


Libro-Redes neuronales

Holman Albino

Related papers

[Download a PDF Pack](#) of the best related papers 



[Redes Neuronales Artificiales y sus Aplicaciones](#)

Ronald Mora

[REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES Tabla de Contenidos](#)

Daniel Guillermo Garcia Murillo

[REDES NEURONALES ARTIFICIALES.pdf](#)

Pablo Antonio Martinez

REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES



Xabier Basogain Olabe

Centro: Escuela Superior de Ingeniería de Bilbao, EHU

Despacho: P3BN11

Teléfono: 34 946014201

E-mail: xabier.basogain@ehu.es



Tabla de Contenidos

TEMA 1.- INTRODUCCIÓN A LA COMPUTACIÓN NEURONAL	1
1.1.- INTRODUCCIÓN	1
1.2.- CARACTERÍSTICAS DE LAS REDES NEURONALES ARTIFICIALES	2
1.3.- ESTRUCTURA BÁSICA DE UNA RED NEURONAL	2
1.4.- COMPUTACIÓN TRADICIONAL Y COMPUTACIÓN NEURONAL	4
1.5.- HISTORIA DE LA COMPUTACIÓN NEURONAL	6
1.6.- APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES	9
1.7.- IMPLEMENTACIÓN Y TECNOLOGÍAS EMERGENTES	11
TEMA 2.- FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES	13
2.1.- EL PROTOTIPO BIOLÓGICO	13
2.2.- LA NEURONA ARTIFICIAL	14
2.3.- REDES NEURONALES ARTIFICIALES DE UNA CAPA Y MULTICAPA	15
2.4.- ENTRENAMIENTO DE LAS REDES NEURONALES ARTIFICIALES	17
TEMA 3.- SELECCIÓN DE LAS REDES NEURONALES ARTIFICIALES	19
TEMA 4.- LAS PRIMERAS REDES NEURONALES ARTIFICIALES	22
4.1.- PERCEPTRON	22
4.2.- ADALINE – MADALINE	25
TEMA 5.- RED BACKPROPAGATION	28
5.1.- INTRODUCCIÓN	28
5.2.- ARQUITECTURA DE LA RED BACKPROPAGATION	29
5.3.- ALGORITMO DE ENTRENAMIENTO	30
5.4.- APLICACIONES DE LA RED BACKPROPAGATION	33
5.5.- VENTAJAS E INCONVENIENTES	34
TEMA 6.- RED SELF ORGANIZING MAP Y RED COUNTERPROPAGATION	35
6.1.- INTRODUCCIÓN RED SELF ORGANIZING MAP	35
6.2.- ARQUITECTURA BÁSICA Y MODO DE OPERACIÓN	36
6.3.- EJEMPLOS RED S.O.M.	37
6.4.- INTRODUCCIÓN RED COUNTERPROPAGATION	38
6.5.- ARQUITECTURA Y FUNCIONAMIENTO	39
6.6.- EJEMPLOS RED COUNTERPROPAGATION	41
TEMA 7.- RED HOPFIELD Y RED BIDIRECTIONAL ASSOCIATIVE MEMORY	43
7.1.- RED HOPFIELD	43
7.2.- APLICACIONES DE LA RED HOPFIELD	47
7.3.- VENTAJAS Y LIMITACIONES	49
7.4.- INTRODUCCIÓN RED BIDIRECTIONAL ASSOCIATIVE MEMORY	50
7.5.- ARQUITECTURA RED B.A.M.	50
TEMA 8.- RED ADAPTIVE RESONANCE THEORY	53
8.1.- INTRODUCCIÓN RED ADAPTIVE RESONANCE THEORY	53
8.2.- ARQUITECTURA RED A.R.T.	54
8.3.- MODO DE OPERACIÓN	56
8.4.- ENTRENAMIENTO DE LA RED A.R.T.	57
TEMA 9.- APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES	58
9.1.- INTRODUCCIÓN	58
9.2.- DISEÑO DE UNA RED PARA UNA APLICACIÓN	59
9.3.- EJEMPLOS DE APLICACIONES	59

TEMA 10.- LÓGICA DIFUSA Y REDES NEURONALES ARTIFICIALES	65
10.1.- INTRODUCCIÓN	65
10.2.- ESTRUCTURA GENERAL DE UN SISTEMA BASADO EN LÓGICA BORROSA	67
10.3.- SISTEMAS NEURO- DIFUSOS	71
 BIBLIOGRAFÍA	73
Libros complementarios del curso	73
Libros de interés y consultados para la elaboración del curso	74

INTRODUCCIÓN A LA COMPUTACIÓN NEURONAL

1

- 1.1. Introducción
- 1.2. Características de las Redes Neuronales Artificiales
- 1.3. Estructura Básica de una Red Neuronal
- 1.4. Computación Tradicional y Computación Neuronal
- 1.5. Historia de la Computación Neuronal
- 1.6. Aplicaciones de las Redes Neuronales Artificiales
- 1.7. Implementación y Tecnologías Emergentes

TEMA 1.- INTRODUCCIÓN A LA COMPUTACIÓN NEURONAL

1.1.- INTRODUCCIÓN

El cerebro humano es el sistema de cálculo más complejo que conoce el hombre. El ordenador y el hombre realizan bien diferentes clases de tareas; así la operación de reconocer el rostro de una persona resulta una tarea relativamente sencilla para el hombre y difícil para el ordenador, mientras que la contabilidad de una empresa es tarea costosa para un experto contable y una sencilla rutina para un ordenador básico.

La capacidad del cerebro humano de pensar, recordar y resolver problemas ha inspirado a muchos científicos intentar o procurar modelar en el ordenador el funcionamiento del cerebro humano.

Los profesionales de diferentes campos como la ingeniería, filosofía, fisiología y psicología han unido sus esfuerzos debido al potencial que ofrece esta tecnología y están encontrando diferentes aplicaciones en sus respectivas profesiones.

Un grupo de investigadores ha perseguido la creación de un modelo en el ordenador que iguale o adopte las distintas funciones básicas del cerebro. El resultado ha sido una nueva tecnología llamada Computación Neuronal o también Redes Neuronales Artificiales.

El resurgimiento del interés en esta nueva forma de realizar los cálculos tras dos décadas de olvido se debe al extraordinario avance y éxito tanto en el aspecto teórico como de aplicación que se está obteniendo estos últimos años.

1.2.- CARACTERÍSTICAS DE LAS REDES NEURONALES ARTIFICIALES

Las Redes Neuronales Artificiales, ANN (*Artificial Neural Networks*) están inspiradas en las redes neuronales biológicas del cerebro humano. Están constituidas por elementos que se comportan de forma similar a la neurona biológica en sus funciones más comunes. Estos elementos están organizados de una forma parecida a la que presenta el cerebro humano.

Las ANN al margen de "parecerse" al cerebro presentan una serie de características propias del cerebro. Por ejemplo las ANN aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos.

Aprender: adquirir el conocimiento de una cosa por medio del estudio, ejercicio o experiencia. Las ANN pueden cambiar su comportamiento en función del entorno. Se les muestra un conjunto de entradas y ellas mismas se ajustan para producir unas salidas consistentes.

Generalizar: extender o ampliar una cosa. Las ANN generalizan automáticamente debido a su propia estructura y naturaleza. Estas redes pueden ofrecer, dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos de ruido o distorsión.

Abstraer: aislar mentalmente o considerar por separado las cualidades de un objeto. Algunas ANN son capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes o relativos.

1.3.- ESTRUCTURA BÁSICA DE UNA RED NEURONAL

Analogía con el cerebro.-

La neurona es la unidad fundamental del sistema nervioso y en particular del cerebro. Cada neurona es una simple unidad procesadora que recibe y combina señales desde y hacia otras neuronas. Si la combinación de entradas es suficientemente fuerte la salida de la neurona se activa. La Figura (1.1) muestra las partes que constituyen una neurona.

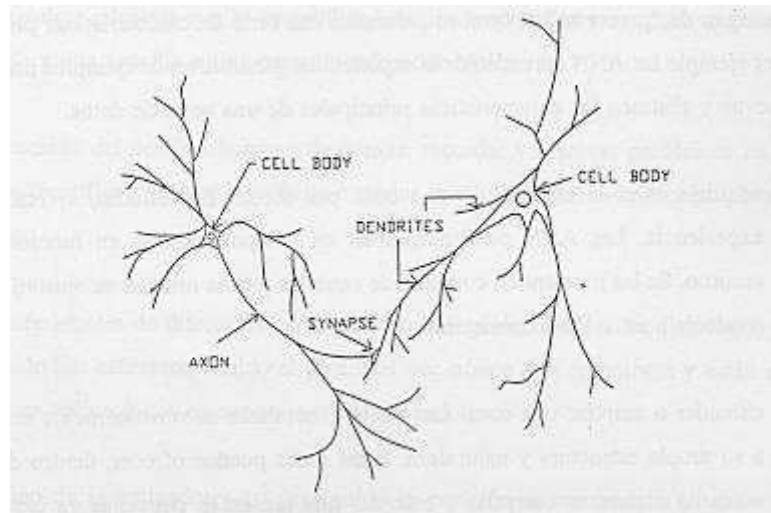


Figura (1.1) - Componentes de una Neurona.

El cerebro consiste en uno o varios billones de neuronas densamente interconectadas. El axón (salida) de la neurona se ramifica y está conectada a las dendritas (entradas) de otras neuronas a través de uniones llamadas sinapsis. La eficacia de la sinapsis es modificable durante el proceso de aprendizaje de la red.

Redes Neuronales Artificiales.-

En las Redes Neuronales Artificiales, ANN, la unidad análoga a la neurona biológica es el elemento procesador, PE (*process element*). Un elemento procesador tiene varias entradas y las combina, normalmente con una suma básica. La suma de las entradas es modificada por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del elemento procesador.

La salida del PE se puede conectar a las entradas de otras neuronas artificiales (PE) mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis de las conexiones neuronales.

La Figura (1.2) representa un elemento procesador de una red neuronal artificial implementada en un ordenador.

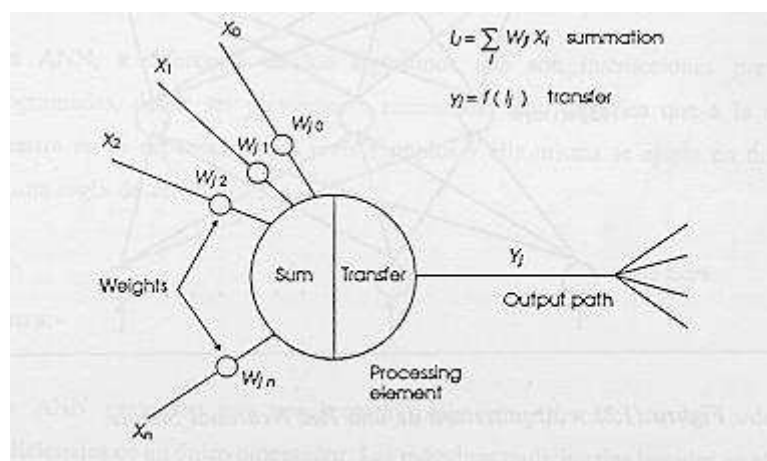


Figura (1.2) - Diagrama de una Neurona Artificial (PE).

Una red neuronal consiste en un conjunto de unidades elementales PE conectadas de una forma concreta. El interés de las ANN no reside sólo en el modelo del elemento PE sino en las formas en que se conectan estos elementos procesadores. Generalmente los elementos PE están organizados en grupos llamados niveles o capas. Una red típica consiste en una secuencia de capas con conexiones entre capas adyacentes consecutivas.

Existen dos capas con conexiones con el mundo exterior. Una capa de entrada, buffer de entrada, donde se presentan los datos a la red, y una capa buffer de salida que mantiene la respuesta de la red a una entrada. El resto de las capas reciben el nombre de capas ocultas. La Figura (1.3) muestra el aspecto de una Red Neuronal Artificial.

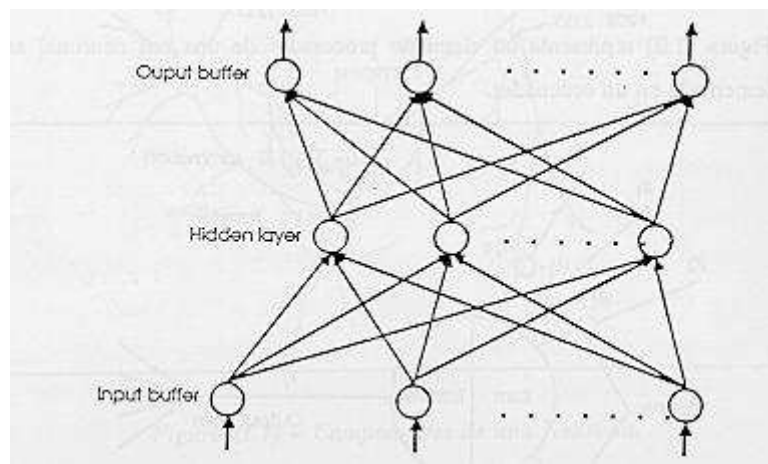


Figura (1.3) - Arquitectura de una Red Neuronal Simple.

1.4.- COMPUTACIÓN TRADICIONAL Y COMPUTACIÓN NEURONAL

Programación/Entrenamiento.-

Las técnicas tradicionales de programación utilizadas para la solución de un problema requieren la creación de un algoritmo. Un algoritmo consiste en una secuencia de instrucciones que indica el modo en el que debe proceder el sistema basado en un ordenador para lograr el fin perseguido que es la resolución del problema.

El diseño de una secuencia de instrucciones para resolver un problema de contabilidad es relativamente sencillo, mientras que existen muchos problemas del mundo real en los que resulta difícil realizar un algoritmo que resuelva dichos problemas. Por ejemplo imaginemos desarrollar un programa para cualquiera de los problemas de reconocimiento de imágenes como el rostro de una persona. Hay muchas variaciones de la imagen de una persona, como que presente un rostro serio o un rostro alegre, variaciones en general que deben tenerse en cuenta a la hora de diseñar el algoritmo.

Las ANN, a diferencia de los algoritmos que son instrucciones previamente programadas, deben ser previamente entrenadas. Esto significa que a la red se le muestra en su capa de entrada unos ejemplos y ella misma se ajusta en función de alguna regla de aprendizaje.

Arquitectura.-

Las ANN presentan una arquitectura totalmente diferente de los ordenadores tradicionales de un único procesador. Las máquinas tradicionales basadas en el modelo de Von Neuman tienen un único elemento procesador, la CPU (*Control Process Unit*) que realiza todos los cálculos ejecutando todas las instrucciones de la secuencia programada en el algoritmo. Cualquier CPU realiza más de cien comandos básicos, incluyendo sumas, restas, y desplazamientos entre otros.

Los comandos o instrucciones se ejecutan secuencialmente y sincronizadas con el reloj del sistema. Sin embargo en los sistemas de computación neuronal cada elemento PE sólo puede realizar uno, o como mucho, varios cálculos. La potencia del procesamiento de las ANN se mide principalmente por el número de interconexiones actualizadas por segundo durante el proceso de entrenamiento o aprendizaje. Sin embargo las máquinas de Von Neuman se miden por el número de instrucciones que ejecuta por segundo el procesador central CPU.

La arquitectura de las ANN parte de la organización de los sistemas de procesamiento en paralelo, es decir, sistemas en los que distintos procesadores están interconectados. No obstante los procesadores son unidades procesadoras simples, diseñadas para la suma de muchas entradas y con un ajuste automático de las conexiones ponderadas.

Sistemas Expertos.-

Los sistemas expertos difieren de la programación tradicional en que la base del conocimiento está separada del motor de inferencia (el método del procesamiento del conocimiento). Esta característica permite que todo el conocimiento adicional puede ser añadido al sistema sin necesidad de tener que ser reprogramado todo el sistema. Esta técnica requiere que exista una persona experta en un área y que se puedan crear reglas que codifiquen el conocimiento.

