenamiento de una ***red neuronal artificial***, ya que controla el tamaño del cambio de los pesos en cada iteración, con lo cual determina en gran medida la velocidad de aprendizaje y el tiempo que la red requiere para entrenarse. En general, este valor debe estar comprendido entre *0.5* y *0.05*. Para mejorar las posibilidades de generalización de la red es conveniente que, a medida que la red vaya aprendiendo, el valor de la ***tasa de aprendizaje*** vaya disminuyendo hasta alcanzar un valor muy próximo a *0*. Sin embargo, el valor de la ***tasa de aprendizaje*** no es crítico para la mayoría de los casos, a no ser que se trate de problemas excesivamente complejos.

Momento:

El factor ***momento μ*** tiene como objetivo reducir el tiempo de entrenamiento de las ***redes neuronales artificiales.*** Acelera considerablemente la convergencia de los pesos, además hace que estos cambien su valor siempre en la misma dirección, a no ser que el sentido global del aprendizaje de la red

indique un cambio de sentido. En general el valor de este factor debe estar dentro del intervalo de *0* y *1*, aunque comúnmente se toma un valor muy próximo a *1* (por ejemplo, *0.9*).

Bias o polarización:

Las ***bias*** o ***polarizaciones*** son unos pesos extras añadidos a las neuronas de las capas intermedias y a las de la capa de salida que proporcionan un nivel de activación independiente de las entradas y de los vectores de entrenamiento.

Conjunto de entrenamiento:

Es el conjunto de datos utilizado para el entrenamiento de la red, y determina lo que la red debería ser capaz de solucionar. Dicho conjunto debe ser representativo del problema, para que la red sea capaz de responder satisfactoriamente a cualquier tipo de entrada. Un entrenamiento insuficiente de la red hace que ésta no sea capaz de proporcionar respuestas claras, y por el contrario, un exceso de entrenamiento hace que la red memorice los patrones de entrenamiento.

V.3.3.- Desarrollo:

Patrones de Entrenamiento

Los ***patrones de entrenamiento*** correspondientes a la ***función OR-EXCLUSIVA*** utilizados para el entrenamiento de la red desarrollada se muestran en la tabla 5.2.

PATRONES DE ENTRENAMIENTO FUNCIÓN OR-EXCLUSIVA			
ID Patron	**Entrada 1**	**Entrada 2**	**Salida deseada**
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Tabla 5.2: Patrones de Entrenamiento.

Topología de la red

Un ***perceptron multicapa*** está compuesto por una capa de entrada, una capa de salida y una o más capas ocultas, aunque se ha demostrado que para la mayoría de los problemas es suficiente una sola capa oculta, por lo tanto para el desarrollo de la red solo se emplean tres capas (la de entrada, la oculta o intermedia y la de salida). En la figura 5.3 se muestra dicha topología.

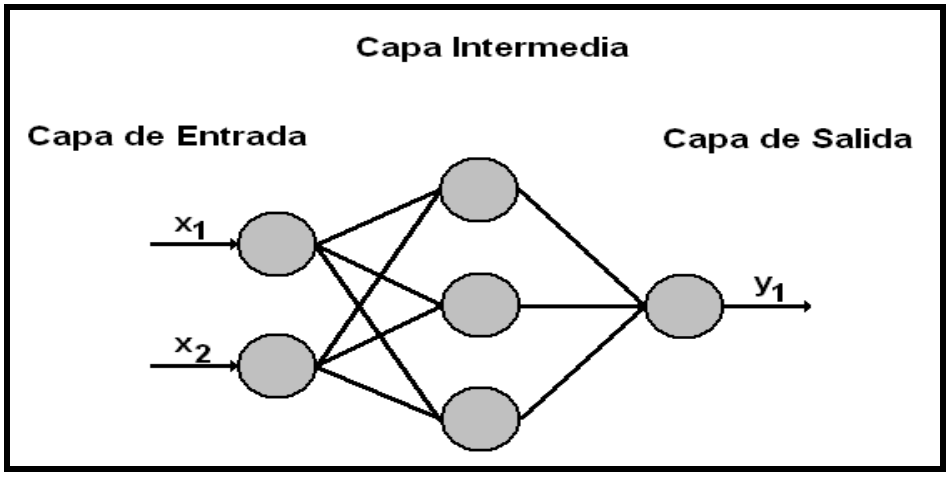


Figura 5.3: Topología red percectron multicapa.