En el desarrollo de una red neuronal no hay que programar ni el conocimiento ni las reglas del procesamiento del conocimiento. La red neuronal aprende las reglas del procesamiento del conocimiento mediante el ajuste de las conexiones ponderadas entre las neuronas de distintas capas de la red.

Mientras que en los Sistemas Expertos el conocimiento se hace explícito en forma de reglas, en la computación neuronal las ANN generan sus propias reglas aprendiendo de los ejemplos que se les muestran en la fase de entrenamiento. El aprendizaje se consigue a través de una regla de aprendizaje que adapta o cambia los pesos de las conexiones en respuesta a los ejemplos de entrada, y opcionalmente también en respuesta a las salidas deseadas. Esta característica de las ANN es lo que permite decir que las redes neuronales aprenden de la experiencia.

Una característica importante de las ANN es la forma o el modo en que se almacena la información. La memoria o el conocimiento de estas redes está distribuida a lo largo de todas las conexiones ponderadas de la red.

Algunas ANN presentan la característica de ser "asociativas" que significa que para una entrada parcial la red elegirá la entrada más parecida en memoria y generará una salida que corresponda a la entrada completa.

La naturaleza de la memoria de las ANN permite que la red responda adecuadamente cuando se le presenta una entrada incompleta o con ruido. Esta propiedad suele ser referida como la capacidad de "generalización".

Otra característica de las ANN es la tolerancia a la falta (*Fault Tolerance*). Tolerancia a la falta se refiere al hecho de que en muchas ANN si resultaran destruidos varios elementos procesadores PE, o se alteraran las conexiones el comportamiento de la red sería mínimamente modificado. El comportamiento varía pero el sistema no se descompone o deja de funcionar.

Esta característica se debe a que las ANN tienen la información distribuida a lo largo de toda la red y no está contenida en un único lugar.

1.5.- HISTORIA DE LA COMPUTACIÓN NEURONAL

En 1943, el neurobiólogo Warren McCulloch, y el estadístico Walter Pitts, publicaron el artículo "*A logical calculus of Ideas Imminent in Nervous Activity*". Este artículo constituyó la base y el inicio del desarrollo en diferentes campos como son los Ordenadores Digitales (John Von Neuman), la Inteligencia Artificial (Marvin Minsky con los Sistemas Expertos) y el funcionamiento del ojo (Frank Rosenblatt con la famosa red llamada Perceptron).

En 1956, los pioneros de la Inteligencia Artificial, Minsky, McCarthy, Rochester, Shanon, organizaron la primera conferencia de Inteligencia Artificial que fue patrocinada por la Fundación Rochester. Esta conferencia se celebró en el verano de 1956 en la localidad inglesa de Darmouth y en muchos libros se hace referencia al verano de este año como la primera toma de contacto seria con las redes neuronales artificiales.

Nathaural Rochester del equipo de investigación de IBM presentó el modelo de una red neuronal que él mismo realizó y puede considerarse como el primer software de simulación de redes neuronales artificiales.

En 1957, Frank Rosenblatt publicó el mayor trabajo de investigación en computación neuronal realizado hasta esas fechas. Su trabajo consistía en el desarrollo de un elemento llamado "**Perceptron**".

El perceptron es un sistema clasificador de patrones que puede identificar patrones geométricos y abstractos. El primer perceptron era capaz de aprender algo y era robusto, de forma que su comportamiento variaba sólo si resultaban dañados los componentes

del sistema. Además presentaba la característica de ser flexible y comportarse correctamente después de que algunas celdas fueran destruidas.

El perceptron fue originalmente diseñado para el reconocimiento óptico de patrones. Una rejilla de 400 fotocélulas, correspondientes a las neuronas de la retina sensibles a la luz, recibe el estímulo óptico. Estas fotocélulas están conectadas a elementos asociativos que recogen los impulsos eléctricos emitidos desde las fotocélulas. Las conexiones entre los elementos asociativos y las fotocélulas se realizan de forma aleatoria.

Si las células presentan un valor de entrada superior a un umbral predeterminado entonces el elemento asociativo produce una salida. La Figura (1.4) presenta la estructura de la red perceptron.

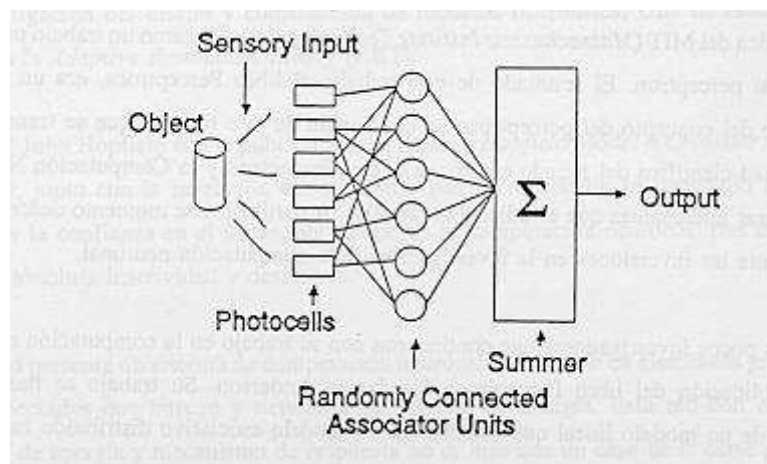


Figura (1.4) - Aplicación de la Red Perceptron.

El perceptron presenta algunas limitaciones debido a que se trataba de un dispositivo en desarrollo. La mayor limitación la reflejaron Minsky y Papert años más tarde, y ponían de manifiesto la incapacidad del perceptron en resolver algunas tareas o problemas sencillos como por ejemplo la función lógica OR exclusivo.

Uno de los mayores cambios realizados en el perceptron de Rosenblatt a lo largo de la década de los 60 ha sido el desarrollo de sistemas multicapa que pueden aprender y categorizar datos complejos.

En 1959, Bernard Widrow en Stanford desarrolló un elemento adaptativo lineal llamado "**Adaline**" (*Adaptive Linear Neuron*). La Adaline y una versión de dos capas, llamada "**Madaline**", fueron utilizadas en distintas aplicaciones como reconocimiento de voz y caracteres, predicción del tiempo, control adaptativo y sobre todo en el desarrollo de filtros adaptativos que eliminen los ecos de las líneas telefónicas.

A mediados de los años 60, Minsky y Papert pertenecientes al Laboratorio de Investigación de Electrónica del MIT (*Massachusetts Institute Technology*) comenzaron un trabajo profundo de crítica al perceptron. El resultado de este trabajo, el libro *Perceptrons*, era un análisis matemático del concepto del perceptron. La conclusión de este trabajo, que se transmitió a la comunidad científica del mundo entero, es que el

Perceptron y la Computación Neuronal no eran temas interesantes que estudiar y desarrollar. A partir de este momento descendieron drásticamente las inversiones en la investigación de la computación neuronal.

Uno de los pocos investigadores que continuaron con su trabajo en la computación neuronal tras la publicación del libro *Perceptrons* fue James Anderson. Su trabajo se basó en el desarrollo de un modelo lineal que consiste en un modelo asociativo distribuido basado en el principio de Hebb (las conexiones son reforzadas cada vez que son activadas las neuronas). Una versión extendida de este modelo lineal es el llamado modelo *Brain-State-in- a Box* (BSB).

Teuvo Kohonen, de la Universidad de Helsinki, es uno de los mayores impulsores de la computación neuronal de la década de los 70. De su trabajo de investigación destacan dos aportaciones: la primera es la descripción y análisis de una clase grande de reglas adaptativas, reglas en las que las conexiones ponderadas se modifican de una forma dependiente de los valores anteriores y posteriores de las sinapsis. Y la segunda aportación es el principio de aprendizaje competitivo en el que los elementos compiten por responder a un estímulo de entrada, y el ganador se adapta él mismo para responder con mayor efecto al estímulo.

Otro investigador que continuó con su trabajo de investigación en el mundo de la computación neuronal a pesar del mal presagio que indicaron Minsky y Papert fue Stephen Grossberg. Grossberg estaba especialmente interesado en la utilización de datos de la neurología para construir modelos de computación neuronal. La mayoría de sus reglas y postulados derivaron de estudios fisiológicos. Su trabajo ha constituido un gran impulso en la investigación del diseño y construcción de modelos neuronales. Una de estas clases de redes es la *Adaptive Resonance Theory* (ART).

En 1982 John Hopfield con la publicación del artículo *Hopfield Model* o *Crossbar Associative Network*, junto con la invención del algoritmo **Backpropagation** se consiguió devolver el interés y la confianza en el fascinante campo de la computación neuronal tras dos décadas de casi absoluta inactividad y desinterés.

Hopfield presenta un sistema de computación neuronal consistente en elementos procesadores interconectados que buscan y tienden a un mínimo de energía. Esta red con este tipo de función de energía y mecanismo de respuesta no es mas que un caso de la clase genérica de redes que consideró Grossberg.

Investigación de hoy en día.-

Existen muchos grupos con sede en diferentes universidades de todo el mundo que están realizando trabajos de investigación en el área de las redes neuronales artificiales. Cada grupo tiene diferente énfasis y motivación con los neurólogos, psicólogos del conocimiento, físicos, programadores y matemáticos. Todos ellos ofrecen nuevos puntos de vista e intuiciones en esta área de la técnica.

Grossberg continua trabajando en compañía de Carpenter en la Universidad de Boston, mientras Teuvo Kohonen está en la Universidad de Helsinki. Uno de los mayores

grupos de investigación de los últimos años ha sido el grupo PDP (*Parallel Distributed Processing*) formado por Rumelhart, McClelland y Hinton.

Rumelhart de la Universidad de Stanford es uno de los principales impulsores de la red más utilizada en la mayoría de las aplicaciones actuales, la famosa red neuronal Backpropagation. En la Universidad de Carnegie-Mellon, el grupo de investigación a la cabeza con McClelland destaca por el estudio de las posibles aplicaciones de la Backpropagation. Y en la Universidad de Toronto, Hinton y Sejnowski han desarrollado una máquina llamada Boltzman que consiste en la red de Hopfield con dos modificaciones significativas.

Bart Kosko ha diseñado una red llamada BAM (*Bidirectional Associate Memory*) basado en la red de Grossberg. Por último indicar la existencia de grandes grupos de investigación como los de California Institute of Technology, Massachussets Institute of Technology, University of California Berkeley y University of California San Diego.

Conviene no olvidar el esfuerzo económico y técnico que están realizando las empresas privadas tanto en USA como en Japón y en la Comunidad Económica Europea. Como botón de muestra de las inversiones en estos países baste conocer que sólo en USA se gasta más de 100 millones de dólares al año.

1.6.- APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES

Las características especiales de los sistemas de computación neuronal permiten que sea utilizada esta nueva técnica de cálculo en una extensa variedad de aplicaciones.

La computación neuronal provee un acercamiento mayor al reconocimiento y percepción humana que los métodos tradicionales de cálculo. Las redes neuronales artificiales presentan resultados razonables en aplicaciones donde las entradas presentan ruido o las entradas están incompletas. Algunas de las áreas de aplicación de las ANN son las siguientes:

Análisis y Procesado de señales	Reconocimiento de Imágenes
Control de Procesos	Filtrado de ruido
Robótica	Procesado del Lenguaje
Diagnósticos médicos	Otros

Conversión Texto a Voz: uno de los principales promotores de la computación neuronal en esta área es Terrence Sejnowski. La conversión texto-voz consiste en cambiar los símbolos gráficos de un texto en lenguaje hablado. El sistema de computación neuronal presentado por Sejnowski y Rosemberg, el sistema llamado

NetTalk, convierte texto en fonemas y con la ayuda de un sintetizador de voz (Dectalk) genera voz a partir de un texto escrito.

La ventaja que ofrece la computación neuronal frente a las tecnologías tradicionales en la conversión texto-voz es la propiedad de eliminar la necesidad de programar un complejo conjunto de reglas de pronunciación en el ordenador. A pesar de que el sistema NetTalk ofrece un buen comportamiento, la computación neuronal para este tipo de aplicación abre posibilidades de investigación y expectativas de desarrollo comercial.

Procesado Natural del Lenguaje: incluye el estudio de cómo se construyen las reglas del lenguaje. Los científicos del conocimiento Rumelhart y McClelland han integrado una red neuronal de proceso natural del lenguaje. El sistema realizado ha aprendido el tiempo verbal *pass tense* de los verbos en Inglés. Las características propias de la computación neuronal como la capacidad de generalizar a partir de datos incompletos y la capacidad de abstraer, permiten al sistema generar buenos pronósticos para verbos nuevos o verbos desconocidos.

Compresión de Imágenes: la compresión de imágenes es la transformación de los datos de una imagen a una representación diferente que requiera menos memoria o que se pueda reconstruir una imagen imperceptible. Cottrel, Munro y Zisper de la Universidad de San Diego y Pittsburgh han diseñado un sistema de compresión de imágenes utilizando una red neuronal con un factor de compresión de 8:1.

Reconocimiento de Caracteres: es el proceso de interpretación visual y de clasificación de símbolos. Los investigadores de Nestor, Inc. han desarrollado un sistema de computación neuronal que tras el entrenamiento con un conjunto de tipos de caracteres de letras, es capaz de interpretar un tipo de carácter o letra que no haya visto con anterioridad.

Reconocimiento de Patrones en Imágenes: una aplicación típica es la clasificación de objetivos detectados por un sonar. Existen varias ANN basadas en la popular Backpropagation cuyo comportamiento es comparable con el de los operadores humanos. Otra aplicación normal es la inspección industrial.

Problemas de Combinatoria: en este tipo de problemas la solución mediante cálculo tradicional requiere un tiempo de proceso (CPU) que es exponencial con el número de entradas. Un ejemplo es el problema del vendedor; el objetivo es elegir el camino más corto posible que debe realizar el vendedor para cubrir un número limitado de ciudades en una área geográfica específica. Este tipo de problema ha sido abordado con éxito por Hopfield y el resultado de su trabajo ha sido el desarrollo de una ANN que ofrece buenos resultados para este problema de combinatoria.

Procesado de la Señal: en este tipo de aplicación existen tres clases diferentes de procesado de la señal que han sido objeto de las ANN como son la predicción, el modelado de un sistema y el filtrado de ruido.

Predicción: en el mundo real existen muchos fenómenos de los que conocemos su comportamiento a través de una serie temporal de datos o valores. Lapedes y Farber del Laboratorio de Investigación de los Álamos, han demostrado que la red backpropagation supera en un orden de magnitud a los métodos de predicción polinómicos y lineales convencionales para las series temporales caóticas.

Modelado de Sistemas: los sistemas lineales son caracterizados por la función de transferencia que no es más que una expresión analítica entre la variable de salida y una variable independiente y sus derivadas. Las ANN también son capaces de aprender una función de transferencia y comportarse correctamente como el sistema lineal que está modelando.

Filtro de Ruido: las ANN también pueden ser utilizadas para eliminar el ruido de una señal. Estas redes son capaces de mantener en un alto grado las estructuras y valores de los filtros tradicionales.

Modelos Económicos y Financieros: una de las aplicaciones más importantes del modelado y pronóstico es la creación de pronósticos económicos como por ejemplo los precios de existencias, la producción de las cosechas, el interés de las cuentas, el volumen de las ventas etc. Las redes neuronales están ofreciendo mejores resultados en los pronósticos financieros que los métodos convencionales.

ServoControl: un problema difícil en el control de un complejo sistema de servomecanismo es encontrar un método de cálculo computacional aceptable para compensar las variaciones físicas que se producen en el sistema. Entre los inconvenientes destaca la imposibilidad en algunos casos de medir con exactitud las variaciones producidas y el excesivo tiempo de cálculo requerido para la obtención de la solución matemática. Existen diferentes redes neuronales que han sido entrenadas para reproducir o predecir el error que se produce en la posición final de un robot. Este error se combina con la posición deseada para proveer una posición adaptativa de corrección y mejorar la exactitud de la posición final.