Esquema de entradas y salidas de la red

Para el desarrollo de la red se utilizó el esquema de entradas y salidas que se muestra a continuación en la tabla 5.3.

Neuronas de entrada	2
Neuronas de la capa intermedia	3
Neuronas de salida	1
Tasa de aprendizaje	0.45
Momento	0.9
Numero de interacciones	4
Numero de patrones de entrenamiento	4

Tabla 5.3: Esquema de entradas y salidas de la red.

V.3.4.- Sumario del método de aprendizaje:

El siguiente sumario comprende las ecuaciones relevantes para el entrenamiento de una red ***perceptron multicapa*** con las características antes mencionadas e ilustradas en la figura 5.3 y que utiliza ***aprendizaje con retropropagación.*** Las ecuaciones están representadas en el orden en que deben ejecutarse.

1.- Presentar el vector de entrada $x^p = (x_1^p, x_2^p, \dots, x_N^p)^T$ en la capa de entrada.

Donde:

x^p: ***p***-ésimo vector de entrada del conjunto de entrenamiento.

2.- Calcular el valor de los niveles de excitación de las neuronas de la capa intermedia, en base a la siguiente expresión:

$$s_i^p = \sum_{j=1}^{N} w_{ji} x_j^p + b_i$$

Donde:

s_i^p: Nivel de excitación para la neurona oculta ***i*** ante el vector de entrada x^p.

I: Número de entradas de la red.

w_{ji}: Valor del peso de la conexión entre la neurona ***j*** de la capa de entrada y la neurona ***i*** de la capa intermedia.

b_i: Valor de las bias asociadas a la neurona ***i***.

3.- Calcular las salidas de las neuronas de la capa intermedia.

$$y_i^p = F_i(s_i^p)$$

Donde:

y_i^p: Valor de salida de la capa intermedia.

F_i: Función de activación de la neurona ***i***.

4.- Calcular el valor de los niveles de excitación de las neuronas de la capa de salida, de acuerdo con la siguiente expresión:

$$s_k^p = \sum_{i=1}^{L} w_{ik} y_i^p + b_k$$

Donde:

s_k^p: Nivel de excitación para la neurona ***k*** de la capa de salida ante el vector de entrada x^p.

L: Número de neuronas de la capa intermedia.

w_{ik}: Valor del peso de la conexión entre la neurona ***i*** de la capa intermedia y la neurona ***k*** de la capa de salida.

b_k: Valor de las bias asociadas a la neurona ***k***.

Nota: El valor inicial de los pesos se genera de forma aleatoria.

5.- Calcular las salidas de la red.

$$y_k^p = F_k(s_k^p)$$

Donde:

y_k^p: Valor de salida de la capa de salida.

F_k: Función de activación.

6.- Calcular la sensibilidad de las neuronas de la capa de salida a partir del error apreciado en la salida con el vector de salida deseado $d^p = (d_1^p, d_2^p, \dots, d_M^p)^T$.

$$\delta_k^p = (d_k^p - y_k^p) F_k(s_k^p)$$

Donde:

d^p: ***p***-ésimo vector de salida esperado.

δ_k^p: Sensibilidad de la ***k***-ésima neurona de la capa de salida.

$(d_k^p - y_k^p)$: Error de la salida presente en la neurona ***k*** para el ***p***-ésimo patron de entrenamiento.

7.- Calcular la sensibilidad de las neuronas de la capa intermedia, en base a la siguiente expresión:

$$\delta_i^p = F_i(s_i^p) \sum_{k=1}^{M} w_{ik}\, \delta_k^p$$

Donde:

δ_i^p: Sensibilidad de la ***i***-ésima neurona de la capa intermedia.

M: Número de salidas de la red.

8.- Actualizar los pesos y las bias de las conexiones que unen las neuronas de la capa intermedia con las de la capa de salida.

$$w_{ik}(t+1) = w_{ik}(t) + \eta \delta_k^p y_i^p + \mu\left(w_{ik}(t) - w_{ik}(t-1)\right)$$

$$b_k(t+1) = b_k(t) + \eta \delta_k^p y_i^p + \mu\left(b_k(t) - b_k(t-1)\right)$$

Donde:

η: Tasa de aprendizaje.

μ: Momento.

9.- Actualizar los pesos y las bias de las conexiones que unen las neuronas de la capa de entrada con las de la capa intermedia.

$$w_{jl}(t+1) = w_{jl}(t) + \eta\delta_i^p x_j^p + \mu\left(w_{jl}(t) - w_{jl}(t-1)\right)$$

$$b_l(t+1) = b_l(t) + \eta\delta_i^p x_j^p + \mu\left(b_l(t) - b_l(t-1)\right)$$

10.- Calcular el término de error.

$$E^p = \frac{1}{2}\sum_{k=1}^{M}\left(d_k^p - y_k^p\right)^2$$

Donde:

E^p: Error.

Nota: Este término refleja la capacidad de adaptación de la red, por lo que hay que tenerlo en cuenta para determinar si la red aprende de forma satisfactoria o no.

V.3.4.- Resultados:

En la realización del programa que ilustra el funcionamiento de una ***red neuronal artificial percetron multicapa*** para solucionar el problema de la ***función OR-EXCLUSIVA,*** se utilizo el compilador Borland C++ 5.5, la interfaz gráfica Relo2.0 y se incluyeron las librerías patron.h, neuronal.h y backprop.h. del material adicional de ***redes neuronales artificiales:*** un enfoque práctico *(Juan Manuel Corchado, 2000).*

La red desarrollada se entreno para 40,000 iteraciones, y para su realización se emplearon los siguientes datos: el esquema de entradas y salidas que se muestra en la tabla 5.3, los datos de la tabla 5.2 como patrones de entrada, función de transferencia continua y el valor inicial de los pesos se genero de forma aleatoria. El algoritmo de este programa se ilustra en el apéndice C.

Mediante los valores fijados se logró un buen desempeño de la red. Los resultados obtenidos se agrupan en las tablas 5.4 y 5.5, las cuales se muestran a continuación:

ITERACCIONES	ERROR
0	0.4925597578
1000	0.1612077626
2000	0.1354869473
3000	0.1310440933
4000	0.1292252239
5000	0.1282398319
6000	0.1276231610
7000	0.1272014295
8000	0.1268950835
9000	0.1266626059
10000	0.1264802305
11000	0.1263333806
12000	0.1262126229
13000	0.1261115855
14000	0.1260258102
15000	0.125952085
16000	0.1258880376
17000	0.1258318776
18000	0.1257822289
19000	0.1257380163

20000	0.1256983873
21000	0.1256626573
22000	0.1256302700
23000	0.1256007684
24000	0.1255737736
25000	0.1255489682
26000	0.1255260844
27000	0.1255048937
28000	0.1254851999
29000	0.1254668330
30000	0.1254496437
31000	0.1254335003
32000	0.1254182842
33000	0.1254038872
34000	0.1253902086
35000	0.1253771520
36000	0.1253646215
37000	0.1253525178
38000	0.1253407314
39000	0.1253291366
40000	0.1253175718

Tabla 5.4: Resultados del Entrenamiento.

VALOR OBTENIDO/VALOR ESPERADO			
Patron	**Entrada**	**Valor de la Red**	**Valor del Patron**
0	0,0	0	0
1	0,1	1	1
2	1,0	1	1
3	1,1	0	0

Tabla 5.5: Comparación del valor obtenido con el esperado.

Como podemos apreciar en la tabla 5.4, al final del proceso iterativo el error entregado por la red fue casi cero *(0.1253175718)*, el cual es un valor bastante aceptable, demostrando la efectividad de la ***red neuronal artificial perceptron multicapa*** para resolver problemas que no son linealmente separables, su facilidad de implementación y su capacidad de generalización, entendida como la facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento.

CONCLUSIONES

Las ***redes neuronales artificiales*** son una teoría relativamente nueva que junto a otras técnicas de ***inteligencia artificial*** han generado soluciones muy confiables a problemas de ingeniería, los cuales a pesar de poder ser solucionados por métodos tradicionales, encuentran en las ***redes neuronales artificiales*** una alternativa altamente segura, la cual se puede implementar en un periodo de tiempo razonable.