1.7.- IMPLEMENTACIÓN Y TECNOLOGÍAS EMERGENTES

El resurgimiento de la computación neuronal en los últimos años se ha producido por el desarrollo teórico de nuevos modelos matemáticos del comportamiento del cerebro y por el desarrollo de nuevas tecnologías que ya están siendo utilizadas en una gran variedad de aplicaciones comerciales.

Entre los avances o desarrollos tecnológicos que permiten la realización de la computación neuronal destacan los programas software de simulación, los aceleradores hardware, los chips de silicio y los procesadores ópticos.

Simuladores Software: constituyen una de las formas más versátiles con las que se pueden implementar redes neuronales. Estos programas constituyen todo un sistema de desarrollo y realización de prototipos de redes neuronales. Estos programas se

utilizan para diseñar, construir, entrenar y probar redes neuronales artificiales para resolver problemas complejos y problemas del mundo real.

Los primeros simuladores software se ejecutaban en ordenadores de grandes prestaciones y el avance de los ordenadores personales en capacidad de procesado y capacidad de memoria hace posible que exista una serie de simuladores software de grandes prestaciones que corren sobre ordenadores personales. Entre otros paquetes software se incluye Neural Works, Neuralyst, Explore Net y Knowledge Net.

Aceleradores Hardware: la naturaleza paralela de la computación neuronal se presta a realizar diseños concretos y a medida de dispositivos físicos, aceleradores hardware, que aceleren la ejecución de los cálculos. Los aceleradores hardware para los sistemas de computación neuronal son dispositivos físicos constituidos por diferentes procesadores interconectados que ayudan a la realización y ejecución del comportamiento de las ANN. Una de las ventajas de los aceleradores hardware diseñados específicamente para la computación neuronal es el aumento de la velocidad de procesado. Esta característica permite la utilización de las ANN en aplicaciones de tiempo real.

Robert Hecht-Nielsen desarrolló el acelerador hardware Mark III que constaba de 8100 procesadores y trabajaba como un periférico de un VAX. La mayoría de las casas comerciales dedicadas al diseño de las ANN han desarrollado diferentes tarjetas basadas en los diferentes procesadores existentes, diseñadas para trabajar en el entorno de un ordenador personal PC y presentando un progresivo ratio de actualizaciones de interconexiones por segundo.

Chips de Silicio: Otro de los campos de la investigación en el mundo de las ANN al margen de los simuladores software y aceleradores hardware, es la integración de todos los componentes de computación neuronal en un chip de silicio. Un ejemplo concreto es el chip Electronic Neural Network (EEN) de la compañía AT&T que contiene 256 transistores-neuronas y más de 100.000 resistencias-sinapsis. Actualmente este chip está siendo utilizado para aplicaciones de compresión del ancho de banda de imágenes de vídeo para poder ser transmitidas por una línea telefónica. Existen muchas compañías y centros de investigación que están trabajando en el desarrollo de circuitos integrados que realizan computación neuronal. La mayoría de las aplicaciones de estos chips está siendo la simulación de procesos sensitivos como la visión de imágenes y la audición de sonidos.

FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES

2

- 2.1. El Prototipo Biológico
- 2.2. La Neurona Artificial
- 2.3. Redes Neuronales Artificiales de una capa y Multicapa
- 2.4. Entrenamiento de las Redes Neuronales Artificiales

TEMA 2.- FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES

2.1.- EL PROTOTIPO BIOLÓGICO

Las diferentes configuraciones y algoritmos que se diseñan para las redes neuronales artificiales están inspiradas en la organización del complejo sistema neuronal del cerebro humano. No obstante conviene aclarar que esta inspiración no supone que las ANN lleguen a emular al cerebro como algunos optimistas lo desean ya que entre otras limitaciones el conocimiento sobre el modo de funcionamiento y comportamiento del cerebro es bastante simple y reducido. De hecho los diseñadores de redes artificiales van más lejos del conocimiento biológico actual y prueban nuevas estructuras que presentan un comportamiento adecuado y útil.

El sistema nervioso humano constituido por células llamadas neuronas presenta una estructura muy compleja. El número estimado de neuronas es de 10^{11} y las interconexiones entre ellas son del orden de 10^{15} .

Cada neurona comparte muchas características con otras células del cuerpo humano pero tiene propiedades particulares y especiales para recibir, procesar y transmitir señales electroquímicas a través de todas las interconexiones del sistema de comunicación del cerebro.

La Figura (2.1) muestra la estructura de un par de neuronas biológicas. Del cuerpo de la neurona se extienden las dendritas hacia otras neuronas donde reciben las señales transmitidas por otras neuronas. El punto de contacto o de conexión se llama sinapsis y

estas entradas son dirigidas al núcleo donde se suman. Algunas de las entradas tienden a excitar a la célula y otras sin embargo tienden a inhibir la célula. Cuando la excitación acumulada supera un valor umbral, las neuronas envían una señal a través del axón a otras neuronas.

La mayoría de los modelos de las ANN presenta este funcionamiento básico de la neurona aun cuando el comportamiento real de una célula nerviosa tiene muchas complejidades y excepciones.

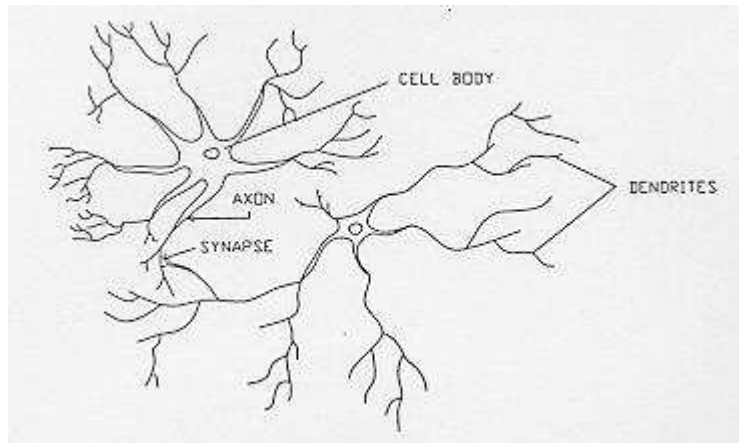


Figura (2.1) - Componentes de una Neurona.

2.2.- LA NEURONA ARTIFICIAL

La neurona artificial fue diseñada para "emular" las características del funcionamiento básico de la neurona biológica. En esencia, se aplica un conjunto de entradas a la neurona, cada una de las cuales representa una salida de otra neurona. Cada entrada se multiplica por su "peso" o ponderación correspondiente análogo al grado de conexión de la sinapsis. Todas las entradas ponderadas se suman y se determina el nivel de excitación o activación de la neurona. Una representación vectorial del funcionamiento básico de una neurona artificial se indica según la siguiente expresión de la ecuación (2.1).

$$NET = X * W \quad \text{ecu.(2.1)}$$

Siendo NET la salida, X el vector de entrada y W el vector de pesos.

Normalmente la señal de salida NET suele ser procesada por una función de activación F para producir la señal de salida de la neurona OUT. La función F puede ser una función lineal, o una función umbral o una función no lineal que simula con mayor exactitud las características de transferencia no lineales de las neuronas biológicas.

La Figura (2.2) representa una neurona artificial con una función de activación F .

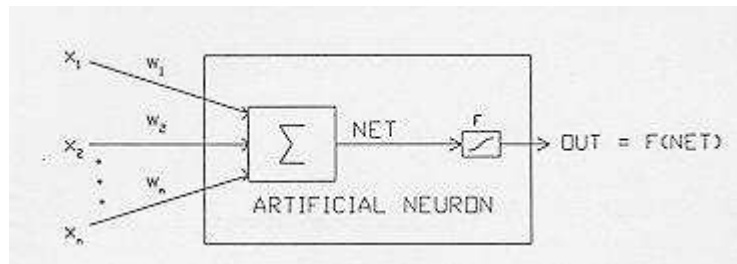


Figura (2.2) - Modelo de Neurona Artificial .

Las funciones F más utilizadas son la función Sigmoid y Tangente hiperbólica expresadas en la Tabla (2.1).

Sigmoid	$OUT = 1 / (1 + e^{-NET})$
Tangente hiperbólica	$OUT = \tanh (NET)$

Tabla 2.1 - Funciones de Activación

Este tipo de modelo de neurona artificial ignora muchas de las características de las neuronas biológicas. Entre ellas destaca la omisión de retardos y de sincronismo en la generación de la salida. No obstante, a pesar de estas limitaciones las redes construidas con este tipo de neurona artificial presentan cualidades y atributos con cierta similitud a la de los sistemas biológicos.

2.3.- REDES NEURONALES ARTIFICIALES DE UNA CAPA Y MULTICAPA

La capacidad de cálculo y potencia de la computación neuronal proviene de las múltiples conexiones de las neuronas artificiales que constituyen las redes ANN.

La red más simple es un grupo de neuronas ordenadas en una capa como se muestra en la Figura (2.3). Los nodos circulares sólo son distribuidores de las entradas y no se consideran constituyentes de una capa.

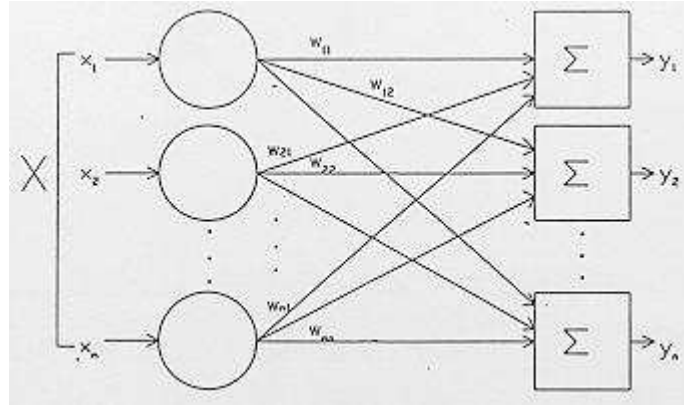


Figura (2.3) - Red Neuronal de una Capa.

Cada una de las entradas está conectada a través de su peso correspondiente a cada neurona artificial. En la práctica existen conexiones eliminadas e incluso conexiones entre las salidas y entradas de las neuronas de una capa. No obstante la figura muestra una conectividad total por razones de generalización.

Normalmente las redes más complejas y más grandes ofrecen mejores prestaciones en el cálculo computacional que las redes simples. Las configuraciones de las redes construidas presentan aspectos muy diferentes pero tienen un aspecto común, el ordenamiento de las neuronas en capas o niveles imitando la estructura de capas que presenta el cerebro en algunas partes.

Las redes multicapa se forman con un grupo de capas simples en cascada. La salida de una capa es la entrada de la siguiente capa. Se ha demostrado que las redes multicapa presentan cualidades y aspectos por encima de las redes de una capa simple. La Figura (2.4) muestra una red de dos capas.

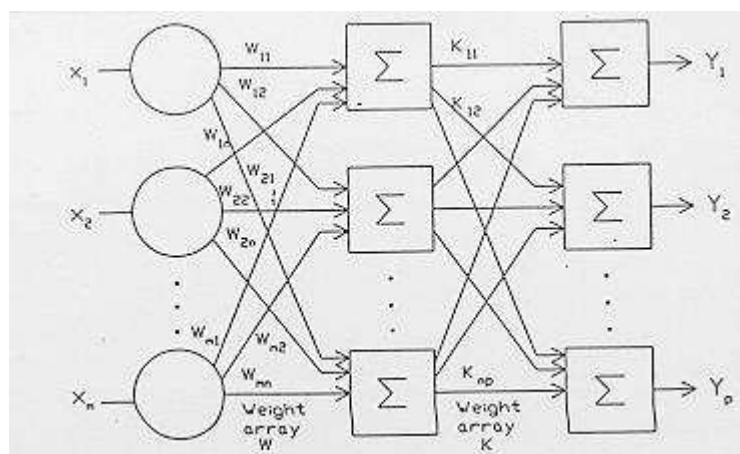


Figura (2.4) - Red Neuronal de dos Capas.

Conviene destacar que la mejora de las redes multicapa estriba en la función de activación no lineal entre capas, pudiéndose llegar al caso de diseñar una red de una

capa simple equivalente a una red multicapa si no se utiliza la función no lineal de activación entre capas.

2.4.- ENTRENAMIENTO DE LAS REDES NEURONALES ARTIFICIALES

Una de las principales características de las ANN es su capacidad de aprendizaje. El entrenamiento de las ANN muestra algunos paralelismos con el desarrollo intelectual de los seres humanos. No obstante aun cuando parece que se ha conseguido entender el proceso de aprendizaje conviene ser moderado porque el aprendizaje de las ANN está limitado.

El objetivo del entrenamiento de una ANN es conseguir que una aplicación determinada, para un conjunto de entradas produzca el conjunto de salidas deseadas o mínimamente consistentes. El proceso de entrenamiento consiste en la aplicación secuencial de diferentes conjuntos o vectores de entrada para que se ajusten los pesos de las interconexiones según un procedimiento predeterminado. Durante la sesión de entrenamiento los pesos convergen gradualmente hacia los valores que hacen que cada entrada produzca el vector de salida deseado.

Los algoritmos de entrenamiento o los procedimientos de ajuste de los valores de las conexiones de las ANN se pueden clasificar en dos grupos: Supervisado y No Supervisado.

Entrenamiento Supervisado: estos algoritmos requieren el emparejamiento de cada vector de entrada con su correspondiente vector de salida. El entrenamiento consiste en presentar un vector de entrada a la red, calcular la salida de la red, compararla con la salida deseada, y el error o diferencia resultante se utiliza para realimentar la red y cambiar los pesos de acuerdo con un algoritmo que tiende a minimizar el error.

Las parejas de vectores del conjunto de entrenamiento se aplican secuencialmente y de forma cíclica. Se calcula el error y el ajuste de los pesos por cada pareja hasta que el error para el conjunto de entrenamiento entero sea un valor pequeño y aceptable.

Entrenamiento No Supervisado: los sistemas neuronales con entrenamiento supervisado han tenido éxito en muchas aplicaciones y sin embargo tienen muchas críticas debido a que desde el punto de vista biológico no son muy lógicos. Resulta difícil creer que existe un mecanismo en el cerebro que compare las salidas deseadas con las salidas reales. En el caso de que exista, ¿de dónde provienen las salidas deseadas?

Los sistemas no supervisados son modelos de aprendizaje más lógicos en los sistemas biológicos. Desarrollados por Kohonen (1984) y otros investigadores, estos sistemas de aprendizaje no supervisado no requieren de un vector de salidas deseadas y por tanto no se realizan comparaciones entre las salidas reales y salidas esperadas. El conjunto de

vectores de entrenamiento consiste únicamente en vectores de entrada. El algoritmo de entrenamiento modifica los pesos de la red de forma que produzca vectores de salida consistentes. El proceso de entrenamiento extrae las propiedades estadísticas del conjunto de vectores de entrenamiento y agrupa en clases los vectores similares.

Existe una gran variedad de algoritmos de entrenamiento hoy en día. La gran mayoría de ellos han surgido de la evolución del modelo de aprendizaje no supervisado que propuso Hebb (1949). El modelo propuesto por Hebb se caracteriza por incrementar el valor del peso de la conexión si las dos neuronas unidas son activadas o disparadas. La ley de Hebb se representa según la ecuación (2.2).

$$w_{ij}(n + 1) = w_{ij}(n) + \alpha \text{OUT}_i \text{OUT}_j \quad \text{ecu.(2.2)}$$

SELECCIÓN DE LAS REDES NEURONALES ARTIFICIALES

3

- | | |
|----------------------------------|--------------------|
| 1.- Adaline y Madaline | 11.- DRS |
| 2.- ART | 12.- FLN |
| 3.- Back-Propagation | 13.- Hamming |
| 4.- BAM | 14.- Hopfield |
| 5.- The Boltzman Machine | 15.- LVQ |
| 6.- Brain-State-in a Box | 16.- Perceptron |
| 7.- Cascade-Correlation-Networks | 17.- PNN |
| 8.- Counter-Propagation | 18.- Recirculation |
| 9.- DBD | 19.- SOM |
| 10.- DNNA | 20.- SPR |

TEMA 3.- SELECCIÓN DE LAS REDES NEURONALES ARTIFICIALES

La clasificación de las redes neuronales artificiales que se presenta en este capítulo es una simple descripción de las diferentes ANN más comunes y frecuentes en la mayoría de los simuladores software de sistemas de computación neuronal. La selección de una red se realiza en función de las características del problema a resolver. La mayoría de éstos se pueden clasificar en aplicaciones de Predicción, Clasificación, Asociación, Conceptualización, Filtrado y Optimización. Los tres primeros tipos de aplicaciones requieren un entrenamiento supervisado.

Adaline y Madaline		AÑO	TIPO
diseñador:	Bernard Widrow	1960	<i>Predicción</i>
características:	Técnicas de Adaptación para el Reconocimiento de Patrones.		

Adaptive Resonance Theory Networks (ART)

diseñador: Carpenter, Grossberg 1960-86 *Conceptualización*
 características: Reconocimiento de Patrones y Modelo del Sistema Neuronal.
 Concepto de Resonancia Adaptativa.

Back-Propagation

diseñador: Rumelhart y Parker 1985 *Clasificación*
 características: Solución a las limitaciones de su red predecesora el Perceptron.

Bi-Directional Associative Memory (BAM) Networks

diseñador: Bart Kosko 1987 *Asociación*
 características: Inspirada en la red ART.

The Boltzmann Machine

diseñador: Ackley, Hinton y Sejnowski 1985 *Asociación*
 características: Similar a la red Hopfield.

Brain-State-in a Box

diseñador: James Anderson 1970-86 *Asociación*
 características: Red Asociativa Lineal.

Cascade-Correlation-Networks

diseñador: Fahhman y Lebiere 1990 *Asociación*
 características: Adición de nuevas capas ocultas en cascada.

Counter-Propagation

diseñador: Hecht-Nielsen 1987 *Clasificación*
 características: Clasificación Adaptativa de Patrones.

Delta-Bar-Delta (DBD) Networks

diseñador: Jacobb 1988 *Clasificación*
 características: Métodos Heurísticos para Acelerar la Convergencia.

Digital Neural Network Architecture (DNNA) Networks

diseñador: Neural Semiconductor Inc. 1990 *Predicción*
 características: Implementación Hardware de la función Sigmoid.

Directed Random Search (DRS) Networks

diseñador: Maytas y Solis 1965-81 *Clasificación*
 características: Técnica de valores Random en el mecanismo de Ajuste de Pesos.

Functional-link Networks (FLN)

diseñador:	Pao	1989	<i>Clasificación</i>
características:	Versión mejorada de la red Backpropagation.		

Hamming Networks

diseñador:	Lippman	1987	<i>Asociación</i>
características:	Clasificador de vectores binarios utilizando la Distancia Hamming.		

Hopfield Networks

diseñador:	Hopfield	1982	<i>Optimización</i>
características:	Concepto de la red en términos de energía.		

Learning Vector Quantization (LVQ) Networks

diseñador:	Kohonen	1988	<i>Clasificación</i>
características:	Red Clasificadora.		

Perceptron Networks

diseñador:	Rosenblatt	1950	<i>Predicción</i>
características:	Primer modelo de sistema Neuronal Artificial.		

Probabilistic Neural Network (PNN)

diseñador:	Spetcht	1988	<i>Asociación</i>
características:	Clasificación de Patrones utilizando métodos estadísticos.		

Recirculation Networks

diseñador:	Hinton y McClelland	1988	<i>Filtrado</i>
características:	Alternativa a la red Backpropagation.		

Self-Organizing Maps (SOM)

diseñador:	Kohonen	1979-82	<i>Conceptualización</i>
características:	Aprendizaje sin supervisión.		

Spatio-Temporal-Pattern Recognition (SPR)

diseñador:	Grossberg	1960-70	<i>Asociación</i>
características:	Red clasificadora Invariante en el espacio y tiempo.		

LAS PRIMERAS REDES NEURONALES ARTIFICIALES

4

- 4.1. Perceptron
- 4.2. Adaline - Madaline

TEMA 4.- LAS PRIMERAS REDES NEURONALES ARTIFICIALES

4.1.- PERCEPTRON

Arquitectura.-

La arquitectura del Perceptron, llamada mapeo de patrones (*pattern-mapping*), aprende a clasificar modelos mediante un aprendizaje supervisado. Los modelos que clasifica suelen ser generalmente vectores con valores binarios (0,1) y las categorías de la clasificación se expresan mediante vectores binarios.

El Perceptron presenta dos capas de unidades procesadoras (PE) y sólo una de ellas presenta la capacidad de adaptar o modificar los pesos de las conexiones. La arquitectura del Perceptron admite capas adicionales pero éstas no disponen la capacidad de modificar sus propias conexiones.

La Figura (4.1) muestra la unidad procesadora básica del Perceptron. Las entradas a_i llegan por la parte izquierda, y cada conexión con la neurona j tiene asignada un peso de valor w_{ji} .

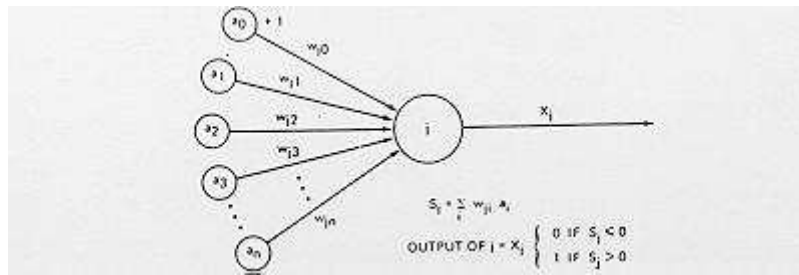


Figura (4.1) - Unidad Procesadora Básica del Perceptron.

La unidad procesadora del Perceptron realiza la suma ponderada de las entradas según la ecuación (4.1).

$$S_j = \sum a_i w_{ji} \quad \text{ec.(4.1)}$$

Un aspecto común en muchas de las ANN es la entrada especial llamada "bias" representada en la parte superior izquierda de la figura (entrada a_0). Esta entrada siempre presenta un valor fijo, +1 y funciona como una masa en un circuito eléctrico donde no varía de valor (se puede utilizar como un valor constante de referencia).

El Perceptron comprueba si la suma de las entradas ponderadas es mayor o menor que un cierto valor umbral y genera la salida " x_j " según la ecuación (4.2).

$$\begin{aligned} \text{si } S_j > 0 \text{ entonces } x_j &= 1 \\ \text{si } S_j \leq 0 \text{ entonces } x_j &= 0 \end{aligned} \quad \text{ec.(4.2)}$$

La salida x_j es transmitida a lo largo de la línea de salida y constituye uno de los componentes del vector de salida de la red.

Las redes Perceptron de dos capas, representadas en la Figura (4.2) tienen una capa de entrada y una capa de unidades procesadoras que constituyen la capa de salida.

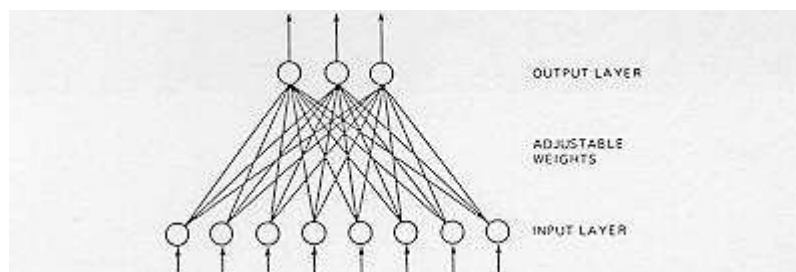


Figura (4.2) - Red Perceptron de dos Capas.

A lo largo de los años 50 y 60 se desarrollaron muchos tipos de topologías de redes basadas en la arquitectura del Perceptron. Las topologías con tres o más capas se caracterizan porque la regla de aprendizaje del perceptron sólo adapta los pesos o

valores de las conexiones de una capa. Una aplicación típica de un sistema de tres capas es la que muestra la Figura (4.3) donde la entrada es la imagen de la letra E y la salida es la categorización de la entrada en dos clases.

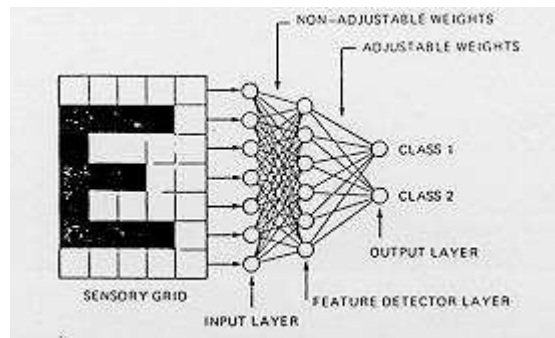


Figura (4.3) - Red Perceptron de tres Capas.

El entrenamiento del Perceptron consiste en presentar a la red todos los elementos del conjunto de entrenamiento constituido por parejas de vectores (entrada y salida deseada) de forma secuencial.

El objetivo del entrenamiento es llegar a un conjunto de valores de los pesos de la red de forma que responda correctamente a todo el conjunto de entrenamiento. Después del entrenamiento los pesos no son ya modificados y la red está ya en disposición de responder adecuadamente a las entradas que se le presenten.

La adaptación de los pesos se puede realizar mediante diferentes reglas. Una de las reglas más simples de aprendizaje del Perceptron se indica en la ecuación (4.3):

$$W_{jinuevo} = W_{jiviejo} + C (t_j * x_j) a_i \quad \text{ec.(4.3)}$$

Siendo t_j el valor de la salida deseada, x_j el valor de salida producida por la unidad procesadora, a_i el valor de la entrada i y C el coeficiente de aprendizaje.

En todo proceso de entrenamiento el comportamiento de la red inicialmente va mejorando hasta que llega a un punto en el que se estabiliza y se dice que la red ha convergido. Esta convergencia tiene dos posibilidades, la primera consiste en que la red haya aprendido correctamente el conjunto de entrenamiento o la segunda se trata de que la red no ha aprendido todas las respuestas correctas.

Separación Lineal.-

El mayor inconveniente del Perceptron, a pesar del éxito que ha tenido en muchas aplicaciones de clasificación de patrones es la imposibilidad de adaptar los pesos de todas las capas. En los años en los que se realizó el Perceptron, los investigadores no fueron capaces de diseñar un algoritmo que propagara las correcciones de los pesos a través de redes multicapa.

La principal limitación funcional del Perceptron es que una unidad de salida sólo puede clasificar patrones linealmente separables. La Figura (4.4) ilustra el concepto general de Separabilidad Lineal, es decir, las clases de patrones que pueden separarse en dos clases mediante una línea. Este concepto se puede extender a tres o más dimensiones simplemente separando dos clases mediante planos e hiperplanos.

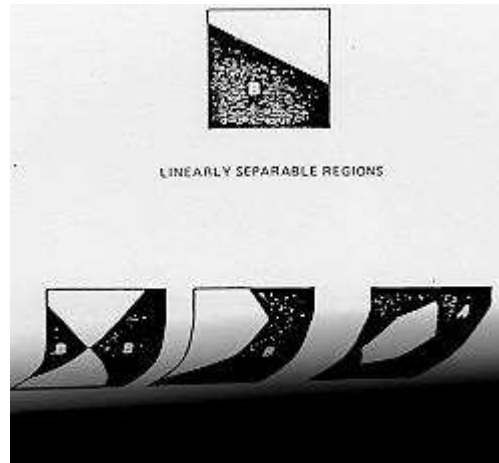


Figura (4.4) - Separabilidad Lineal.

En este punto Minsky y Papert centraron las críticas al Perceptron en su publicación *Perceptrons* (1969). El libro incluía opiniones negativas sobre la posibilidad de extender el Perceptron en una herramienta útil en la computación neuronal; por ejemplo para pequeños problemas de clasificación de patrones como el OR exclusivo, el Perceptron es incapaz de resolverlo con éxito.

Afortunadamente para la computación neuronal surgieron nuevas reglas de aprendizaje para redes multicapa y nuevas arquitecturas, entre ellas la más popular Backpropagation, que resolvieron entre otros los problemas de clasificación de patrones no separables linealmente.

4.2.- ADALINE - MADALINE

La arquitectura de Adaline (*Adaptive Linear Neuron*) fue creada por Bernard Widrow en 1959. Utiliza un dispositivo lógico que realiza una suma lineal de las entradas y genera una función umbral para el resultado de dicha suma.

La arquitectura Madaline (*Multilayer Adaline*) creada también por Widrow presenta una configuración constituida por dos o más unidades Adaline.

A lo largo del tiempo se han estudiado diferentes variaciones de los algoritmos de aprendizaje de la Adaline, y Madaline, y entre las aplicaciones investigadas destacan entre otras, filtros adaptativos de eliminación de ruido y reconocimiento de patrones de señales .

No obstante, desde los primeros experimentos con la Adaline y Madaline se constató la capacidad de clasificar patrones linealmente separables, presentando la misma

limitación que el Perceptron: la carencia de un método que ajuste más de una capa de pesos.

Estructura Adaline.-

La Figura (4.5.a) muestra una Adaline básica. La unidad procesadora representada por un círculo con el símbolo sumatorio implementa una función umbral. Las conexiones de cada una de las entradas tienen asociadas un valor de ponderación llamado también peso w_i .

El mecanismo de ajuste de los pesos representado en la Figura (4.5.b), consiste en utilizar

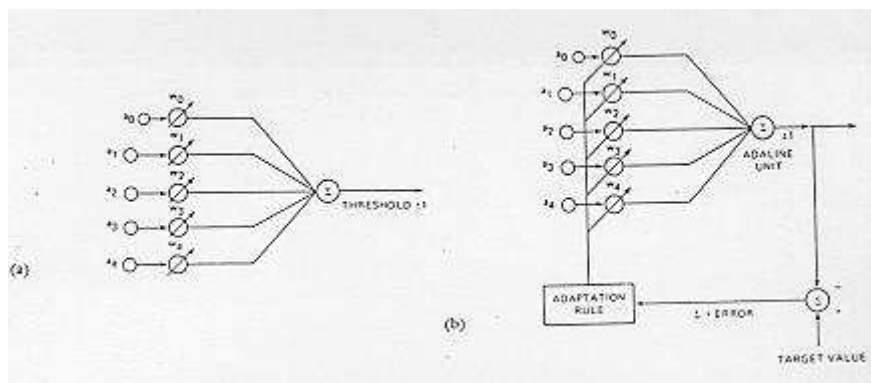


Figura (4.5) - Red Adaline.

la diferencia entre el valor de la salida y el valor esperado. La unidad procesadora actúa como un sumador y después realiza la función umbral según la ecuación (4.4)

$$x_j = \begin{cases} 1 & \text{si } S = \sum_i a_i w_i \geq 0 \\ -1 & \text{si } S = \sum_i a_i w_i < 0 \end{cases} \quad \text{ec.(4.4)}$$

La salida de la unidad Adaline es ± 1 a diferencia de la arquitectura del Perceptron que sólo permite los valores 0 y 1.

El entrenamiento se realiza presentando repetidamente una serie de parejas de entradas y salidas. El objetivo de la Adaline durante el proceso de la adaptación es producir la salida deseada como propia suya.

La regla de aprendizaje en la arquitectura de la Adaline es la regla de Widrow-Hoff expresada en la ecuación (4.5)

$$\Delta w_i = \eta a_i (t * x) \quad \text{ec.(4.5)}$$

siendo η la constante de aprendizaje, a_i la salida de la unidad i , t la salida deseada y por último x la salida de la unidad Adaline. No obstante la variante de esta regla más utilizada considera el valor de la suma ponderada S en vez del valor de la salida de la unidad Adaline.