El ***perceptron***, a pesar de que cuenta con serias limitaciones en especial su incapacidad para solucionar problemas que no sean linealmente separables, sigue conservando su importancia, ya que ha servido de inspiración para el desarrollo de otros modelos de redes tales como las ***redes multicapa***.

Con el objetivo de ilustrar el funcionamiento de una ***red neuronal artificial*** aplicada a la Ingeniería Eléctrica-Electrónica, se desarrolló una red tipo ***perceptron multicapa*** entrenada mediante el método de aprendizaje ***backpropagation error***, con la cual se logró un buen desempeño mediante los valores fijados, obteniendo al final del proceso iterativo un error muy próximo a cero.

En el proceso de entrenamiento de la red desarrollada se observó que las ***redes neuronales artificiales*** tienen gran capacidad para aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que presentan información irrelevante; en términos generales las ***redes neuronales artificiales*** son una teoría relativamente nueva y como tal presentan aún algunas limitaciones,

pero su relativa facilidad de implementación y la calidad en la información que entregan como respuesta, son razón suficiente para que continúe su estudio y desarrollo.

Para finalizar se espera que este trabajo introductorio sirva como una base a los alumnos que estén interesados en realizar estudios especializados referentes a las ***redes neuronales artificiales,*** en especial los relacionados a alguna área de aplicación dentro de la Ingeniería Eléctrica-Electrónica, las cuales se abordaron de forma general en este trabajo.

APÉNDICE A:
GLOSARIO DE TÉRMINOS

1) ***Algoritmo genético:*** Es un procedimiento de resolución de problemas, el cual imita los métodos de la evolución genética de los seres vivos para solucionar problemas de búsqueda y optimización.

2) ***Arquitectura Von Neumann:*** Se refiere a las arquitecturas de computadoras que utilizan el mismo dispositivo de almacenamiento tanto para las instrucciones como para los datos. Dicha arquitectura constan de cinco partes esenciales: la unidad aritmético-lógica o ALU, la unidad de control, la memoria, un dispositivo de entrada/salida y el bus de datos.

3) ***Axón:*** Es la rama de "salida" de la neurona encargada de enviar impulsos o señales a otras células nerviosas.

4) ***Bases de conocimiento:*** Son la evolución lógica de los sistemas de bases de datos tradicionales, en un intento de no plasmar más cantidades gigantes de datos, sino elementos de conocimiento (normalmente en forma de hechos y reglas) además de que se les trata de dotar de conocimiento sobre sí mismas.

5) ***Cibernética:*** Ciencia que estudia el funcionamiento de las conexiones nerviosas en los seres vivos y los sistemas de comunicación, así como la regulación automática de los seres vivos con sistemas artificiales que simulan a los biológicos.

6) ***Cognitron:*** Red neuronal competitiva multicapa y autoorganizada propuesta por **Fukushima** en 1975.

7) ***Computación evolutiva:*** Rama de la inteligencia artificial que tiene su inicio en 1993, retomando conceptos de la evolución y la genética para resolver principalmente problemas de optimización.

8) ***Cuerpo celular o soma:*** Contiene al núcleo y ***s***e encarga de todas las actividades metabólicas de la neurona además de que recibe la información de otras neuronas vecinas a través de las conexiones sinápticas.

9) ***Chips neuronales:*** Elemento o conjunto de elementos construidos a partir de circuitos integrados específicos, que tienen como finalidad emular el comportamiento de una red neuronal**.** De tal forma que la estructura del circuito integrado refleja la arquitectura de la red, obteniendo con ello una mayor velocidad, la cual permite en varias ocasiones un proceso en tiempo real.

10) ***Dendritas:*** Son ramas de extensión que parten del cuerpo celular o soma y tienen ramificaciones. Se encargan de la recepción de señales de las otras células a través de conexiones sinápticas.

11) ***Inteligencia artificial:*** Ciencia que intenta la creación de programas para máquinas que imiten el comportamiento y la comprensión humana.