Estructura Madaline.-

El sistema Madaline tiene una capa de unidades Adaline que están conectadas a una simple unidad Madaline. La Figura (4.6) muestra cuatro unidades en la capa de entrada, tres unidades Adaline en la segunda capa y una unidad Madaline en la tercera capa.

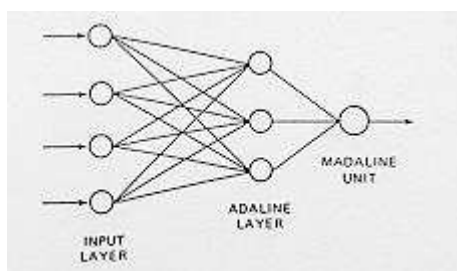


Figura (4.6) - Sistema Madaline.

Las conexiones entre la capa de entrada y la capa de las unidades Adaline tienen asociadas un peso ajustable por cada una de ellas. Sin embargo, las conexiones entre la capa de Adaline y la unidad Madaline no tienen asociado ningún peso. Cada unidad Adaline transmite su salida (-1 ó +1) a la unidad Madaline. La Madaline emplea una regla de mayorías para obtener su salida: si la mitad o más de las unidades Adaline presentan un valor de salida +1, entonces la salida de la Madaline es +1. En caso contrario el valor de salida de la red Madaline es -1.

El entrenamiento de los sistemas Madaline es similar al entrenamiento de las Adaline. El conjunto de entrenamiento es un conjunto de patrones de entrada emparejados con las salidas deseadas. Una vez que se presenta el patrón a la entrada, el sistema Madaline calcula su salida y a continuación se compara con la salida deseada. Los pesos son modificados después de que cada patrón sea presentado a la entrada del sistema.

RED BACKPROPAGATION

5

- 5.1. Introducción
- 5.2. Arquitectura de la Red Backpropagation
- 5.3. Algoritmo de Entrenamiento
- 5.4. Aplicaciones de la Red Backpropagation
- 5.5. Ventajas e Inconvenientes

TEMA 5.- RED BACKPROPAGATION

5.1.- INTRODUCCIÓN

Durante muchos años no se obtuvo ningún tipo de éxito en el diseño de algoritmos de entrenamiento de redes multicapa. A partir de la comprobación de la severa limitación de los sistemas de una capa, el mundo de la computación neuronal entró en un obscurecimiento y abandono casi general durante dos décadas.

La invención del algoritmo Backpropagation ha desempeñado un papel vital en el resurgimiento del interés de las redes neuronales artificiales. Backpropagation es un método de entrenamiento de redes multicapa. Su potencia reside en su capacidad de entrenar capas ocultas y de este modo supera las posibilidades restringidas de las redes de una única capa.

El concepto básico de Backpropagation fue presentado en 1974 por Paul Werbos e independientemente reinventado por David Parker en 1982, y también presentado en 1986 por Rumelhart, Hinton y Williams. La duplicidad de esfuerzos y trabajos es frecuente en cualquier disciplina, y más en el mundo de las ANN debido a su naturaleza interdisciplinaria.

5.2.- ARQUITECTURA DE LA RED BACKPROPAGATION

La unidad procesadora básica de la red Backpropagation se representa en la Figura (5.1). Las entradas se muestran a la izquierda, y a la derecha se encuentran unidades que reciben la salida de la unidad procesadora situada en el centro de la figura.

La unidad procesadora se caracteriza por realizar una suma ponderada de las entradas llamada S_j , presentar una salida a_j y tener un valor δ_j asociado que se utilizará en el proceso de ajuste de los pesos. El peso asociado a la conexión desde la unidad i a la unidad j se representa por w_{ji} , y es modificado durante el proceso de aprendizaje.

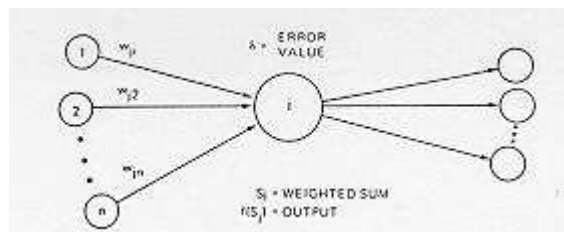


Figura (5.1) - Unidad Procesadora Básica Backpropagation .

Normalmente, la Backpropagation utiliza tres o más capas de unidades procesadoras. La Figura (5.2) muestra la topología backpropagation típica de tres capas. La capa inferior es la capa de entrada, y se caracteriza por ser la única capa cuyas unidades procesadoras reciben entradas desde el exterior. Sirven como puntos distribuidores, no realizan ninguna operación de cálculo. Las unidades procesadoras de las demás capas procesan las señales como se indica en la figura (5.1). La siguiente capa superior es la capa oculta, y todas sus unidades procesadoras están interconectadas con la capa inferior y con la capa superior. La capa superior es la capa de salida que presenta la respuesta de la red.

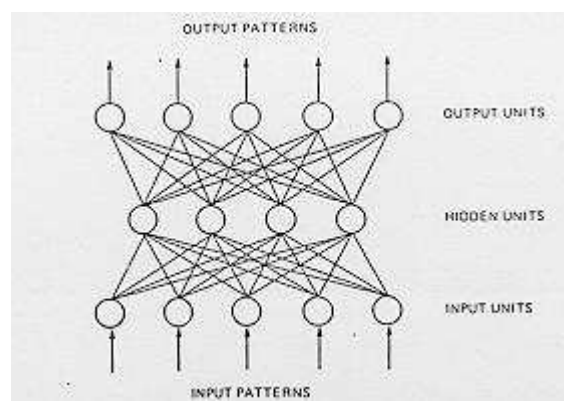


Figura (5.2) - Red Backpropagation completamente interconectada.

5.3.- ALGORITMO DE ENTRENAMIENTO

Las redes Backpropagation tienen un método de entrenamiento supervisado. A la red se le presenta parejas de patrones, un patrón de entrada emparejado con un patrón de salida deseada. Por cada presentación los pesos son ajustados de forma que disminuya el error entre la salida deseada y la respuesta de la red.

El algoritmo de aprendizaje backpropagation conlleva una fase de propagación hacia adelante y otra fase de propagación hacia atrás. Ambas fases se realizan por cada patrón presentado en la sesión de entrenamiento.

Propagación hacia Adelante.-

Esta fase de propagación hacia adelante se inicia cuando se presenta un patrón en la capa de entrada de la red. Cada unidad de la entrada se corresponde con un elemento del vector patrón de entrada. Las unidades de entrada toman el valor de su correspondiente elemento del patrón de entrada y se calcula el valor de activación o nivel de salida de la primera capa. A continuación las demás capas realizarán la fase de propagación hacia adelante que determina el nivel de activación de las otras capas.

La unidad procesadora j obtiene la cantidad S_j según la ecuación (5.1)

$$S_j = \sum_i a_i w_{ji} \quad \text{ec.}(5.1)$$

y genera la salida o nivel de activación según la ecuación (5.2)

$$\text{Salida} = f(S_j) \quad \text{ec.}(5.2)$$

La función f es una función umbral genérica, entre las que cabe destacar la función Sigmoid y la función Hiperbólica.

El valor de la salida de la unidad j es enviado o transmitido a lo largo de todas las conexiones de salida de dicha unidad.

La Figura (5.3) muestra la fase de propagación hacia adelante.

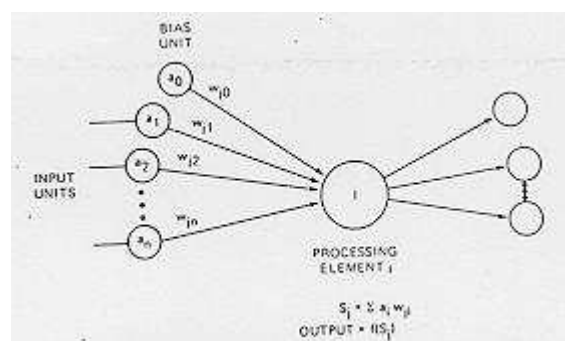


Figura (5.3) - Fase de Propagación hacia Adelante.

Conviene indicar que las unidades procesadoras de la capa de entrada no realizan ninguna operación de cálculo con sus entradas, ni operaciones con funciones umbrales, sólo asumen su salida como el valor del correspondiente elemento del vector de entrada.

Por otro lado, algunas redes backpropagation utilizan unidades llamadas bias como parte de cualquiera de las capas ocultas y de la capa de salida. Estas unidades presentan constantemente un nivel de activación de valor 1. Además esta unidad está conectada a todas las unidades de la capa inmediatamente superior y los pesos asociados a dichas conexiones son ajustables en el proceso de entrenamiento. La utilización de esta unidad tiene un doble objetivo, mejorar las propiedades de convergencia de la red y ofrecer un nuevo efecto umbral sobre la unidad que opera.

Propagación hacia Atrás.-

Una vez se ha completado la fase de propagación hacia adelante se inicia la fase de corrección o fase de propagación hacia atrás. Los cálculos de las modificaciones de todos los pesos de las conexiones empiezan por la capa de salida y continua hacia atrás a través de todas las capas de la red hasta la capa de entrada. Dentro de los tipos de ajuste de pesos se puede clasificar dos grupos, ajuste de unidades procesadoras de la capa de salida y ajuste de unidades procesadoras de las capas ocultas.

Ajuste de Pesos de la Capa de Salida: el ajuste de estos pesos es relativamente sencillo debido a que existe y se conoce el valor deseado para cada una de las unidades de la capa de salida. Cada unidad de la capa de salida produce un número real como salida y se compara con el valor deseado especificado en el patrón del conjunto de entrenamiento.

A partir del resultado de la comparación se calcula un valor de error δ_j , según la ecuación (5.3) para cada unidad de la capa de salida.

$$\delta_j = (t_j - a_j) f'(S_j) \quad \text{ec.(5.3)}$$

siendo t_j el valor de salida deseado y f' la derivada de la función umbral f .

La Figura (5.4) muestra el cálculo de los valores δ_j de las unidades de la capa de salida.

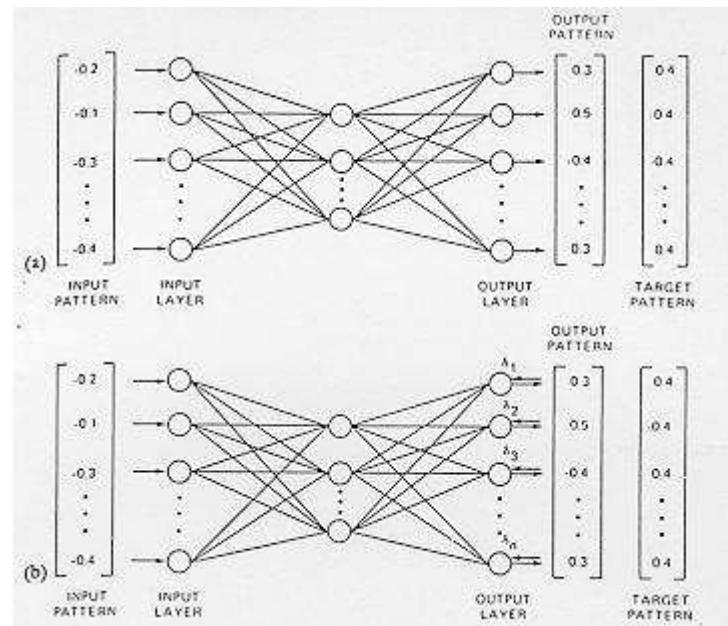


Figura (5.4) - Cálculo de los valores δ_j de la Capa de Salida.

Ajuste de Pesos de las Capas Ocultas: estas capas no tienen un vector de salidas deseadas y por tanto no se puede seguir el método de propagación de error mencionado en el caso de unidades procesadoras de la capa de salida. El valor de error calculado para este tipo de unidades procesadoras se obtiene a partir de la ecuación (5.4).

$$\delta_j = [\sum_k \delta_k w_{kj}] f' \quad (5)$$

ec.(5.4)

La Figura (5.5) representa la obtención del valor δ_j para las unidades de las capas ocultas.

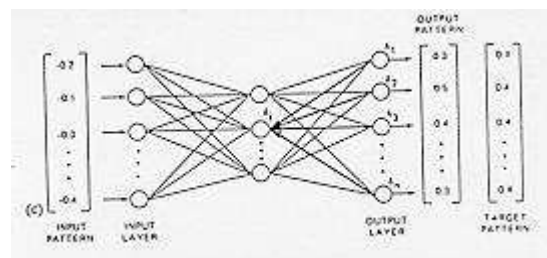


Figura (5.5) - Cálculo de los valores δ_j de las Capas Ocultas.

El ajuste de los pesos asociados a las conexiones se realiza a partir del valor δ_j de cada unidad de proceso. Cada peso es ajustado según la ecuación (5.5) conocida como la regla δ_j generalizada (Rumelhart y McClelland 1986)

$$\Delta w_{ji} = \eta \delta_j a_i$$

ec.(5.5)

La variable η es el coeficiente de aprendizaje. Este coeficiente, normalmente entre 0.25 y 0.75 refleja el grado de aprendizaje de la red. Algunas veces este coeficiente es

modificado de un valor inicial alto a valores progresivamente menores durante la sesión del entrenamiento con el objetivo de lograr un mejor aprendizaje.

Convergencia: en el proceso de entrenamiento o aprendizaje de la Backpropagation es frecuente medir cuantitativamente el aprendizaje mediante el valor RMS (*Root Mean Square*) del error de la red. Esta medida refleja el modo en el que la red está logrando respuestas correctas; a medida que la red aprende, su valor RMS decrece.

Debido a que los valores de salida de la red y los valores de salidas deseadas son valores reales, es necesario definir un parámetro de corte o un valor umbral del valor RMS del error de la red que permita decir que la red se aproxima a la salida deseada y considerar que la respuesta es correcta.

La convergencia es un proceso en el que el valor RMS del error de la red tiende cada vez más al valor 0. La convergencia no siempre es fácil de conseguirla porque a veces el proceso puede requerir un tiempo excesivo o bien porque la red alcanza un mínimo local y deja de aprender.

5.4.- APLICACIONES DE LA RED BACKPROPAGATION

Las redes Backpropagation han demostrado su capacidad de trabajar con éxito en un amplio rango de aplicaciones incluyendo clasificación de imágenes, síntesis de voz, clasificación de ecos de sonar, sistemas de base de conocimiento, codificación de información y muchos otros problemas de clasificación y problemas de percepción.

Algunos ejemplos y estudios de aplicaciones de la Backpropagation son los siguientes:

Sejnowski y Rosenberg (1987) lograron un gran éxito con el sistema llamado NetTalk, un sistema que convierte texto escrito en Inglés a voz de alta inteligibilidad. La voz obtenida en la sesión de entrenamiento recuerda los sonidos de un niño en sus diferentes estados del aprendizaje del hablar.

En Japón NEC ha anunciado la utilización de una red backpropagation en un sistema de reconocimiento óptico de caracteres, obteniendo una exactitud superior al 99%. Esta mejora ha sido conseguida mediante la combinación de algoritmos convencionales y una backpropagation que provee una verificación adicional.

Otra aplicación de la red backpropagation es el reconocimiento de formas de dos dimensiones. Este tipo de sistemas es muy útil en aplicaciones de identificación de números escritos a mano, lectura de caracteres escritos a mano, ordenamiento de partes en una producción industrial, inspección automática de defectos y procesamiento de imágenes médicas (Dayhoff 1988).

Cotrell, Munro y Zipper (1987) han realizado una aplicación de compresión de imagen en la que las imágenes se representan con un bit por pixel, obteniendo una reducción de 8:1 sobre los datos de entrada.

La aplicación de Waibel (1988) consiste en un estudio de clasificación de patrones que son presentados fuera de un período de tiempo. La red de Waibel es una red neuronal constituida con elementos de retardo en el tiempo que ha sido entrenada para reconocer sílabas habladas.

Un ejemplo clásico de la utilización de red Backpropagation es la función OR exclusivo. La red Perceptron no es capaz de resolver este problema porque el problema no es linealmente separable y su solución requiere dos capas de pesos ajustables. Sin embargo, la red Backpropagation dispone de un método de entrenamiento que ajusta los pesos de todas las capas y resuelve este problema linealmente no separable. No obstante uno de los problemas comunes en resolver la función X-or con la red Backpropagation es la presencia de mínimos locales y por consiguiente la falta de convergencia de las respuestas correctas para todos los patrones del conjunto de entrenamiento.

5.5.- VENTAJAS E INCONVENIENTES

La principal ventaja de la Backpropagation es su capacidad genérica de mapeo de patrones. La red es capaz de aprender una gran variedad de relaciones de mapeo de patrones. No requiere un conocimiento matemático de la función que relaciona los patrones de la entrada y los patrones de salida. La Backpropagation sólo necesita ejemplos de mapeo para aprender. La flexibilidad de esta red es aumentada con la posibilidad de elegir número de capas, interconexiones, unidades procesadoras, constante de aprendizaje y representación de datos. Como resultado de estas características la red Backpropagation es capaz de participar con éxito en una amplia gama de aplicaciones.

El mayor inconveniente es el tiempo de convergencia. Las aplicaciones reales pueden llegar a tener miles de ejemplos en el conjunto de entrenamiento y ello requiere días de tiempo de cálculo. Además la backpropagation es susceptible de fallar en el entrenamiento, es decir, la red puede que nunca llegue a converger.

Existe una variedad de técnicas desarrolladas para disminuir el tiempo de convergencia y evitar los mínimos locales. El término de "momentum" se utiliza para aumentar la velocidad del proceso de convergencia. Otra forma de mejorar la convergencia se basa en la variación del parámetro de aprendizaje η comenzando con valores altos y adquiriendo progresivamente valores más pequeños. Entre las técnicas utilizadas para evitar los mínimos locales destacan cambiar la red, cambiar el conjunto de entrenamiento y añadir ruido aleatorio a los pesos.

RED SELF ORGANIZING MAP Y RED COUNTERPROPAGATION

6

- 6.1. Introducción Red Self Organizing Map
- 6.2. Arquitectura Básica y Modo de Operación
- 6.3. Ejemplos red S.O.M.
- 6.4. Introducción Red Counterpropagation
- 6.5. Arquitectura y Funcionamiento
- 6.6. Ejemplos red Counterpropagation

TEMA 6.- RED SELF ORGANIZING MAP Y RED COUNTERPROPAGATION

6.1.- INTRODUCCIÓN RED SELF ORGANIZING MAP

La red S.O.M. (*Self Organizing Map*) tiene la característica de organizar mapas topológicos. El mapa que presenta la red a partir de una situación inicial aleatoria muestra las relaciones existentes entre los diferentes patrones presentados a la red. Este modelo de red fue presentado por Kohonen (1988) aun cuando otros investigadores como Grossberg también se encontraban trabajando en la misma red.

Esta nueva red muestra de forma efectiva la idea básica de tener una red neuronal artificial que organice un mapa topológico, constituyendo como tal una primera aproximación a los mapas topológicos de los fenómenos motores y sensoriales existentes en la superficie del cerebro humano. La red de Kohonen presenta ventajas sobre las técnicas clásicas de reconocimiento de patrones porque además de utilizar la arquitectura paralela de las redes neuronales provee una representación gráfica de las relaciones entre los patrones.

Un aspecto diferenciador de la red SOM de otras muchas redes es que aprende sin supervisión, de aquí su nombre en inglés. No obstante, cuando la red SOM está en combinación con otras capas neuronales para aplicaciones de categorización y/o

predicción la red aprende primeramente en modo no supervisado y después cambia a modo supervisado.

Las aplicaciones más frecuentes de esta red son visualizar topologías y estructuras jerárquicas de espacios de entrada de dimensión elevada, así como su utilización en redes híbridas para problemas de predicción y clasificación.

6.2.- ARQUITECTURA BÁSICA Y MODO DE OPERACIÓN

La red SOM presenta una topología constituida por dos capas. La primera capa de la red es la capa de entrada, y la segunda capa, llamada capa competitiva o de Kohonen está organizada en una rejilla de dos dimensiones.

Las dos capas están totalmente interconectadas como se muestra en la Figura (6.1). Cada una de las conexiones tiene asociado un peso que será modificado a lo largo de la sesión de entrenamiento.

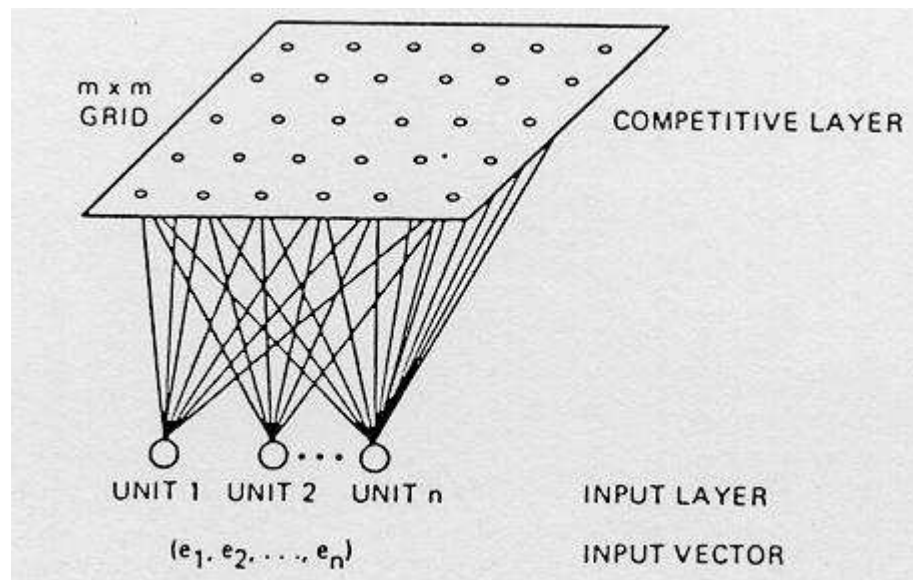


Figura (6.1) - Estructura básica de la Red SOM.

Las reglas básicas de operación de la red SOM son las siguientes:

- a) Localizar la neurona en la capa de Kohonen cuyos pesos asociados se aproximen mejor al patrón de entrada.
- b) Aumentar la aproximación de los pesos asociados de dicha unidad y sus vecinas al vector de entrada.
- c) Disminuir gradualmente los dos parámetros que intervienen en dicho proceso, el parámetro de aprendizaje y el tamaño del vecindario.

La localización de la neurona cuyos pesos se aproximan a la entrada responde a que el aprendizaje de la capa de Kohonen sigue el modelo de aprendizaje competitivo, de ahí el nombre de dicha capa.

El ajuste de los pesos de la neurona ganadora se realiza para que se aproxime más a los datos de la entrada; y por otra parte el ajuste de los pesos de las neuronas vecinas contribuye a mantener el orden del propio espacio de entrada.

6.3.- EJEMPLOS RED S.O.M.

Mapeo de Diferentes Dimensiones.-

Una de las aplicaciones más interesantes de la red SOM es el mapeo de patrones de una cierta dimensión a otra determinada dimensión. Este tipo de transformación resulta especialmente interesante para la reducción de dimensiones de los datos de entrada.

En las redes SOM la dimensión de los patrones de entradas es el número de componentes del vector de entrada; y la dimensión de la salida es el número de dimensiones que tiene la rejilla de las neuronas competitivas (una línea, un plano, un array de tres dimensiones,...).

La Figura (6.2.a) muestra el mapa de Kohonen de una red de una sola dimensión para entradas de dos dimensiones uniformemente distribuidas en un triángulo. La Figura (6.2.b) representa el caso de una transformación de tres dimensiones a dos dimensiones.

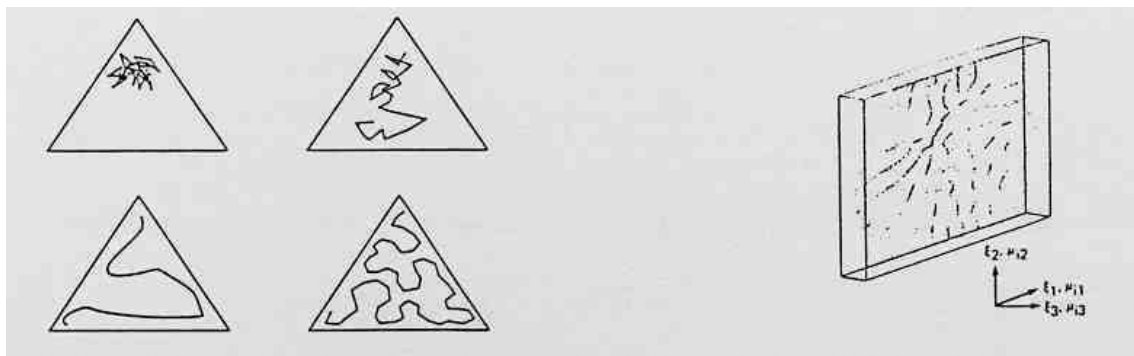


Figura (6.2) - Evolución de los Pesos de la capa de Kohonen.

Redes Híbridas.-

Las redes constituidas por la combinación de una red SOM cuya salida es la entrada a otro tipo de capas o redes muestran resultados satisfactorios en aplicaciones de categorización y/o predicción.

La fase de entrenamiento o aprendizaje de la red híbrida tiene dos partes. La primera parte es la correspondiente a la estabilización de la capa de Kohonen y los coeficientes

de aprendizaje de los pesos de las conexiones hacia la salida permanecen a cero. En la segunda fase del entrenamiento se ajustan los pesos de la capa de salida manteniendo los coeficientes de la capa de Kohonen a cero. La Figura (6.3) muestra la arquitectura de una red híbrida para la categorización de entradas de cuatro dimensiones en tres posibles categorías. La red está constituida por una capa de entrada, una capa de Kohonen (4x4) y una capa de salida. Esta última utiliza la regla de aprendizaje de Widrow-Hoff.

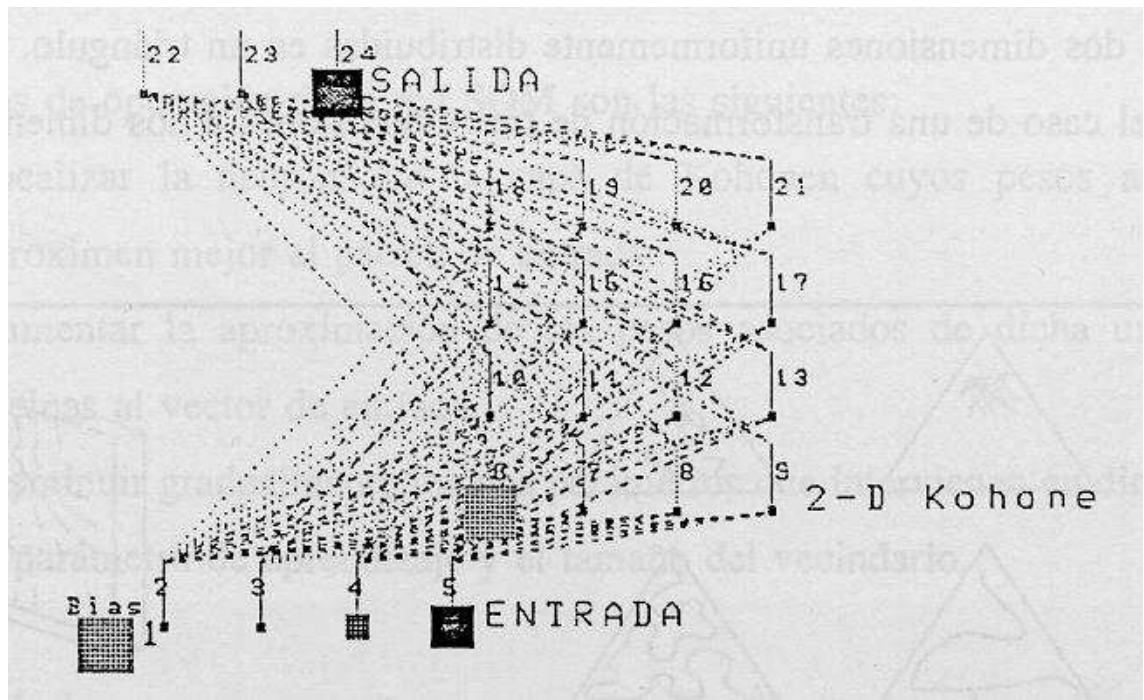


Figura (6.3) - Sistema Híbrido SOM con Categorización.

6.4.- INTRODUCCIÓN RED COUNTERPROPAGATION

La red Counterpropagation desarrollada por Robert Hecht-Nielsen (1987) constituye un buen ejemplo de combinación de diferentes capas de distintas redes o arquitecturas para la construcción de un nuevo tipo de red. Se utilizan dos tipos diferentes de capas: la capa oculta es una capa de Kohonen con neuronas competitivas y aprendizaje no supervisado. Y la capa de salida que está totalmente conectada a la capa oculta no es competitiva. El entrenamiento de esta capa sigue la regla de Widrow-Hoff o la regla de Grossberg.

Entre las aplicaciones más adecuadas para este tipo de red se encuentran clasificación de patrones, aproximación a funciones, análisis estadístico y compresión de datos. Se puede resumir que básicamente el objetivo de esta red es el mapeo de un patrón en otro patrón.

6.5.- ARQUITECTURA Y FUNCIONAMIENTO

La topología o arquitectura de la red Counterpropagation típica de tres capas se muestra en la Figura (6.4).

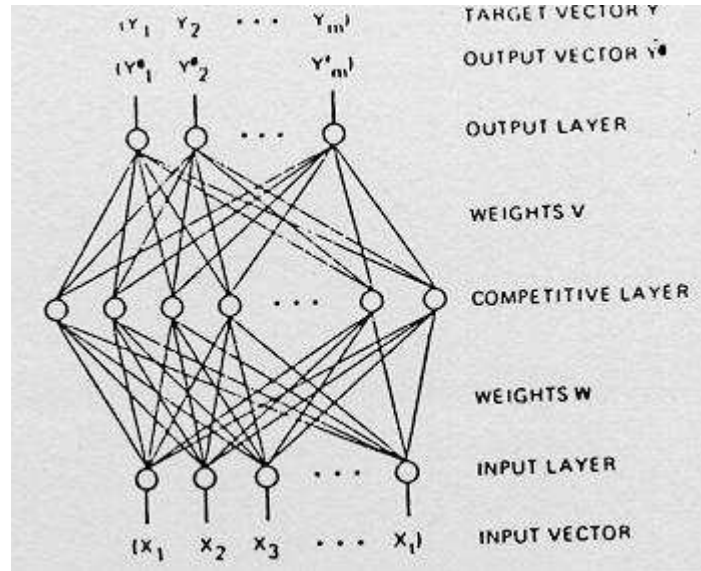


Figura (6.4) - Red Counterpropagation de tres Capas.

La primera capa constituye la capa de entrada, la segunda capa es la capa competitiva o de Kohonen y la tercera capa es la capa de salida llamada también de Grossberg. En la figura se muestra el objetivo perseguido por esta red, el emparejamiento del vector de entrada X (n componentes) y el vector deseado Y (m componentes).

Una vez entrenada la red Counterpropagation funciona de la siguiente forma: ante un patrón presentado en la entrada, las unidades o neuronas de la capa oculta o de Kohonen compiten por responder a dicha entrada. Una única neurona será la ganadora, presentando el nivel de activación, mientras las demás permanecerán inactivas. La neurona ganadora representa la categoría a la que pertenece la entrada. Esta neurona activa un patrón en la capa de salida convirtiéndose en la salida de la red. Es manifiesto la importancia de los pesos asociados de dicha neurona a la capa de salida, ya que tienen una influencia total en el valor final de las neuronas de la salida.

Entrenamiento.-

Durante entrenamiento se ajustan los pesos de las conexiones de las dos capas, primero los correspondientes a la capa de Kohonen (una vez elegida la neurona ganadora) y después los pesos de la capa de salida.