La investigación en el campo de la ***inteligencia artificial*** se caracteriza por la producción de máquinas para la automatización de tareas que requieran un comportamiento inteligente. La ***inteligencia artificial*** se divide en dos escuelas de pensamiento: la inteligencia artificial convencional y la inteligencia computacional.

12) ***Inteligencia computacional:*** Rama de la inteligencia artificial la cual implica desarrollo o aprendizaje iterativo. El aprendizaje se realiza basándose en datos empíricos. Algunos métodos de esta rama incluyen: redes neuronales, sistemas difusos, computación evolutiva entre otros.

13) ***Neocognitron:*** Red neuronal jerárquica multicapa la cual evolucionó a partir de un modelo anterior llamado cognitron, propuesta por Fukushima en 1980.

14) ***Neurona:*** Célula fundamental del sistema nervioso que tiene la capacidad de conectarse con otras neuronas, ya sea para inhibirlas, excitarlas o simplemente para re-transmitirles el impulso nervioso, es decir, la señal electroquímica que viene desde el cerebro, y cuyo destino son las unidades motoras. La ***neurona*** fue descubierta por Santiago Ramón y Cajal en 1888. En general, una neurona consta de las siguientes partes fundamentales: cuerpo celular o soma (el cual contiene al núcleo), las dendritas y el axón. Existen tres tipos de neuronas: sensoriales, motoras e interneuronas.

15) ***Neurona artificial:*** Se denomina ***neurona artificial o procesador elemental*** a un dispositivo simple de cálculo que tiene como función recibir las entradas procedentes del exterior o de otras neuronas, y calcular un valor de salida, el cual es enviado a todas las células restantes.

16) ***Neuronas postsinápticas:*** Con relación a la sinapsis, son las neuronas que envían las señales a otras células nerviosas.

17) ***Neuronas presinápticas:*** Con relación a la sinapsis, son las neuronas que reciben las señales de otras células nerviosas.

18) ***Neurotransmisor:*** Biomolécula, sintetizada generalmente por las neuronas, que se vierte a partir de vesículas existentes en la neurona presináptica, hacia la brecha sináptica y produce un cambio en el potencial de acción de la neurona postsináptica. Los neurotransmisores son por tanto las principales sustancias de las sinapsis.

19) ***Neurocomputadora:*** Son arquitecturas orientadas a la ejecución de procesos con un alto grado de paralelismo.

20) ***Perceptron:*** Primer modelo de red neuronal artificial, desarrollado por el psicólogo **Frank Rosenblatt** en 1958. El ***perceptron*** está constituido por un conjunto de unidades de procesamiento (neuronas artificiales), organizadas en dos capas. La capa de entrada está formada por varias

neuronas lineales, las cuales actúan como sensores del ambiente y capturan los patrones de entrada a la red, mientras que la capa de salida se encuentra constituida por una única neurona.

21) Peso: Parámetro que en las redes neuronales artificiales representa el conocimiento. El aprendizaje resulta de la modificación de estos ***pesos***, que unido al procesamiento de información de la neurona determina el mecanismo básico de la memoria.

22) Peso sináptico: El ***peso sináptico*** $\mathbf{w_{ji}}$ define en este caso la intensidad de interacción entre la neurona presináptica j y la postsináptica i. Dada una entrada positiva, si el ***peso*** es positivo tenderá a excitar a la neurona postsináptica**,** si el ***peso*** es negativo tenderá a inhibirla. Así se habla de sinapsis excitadoras (***peso positivo***) e inhibidoras (***peso negativo***).

23) Red Neuronal Artificial: También denominada sistema neuronal artificial (ANS), son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

24) Sinapsis: Se denomina ***sinapsis*** a la unión entre dos neuronas. Las ***sinapsis*** son direccionales, es decir, la información fluye siempre en un único sentido. En el tipo de ***sinapsis*** más común no existe un contacto físico entre las neuronas, sino que éstas permanecen separadas por un

pequeño vacío de unas 0.2 micras.

25) ***Sistemas difusos:*** Son técnicas para lograr el razonamiento bajo incertidumbre. La lógica difusa o borrosa se basa en lo relativo de lo observado es decir, se toman dos valores aleatorios, pero contextualizados y referidos entre sí.