Se elige la neurona ganadora en respuesta a la presentación de un patrón en la entrada. Sólo se ajustan los pesos de las conexiones entre la entrada y la neurona ganadora, permaneciendo los demás pesos inalterados. Después de seleccionar la neurona ganadora se calcula la salida de la red, se compara con la salida o patrón deseado y se

ajustan los pesos de la segunda capa. Los pesos de las conexiones entrantes a la neurona ganadora se modifican según la ecuación (6.1).

$$\Delta w_i = \alpha (x_i * w_i) \quad \text{ec. (6.1)}$$

siendo α la constante de aprendizaje, x el vector de entrada y w los pesos entrantes a la neurona ganadora.

Una vez ajustados los pesos según la ecuación anterior es necesario renormalizar el vector w . Un aspecto notable de este mecanismo de ajuste de pesos y normalización de las entradas y pesos es el efecto de mover el vector w hacia el vector de entrada x como se muestra gráficamente en la Figura (6.5).

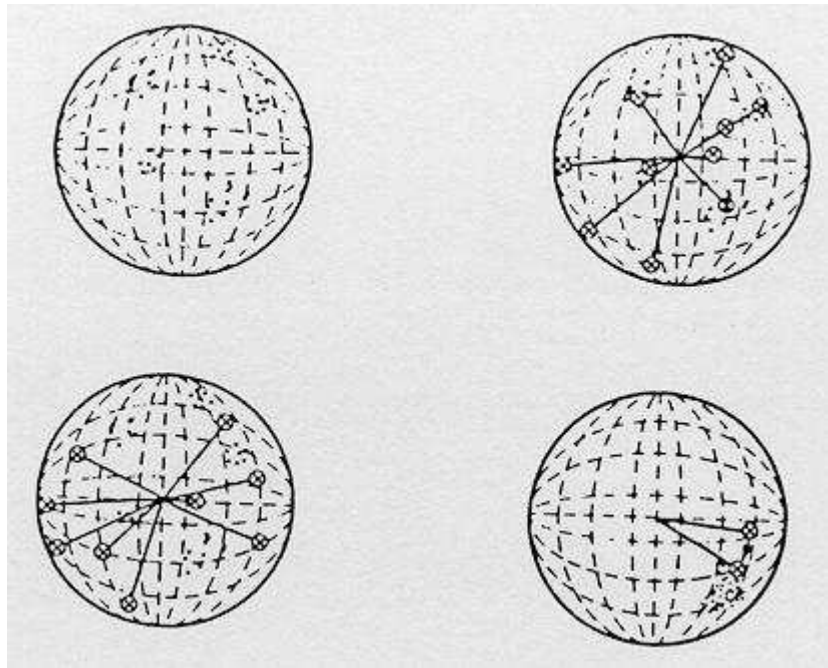


Figura (6.5) - Movimiento de los vectores Pesos hacia los vectores Entrada.

El próximo vector de entrada para la cual la neurona ganadora sea la misma, moverá otra vez el vector w y esta vez hacia la nueva entrada. De esta forma a lo largo del entrenamiento el vector de los pesos w generalmente se aproxima a la media de todas las entradas para las que su neurona resulta ganadora. A estos vectores que representan a un grupo de vectores de entrada afines se les llama vectores modelo (*exemplars*). Así para un vector de entrada dado éste activará el vector modelo que más próximo esté al vector de entrada, es decir al vector w correspondiente a la neurona ganadora.

Los pesos entrantes a la capa de salida se ajustan según la regla de Widrow-Hoff expresada en la ecuación (6.2)

$$\Delta v_i = \beta z_i (y_i * y'_i) \quad \text{ec. (6.2)}$$

siendo β el coeficiente de aprendizaje, y e y' la salida de la red y la salida deseada respectivamente, z la activación de las neuronas de la capa de Kohonen y v los pesos de las conexiones. Conviene recordar que en todo instante sólo una neurona es ganadora,

por tanto sólo ella presenta nivel de activación con lo cual el único peso ajustado de cada neurona de salida es el peso conectado a la neurona ganadora.

6.6.- EJEMPLOS RED COUNTERPROPAGATION

Clasificación de Patrones.-

Los patrones a clasificar en esta aplicación son vectores de dos dimensiones representados en un plano como se indica en la Figura (6.6.b). Las clases de los patrones de entrada consisten en cuatro grupos de vectores claramente separados, y dentro de cada clase los vectores se encuentran dentro de un cuadrado de tamaño pequeño.

En este ejemplo inicialmente el número de neuronas de la capa de entrada es dos, y el número de neuronas de la capa de salida es cuatro correspondientes a las cuatro clases de patrones. La capa oculta o de Kohonen puede tener un número de neuronas aleatorio. La Figura (6.6.a) muestra la arquitectura de la red Counterpropagation para esta aplicación en la que se observa que los vectores de entrada previamente son normalizados añadiendo para ello un tercer componente al vector de entrada.

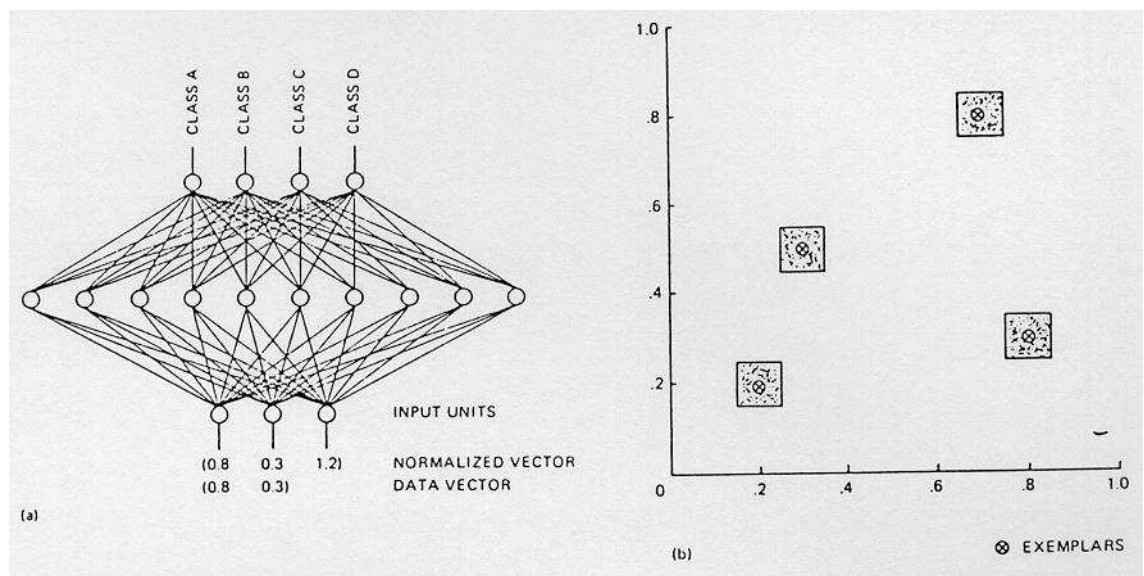


Figura (6.6) -Red Counterpropagation Clasificadora de patrones.

Asesor del tiempo libre.-

Este ejemplo es una muestra de la utilización de redes neuronales artificiales en aplicaciones resueltas tradicionalmente por sistemas expertos. Se desea tener un consejero que en función de las dos variables de entrada, trabajo de la oficina y sentimiento hacia su pareja, asesore indicando qué clase de actividad conviene desarrollar el domingo a la tarde. La Figura (6.7.a) muestra la arquitectura de la red counterpropagation elegida para esta aplicación utilizando el paquete software Neural Works Professional II. La normalización de la entrada se lleva a cabo con la inclusión

de una nueva capa de neuronas que tenga tantas neuronas como tenga la capa de entrada más una (la neurona adicional no está conectada a la capa de entrada). La Figura (6.7.b) muestra los vectores de entradas y salidas del conjunto de entrenamiento, y los consejos ofrecidos por la red para un conjunto de nuevas entradas. La red tiene la capacidad de generalizar a partir de los casos mostrados y proveer una respuesta a una entrada que nunca le ha sido formulada. Este tipo de generalización también se encuentra en otros tipos de redes como la Backpropagation.

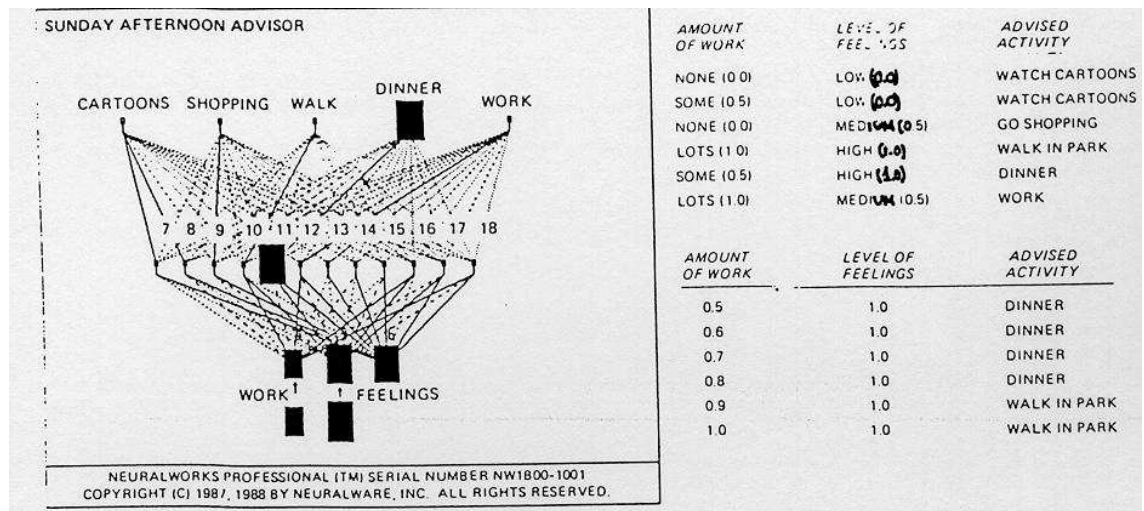


Figura (6.7) -Red Counterpropagation Asesora

RED HOPFIELD Y RED BIDIRECTIONAL ASSOCIATIVE MEMORY

7

- 7.1. Red Hopfield
- 7.2. Aplicaciones de la red Hopfield
- 7.3. Ventajas y limitaciones
- 7.4. Introducción red Bidirectional Associative Memory
- 7.5. Arquitectura red B.A.M.

TEMA 7.- RED HOPFIELD Y RED BIDIRECTIONAL ASSOCIATIVE MEMORY

7.1.- RED HOPFIELD

Estructura Básica.-

La red Hopfield tiene una única capa de unidades procesadoras. Cada una de las unidades procesadoras tiene un valor o nivel de activación, también llamado estado, que es binario (la red presentada en 1982 se llama Red Hopfield Binaria).

Se considera que la red Hopfield tiene un estado en cada momento; este estado se define por el vector de unos y ceros constituido por los estados de todas las unidades procesadoras. El estado de una red con n unidades procesadoras, donde el elemento i tiene el estado u_i se representa según la ecuación (7.1)

$$U = (u_1, u_2, \dots, u_n) = (+++--\dots++) \quad \text{ec.(7.1)}$$

En esta notación el signo $+$ representa una unidad procesadora con el estado o valor binario 1 y el signo $-$ representa una unidad procesadora con el estado o valor binario 0.

Las unidades procesadoras de la red Hopfield están completamente interconectadas, cada unidad está conectada con todas las demás unidades. Esta topología convierte a la red Hopfield en una red recursiva ya que la salida de cada unidad está realimentada con las entradas de las demás unidades.

La Figura (7.1) muestra un diagrama de las unidades procesadoras de una red Hopfield y un ejemplo del estado de la red.

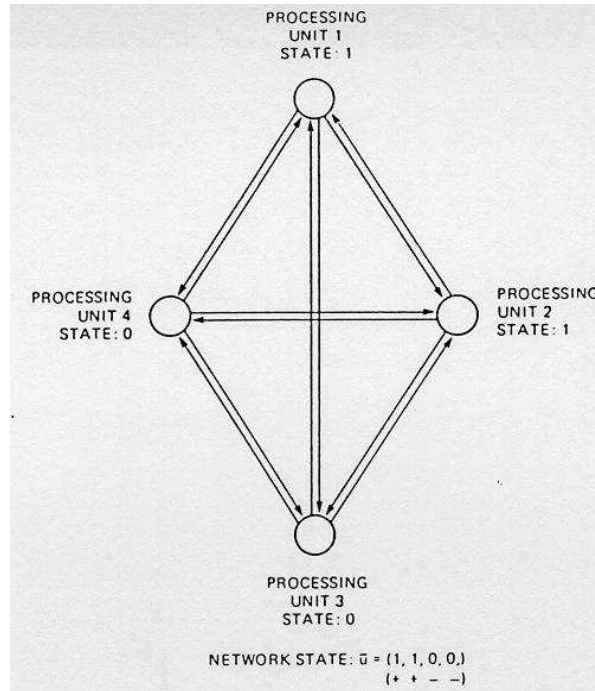


Figura (7.1) - Red Hopfield Binaria.

Una característica de las redes Hopfield es la doble conexión por cada pareja de unidades procesadoras, como se aprecia en la figura anterior. Además los pesos asignados a ambas conexiones tienen el mismo valor.

La Figura (7.2) muestra un método alternativo de representación de la estructura y conexiones de la red Hopfield.

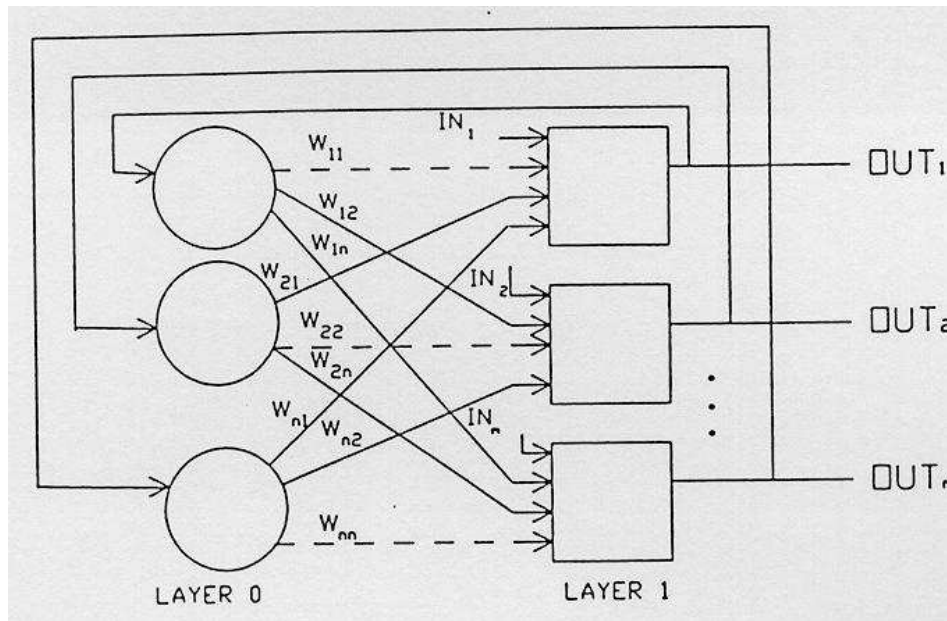


Figura (7.2) - Red Recursiva de una Capa.

Procedimiento de Actualización.-

Inicialmente, la red tiene asignado un estado para cada unidad de proceso. El procedimiento de actualización se aplica a todas las unidades de una en una. Este procedimiento afecta al estado de cada unidad modificándolo o manteniéndolo constante. Este procedimiento de actualización permanece hasta que no se produzca ninguna modificación en la red.