26) ***Sistemas expertos:*** Son complejos programas de computadoras en los que se codifica el conocimiento de expertos en una materia muy concreta (concesión de créditos, diagnóstico de enfermedades, etc.), en forma de reglas de decisión.

27) ***Terminales axónicos:*** Son las ramificaciones del axón que se encargan de distribuir información a un nuevo conjunto de neuronas.

APÉNDICE B:

NEMÓNICOS

1. ***ADALINE:*** **Adaptative Linear Elements** (Elementos Lineales Adaptativos), red neuronal artificial desarrollada en 1960 por el profesor Bernard Widrow de la Universidad de Stanford.

2. ***ANS:*** **Artificial Neural Systems (**Sistemas Neuronales Artificiales), también denominados redes neuronales artificiales.

3. ***ENIAC:*** **Electronic Numerical Integrator And Computer** (Computadora e Integrador Numérico Electrónico), utilizada por el laboratorio de investigación balística del ejército de los Estados Unidos.

4. ***IA:*** **Inteligencia Artificial,** ciencia que intenta la creación de programas para máquinas que imiten el comportamiento y la comprensión humana.

5. ***IBM:*** **International Business Machines** (Máquinas de Negocios Internacionales), empresa que fabrica y comercializa hardware, software, consultoría y otros servicios relacionados con la informática. Tiene su sede en Armonk, New York, EE UU. y está constituida como tal desde el 15 de junio de 1911, pero lleva operando desde 1888.

6. ***LISP:*** **List Processing** (Procesador de Lista), es el segundo lenguaje de programación de alto nivel después de Fortran. Fue desarrollado en 1958 por John McCarthy y sus colaboradores en el Instituto Massachussets de Tecnología.

7. ***PC:*** **Personal Computer** (Computadora Personal), término genérico normalmente utilizado para referirse a todos los tipos de computadoras personales, no sólo los compatibles con **IBM**. Una computadora personal es generalmente de tamaño medio y es usada por un sólo usuario aunque existen sistemas operativos que permiten varios usuarios simultáneamente.

8. ***VLSI:*** **Very Large Scale Integration** (Integración a Muy Gran Escala), la **VLSI** en sistemas de circuitos comenzó en 1980 con la utilización de transistores en circuitos integrados, como parte de las tecnologías de semiconductores que se estaban desarrollando. Actualmente con el avance de la tecnología ya se cuenta con sistemas integrados que van desde varias decenas de miles hasta varios millones de compuertas en un solo chip.

APÉNDICE C:

PROGRAMA

Para la realización del programa que ilustra el funcionamiento de una ***red neuronal artificial percetron multicapa*** para solucionar el problema de la ***función OR exclusiva,*** se utilizo el compilador Borland C++ 5.5, la interfaz gráfica Relo2.0 y se incluyeron las librerías patron.h, neuronal.h y backprop.h. del material adicional de ***redes neuronales artificiales:*** un enfoque práctico *(Juan Manuel Corchado, 2000).*

```
#include<fstream.h>
#include<stdlib.h>
#include<iomanip.h>
#include<math.h>

#include "patron.h"
#include "neuronal.h"
#include "backprop.h"

char a;
void main( void )
{
// Conjunto de entrenamiento
        Patron *ListaPatrones[4];

                                                // tamaño  Id   entrada    salida
                                                // ------  --   -------    ------
        ListaPatrones[0]=new Patron(2,1,   1,   0.0,0.0,   0.0 );
        ListaPatrones[1]=new Patron(2,1,   2,   0.0,1.0,   1.0 );
        ListaPatrones[2]=new Patron(2,1,   3,   1.0,0.0,   1.0 );
        ListaPatrones[3]=new Patron(2,1,   4,   1.0,1.0,   0.0 );

// Se crea la red Perceptron Multicapa
// (3 capas: 2 neuronas de entrada, 3 intermedias y 1 de salida)
// Ratio de aprendizaje 0.45, momento 0.9
        RedNeuronalBP Red(0.45, 0.9, 3, 2,3,1);
```

```
// Entrenamiento de la red hasta acertar con todos los patrones
        long iteracion = 0;
        int aciertos = 0;
        double tolerancia=0.5;
        double error_total;

        while (aciertos<4)
        {

                aciertos=0;
                error_total=0.0;
                for (int i=0; i<4; i++)
                {
                        Red.Set_Valor(ListaPatrones[i]);

                        Red.Ejecutar(); // Hacia delante

                        Red.Set_Error(ListaPatrones[i]);

                        Red.Aprender(); // Hacia atrás

                        If (fabs(Red.Get_Valor(0)-ListaPatrones[i]->Salidas(0))<tolerancia)
                                aciertos++;

                        error_total+=fabs(Red.Get_Error(0));
                }
                // Visualiza el estado cada 1000 iteraciones
                if (iteracion % 1000 == 0)
                        cout << iteracion << ".   " << aciertos
                                << "/4   Error: " << setprecision(10)
                                << error_total << endl;
                iteracion++;
        }
        cout << endl << endl;

// La red se almacena
        ofstream fichero_salida("backprop.txt");
        Red.Grabar(fichero_salida);
```

```
        fichero_salida.close();

// Se crea y se carga la red
        RedNeuronalBP RedEntrenada;

        ifstream fichero_entrada("backprop.txt");
        RedEntrenada.Cargar(fichero_entrada);
        fichero_entrada.close();

        // Ejecutar la red
        for (int i=0; i<4; i++)
        {
                RedEntrenada.Set_Valor(ListaPatrones[i]);
                RedEntrenada.Ejecutar();

                // Salida de la red redondeada a 0 o 1
                int out=(int)(RedEntrenada.Get_Valor(0)+0.5);

                cout << "Patron: " << setw(3) << i << "  Entrada: ("
                << ListaPatrones[i]->Entradas(0) << ","
                << ListaPatrones[i]->Entradas(1)
                << ")   Valor Red: (" << out
                << ")    Valor del patron: (" << ListaPatrones[i]->Salidas(0)
                << ")" << endl;
                cin>>a;
        }
}
```

BIBLIOGRAFÍA

1) Adenso Díaz, Fred Glover, Hassan M. Ghaziri, J.L. González, Manuel Laguna, Pablo Moscazo, Fan T. Tseng. “Optimización Heurística y Redes Neuronales: En Dirección de Operaciones e Ingeniería”. Edit. Paraninfo, 1996.

2) Bonifacio Martín del Brío, Alfredo Sanz Molina. “Redes Neuronales y Sistemas Difusos”. Edit. Alfaomega, RA-MA, 2001.

3) Ignacio Olmeda, Sergio Barda-Romero. “Redes Neuronales Artificiales: Fundamentos y Aplicaciones”. Edit. Universidad de Alcalá de Henares, Servicios de Publicaciones, 1993.

4) Jacek M. Zurada. “Introducción to Artificial Neural Systems”. Edit. WEST PUBLISHING COMPANY, 1992.

5) James A. Freeman, David M. Skapura,Versión en español de: Rafael García-Bermejo Giner,Con la colaboración de: Luis Joyanes Aguilar. “Redes Neuronales: Algoritmos, Aplicaciones y Técnicas de Programación”. Edit. ADDISON-WESLEY/DIAZ DE SANTOS, 1993.

6) José R. Hilera, Victor J. Martínez. “Redes Neuronales Artificiales: Fundamentos Modelos y Aplicaciones”. Edit. ADDISON-WESLEY IBEROAMERICA, RA-MA, 1995.

7) Juan Manuel Corchado, Fernando Díaz, Lourdes Borrajo, Florentino Fernández. “Redes Neuronales Artificiales: Un Enfoque Práctico”. Edit. Servicios de publicaciones de la Universidad de Vingo, 2000.

8) Luciano Boquete Vázquez, Rafael Barea Navarro. “Control Neuronal”. Edit. Universidad de Alcalá, Servicio de Publicaciones, 1998.

9) Perambur S. Neelakanta, Ph. D., C. Eng. “Information-Theoretic Aspects of Neural Networks”. Edit. CRC Press LLC, 1999.

10) Senén Barro, José Mira. “Computación Neuronal”. Universidad de Santiago de Compostela, Servicios de Publicaciones e Intercambio Científico. Edit. Intercambio Científico, 1995.