El modo de operación de la red se puede visualizar geoméricamente. Para un caso genérico de n neuronas, el número de estados posibles es 2^n y se le asocia un hipercubo de n dimensiones. Cuando se le presenta una nueva entrada, la red se mueve de un vértice a otro hasta que se estabiliza. El vértice estable está definido por los pesos de la red, las entradas actuales y el valor umbral de la función f de las neuronas. Si el vector entrada está parcialmente incompleto o es parcialmente incorrecto, la red se estabiliza en el vértice más próximo al vértice deseado.

Convergencia/Estabilidad.-

Una de las principales aportaciones de John Hopfield es su visión del estado de una red como una superficie de energía. La energía asociada a cada estado de la red se describe en la ecuación (7.2)

$$E = -1/2 \sum_{j=i} \sum_i w_{ij} OUT_i OUT_j \quad \text{ec.(7.2)}$$

La continua actualización de la red Hopfield produce un proceso convergente, y de esta manera la energía global de la red se hace cada vez más pequeña hasta que finalmente la

red alcanza un estado estable. En este estado estable la energía está en un mínimo que puede ser local o global.

Se demuestra, a partir de la ecuación (7.3) que en cada instante en el que una unidad procesadora se actualiza, la energía de la red permanece invariante o decrece.

$$\Delta E_i = -1/2 \Delta OUT_i \sum_j (w_{ij} OUT_j) \quad \text{ec.(7.3)}$$

De esta forma, el procedimiento de actualización garantiza que la energía de la red converge en un mínimo.

En la red Hopfield no existe un modo de alcanzar el mínimo global desde un mínimo local. Una red diferente como la Boltzman Machine utiliza "ruido" para sacar a la red de un mínimo local. No obstante la red Hopfield puede llegar a alcanzar el mínimo global partiendo de una posición inicial diferente.

Memoria Asociativa.-

La memoria humana funciona de una manera asociativa. A partir de una porción de información es capaz de obtener la información completa. Por ejemplo, escuchando los primeros acordes de una canción el cerebro es capaz de reproducir toda una experiencia completa, incluyendo escenas, ruidos y olores.

Una red recursiva constituye una memoria asociativa. Al igual que el humano, si se le presenta una porción de datos es capaz de recuperar todos los datos. Para realizar una memoria asociativa mediante una red recursiva (Hopfield propuso originalmente esta aplicación para su red binaria), es necesario elegir los pesos de forma que produzcan un mínimo de energía en los vértices deseados del hipercubo. Cada vector de estado correspondiente a un mínimo de energía se llama "memoria".

La red parte de un estado inicial y el propio procedimiento de actualización mueve el estado de la red hasta llegar a un estado de energía mínimo. Este mínimo se supone que corresponde a una "memoria" de la red. Entonces, la red converge en una memoria almacenada que es la más similar o la más accesible al estado inicial.

En aplicaciones de memoria asociativa, se elige a priori los patrones que van a ser almacenados como memorias. El número de unidades procesadoras es igual al número de elementos del vector que representa el patrón que va a ser almacenado. Los pesos se fijan en función de los patrones elegidos. La regla de aprendizaje de Hopfield es la indicada en la ecuación (7.4)

$$w_{ij} = \sum_p (2x_{ip} - 1) (2x_{jp} - 1) \quad \text{ec.(7.4)}$$

Según esta ecuación, las conexiones son reforzadas cuando la salida de la unidad procesadora es igual que la entrada. Sin embargo las conexiones son reducidas cuando la entrada difiere de la salida de la unidad procesadora.

7.2.- APLICACIONES DE LA RED HOPFIELD

En 1984 Hopfield extendió el diseño de la red Hopfield binaria obteniendo la red Hopfield Continua que se caracteriza por el rango continuo de valores de salida que pueden presentar las unidades procesadoras.

Esta nueva red mantiene la topología de la red binaria y la principal diferencia es la función umbral que utiliza cada unidad procesadora. La función umbral elegida para esta red es la función continua Sigmoid expresada en la ecuación (7.5)

$$f(NET) = 1 / (1 + e^{-\lambda NET}) \quad \text{ec.(7.5)}$$

Al igual que en los sistemas Hopfield binario, la estabilidad se asegura si los pesos son simétricos y la diagonal principal es nula.

Las aplicaciones de la red Hopfield tanto en problemas de asociación de memoria como en los problemas de optimización quedan satisfactoriamente resueltos cuando la red alcanza un estado estable en un mínimo de energía.

Un ejemplo típico y muy ilustrativo de la capacidad de la red Hopfield en problemas de optimización es el conocido problema del vendedor ambulante (*traveler sales person*, TSP). Este problema es un problema de extrema dificultad de optimización clásica; es un problema de la clase NP completa (*non deterministic polynomial*). Este problema NP no tiene un método conocido para obtener la solución mejor que el de probar todas las posibles alternativas. Este procedimiento requiere una gran cantidad de tiempo de cálculo. Hopfield y Tank encontraron un modo de abordar este problema utilizando la red Hopfield continua. Esta red encuentra una buena solución al problema TSP en un tiempo razonable.

El problema TSP se define de la siguiente manera: Un vendedor tiene un número de ciudades que visitar. El vendedor comienza y acaba en una ciudad concreta, y viaja a todas las demás ciudades sin estar dos veces en ninguna de las ciudades. El objetivo es encontrar la ruta que ha de seguir el vendedor de manera que la distancia recorrida sea mínima.

La forma de representar la ruta que debe realizar el vendedor de este problema con una red Hopfield es mediante una matriz de ceros y unos. Las filas de la matriz representan diferentes ciudades y las columnas de la matriz representan las diferentes posiciones de cada ciudad dentro de la ruta. La solución es un conjunto de n ciudades ordenadas como se indica en la Figura (7.3)

La red Hopfield se realiza con tantas unidades procesadoras como elementos tiene la matriz, es decir $n \times n$. El objetivo de esta aplicación es obtener el valor de las $n \times n$ unidades procesadoras de un estado estable de la red Hopfield que represente una ruta que sea una buena solución para el problema TSP.

POSITION IN TOUR										
CITY	1	2	3	4	5	6	7	8	9	10
A	0	0	0	1	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	1	0	0	0
D	0	0	0	0	1	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	1
F	0	0	0	0	0	0	0	1	0	0
G	1	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	1	0	0	0	0
I	0	0	0	0	0	0	0	0	1	0
J	0	0	1	0	0	0	0	0	0	0

TOUR: G B J A D H C F I E

Figura (7.3) - Matriz del problema TSP.

La función de energía de la red debe cumplir dos requisitos: Primero, debe ser pequeña sólo para aquellas soluciones que presenten un único uno en cada columna y en cada fila de la matriz. Segundo, debe favorecer las soluciones que presenten una distancia corta.

El primer requisito se logra con los tres primeros términos de la ecuación de energía de la red Hopfield expresada en la ecuación (7.6), y el cuarto término satisface el segundo requerimiento.

$$\begin{aligned}
 E = & A/2 \sum_X \sum_i \sum_{j \neq i} OUT_{X,i} OUT_{X,j} \\
 & + B/2 \sum_i \sum_X \sum_{Y \neq X} OUT_{X,i} OUT_{Y,i} \\
 & + C/2 [(\sum_X \sum_i OUT_{X,i})^2 - n] \\
 & + D/2 \sum_X \sum_{Y \neq X} \sum_i d_{X,Y} OUT_{X,i} (OUT_{Y,i+1} + OUT_{Y,j-1})
 \end{aligned}
 \tag{7.6}$$

Siendo A, B, C y D valores constantes ajustables y $OUT_{X,i}$ es la salida de la unidad procesadora identificada por los subíndices X e i, que indican que la ciudad X ocupa la i-ésima posición en la ruta.

Para valores suficientemente grandes para A, B, y C se consigue que los estados de baja energía representen recorridos válidos, y para un valor alto de D se asegura que se encontrará un recorrido corto.

El siguiente paso es obtener o calcular el valor de los pesos relacionándolos con los términos de la función energía a partir de la ecuación (7.7).

$$\begin{aligned}
 W_{xi,yi} = & - A \delta_{x,y} (1 - \delta_{i,j}) \\
 & - B \delta_{x,y} (1 - \delta_{i,j}) \\
 & - C \\
 & - D \delta_{x,y} (\delta_{j,i+1} + \delta_{j,i-1})
 \end{aligned}
 \quad \text{ec.(7.7)}$$

La Figura (7.4) muestra la evolución de la convergencia de la red para un ejemplo de diez ciudades.

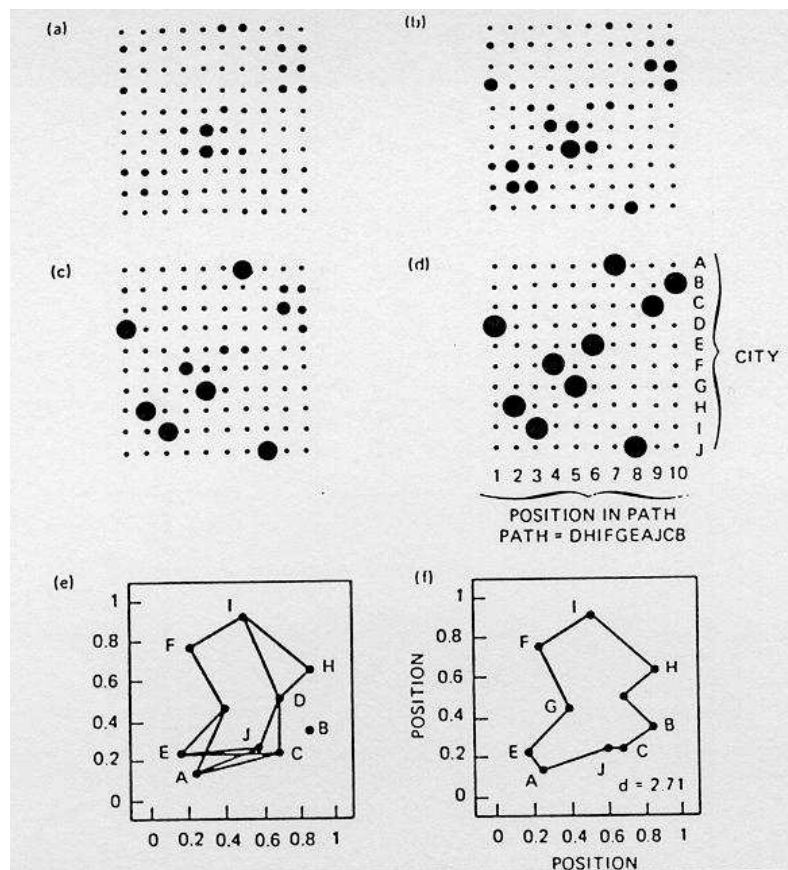


Figura (7.4) - Solución del problema TSP.

7.3.- VENTAJAS Y LIMITACIONES

Las dos aplicaciones de la red Hopfield estudiadas, memoria asociativa y optimización, quedan satisfactoriamente resueltas cuando la red alcanza un estado estable en un mínimo de energía.

No obstante ambas aplicaciones de la red Hopfield presentan algunas limitaciones. La existencia de mínimos locales puede producir que en la aplicación de memoria asociativa una memoria evocada por la red no sea necesariamente la memoria patrón más parecida al patrón

de entrada, o en el problema TSP la solución obtenida sea una solución buena pero no la óptima. Otro tipo de limitación es la capacidad de memoria de la red, es decir el número de memorias que puede almacenar la red. Aunque una red de N neuronas puede tener muchos más estados que 2^N , la realidad es que la máxima capacidad obtenida es mucho menor que ese valor. Si la red almacena demasiadas memorias, la red no se estabiliza en ninguna de ellas y además puede que recuerde estados que nunca le han sido enseñados.

La mayor ventaja de esta red es su rápida capacidad computacional. Esta rapidez se debe a la naturaleza altamente paralela del proceso de convergencia. Además el tiempo de convergencia varía poco con el tamaño del problema, mientras que los métodos convencionales aumentan el tiempo de procesamiento exponencialmente con el tamaño del problema.

Existen muchas aplicaciones posibles para la red Hopfield. Entre ellas destacan el procesamiento de voz, recuperación de base de datos, procesamiento de imagen, memorias con tolerancia a fallos y clasificación de patrones.

7.4.- INTRODUCCIÓN RED BIDIRECTIONAL ASSOCIATIVE MEMORY

El modelo de la red B.A.M. (*Bidirectional Associative Memory*) fue desarrollada por Kosko (1987) aun cuando presenta varios aspectos inspirados en el trabajo de Grossberg (1982).

También se puede considerar esta red como la generalización del modelo de Hopfield en redes heteroasociativas. La red BAM es heteroasociativa en cuanto que acepta un vector de entrada en un conjunto de neuronas y produce otro vector relacionado, un vector de salida en otro conjunto de neuronas. Sin embargo la red de Hopfield debido a su única capa de neuronas requiere que el vector de salida aparezca en las mismas neuronas en las que se aplica el vector de entrada, de aquí el carácter de red o memoria autoasociativa.

7.5.- ARQUITECTURA RED B.A.M.

La red tiene dos capas centrales de neuronas totalmente interconectadas además de las capas buffer de entrada y salida. La Figura (7.5) muestra la estructura de una red BAM siguiendo la forma de la red de Hopfield (no se muestran las capas de entrada y salida).

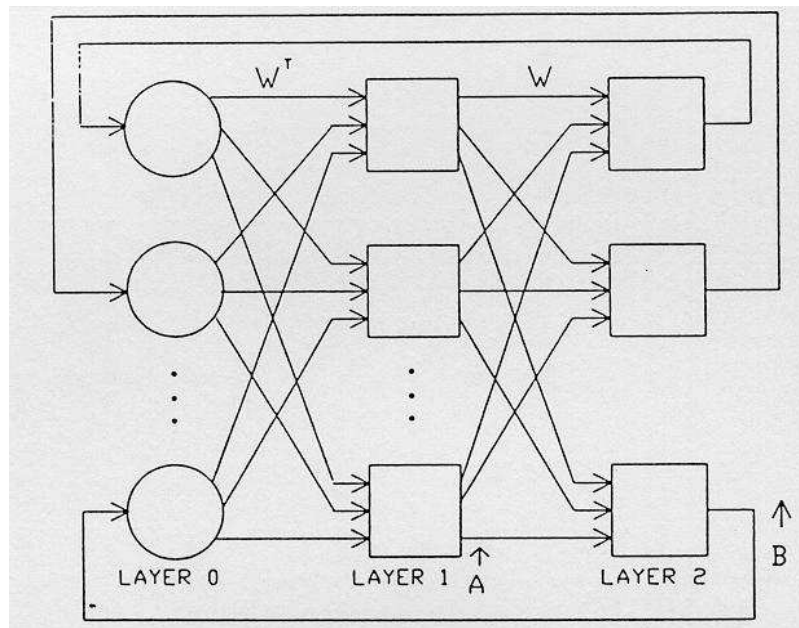


Figura (7.5) - Estructura simplificada de la Red BAM.

La red se diseña para almacenar parejas asociadas de vectores. Los pesos de las conexiones entre las capas centrales almacenan la información asociativa. Si la primera capa tiene N neuronas y la segunda capa tiene M neuronas los pesos de estas conexiones se almacenan en una matriz W de orden $N \times M$. En esta arquitectura si se aplica un vector de entrada A a los pesos W (es decir a la salida de la primera capa) se produce un vector de salida B . Cuando se aplica el vector B a la matriz traspuesta W^T se produce nuevas salidas para el vector A . Este proceso se repite hasta que la red alcanza un punto estable en el que A y B no cambian.

Recuperación de las Asociaciones Almacenadas.-

Una vez definida la arquitectura de la red, ésta debe ser entrenada para reconocer o recordar una serie de memorias o vectores. El conjunto de entrenamiento está constituido por parejas de vectores A y B . El entrenamiento consiste en el cálculo de la matriz de pesos W según la ecuación (7.8)

$$W = \sum_i A_i \wedge B_i \quad \text{ec.(7.8)}$$

Este cálculo lo realizan las redes BAM modificando sus pesos a lo largo de la fase de entrenamiento utilizando la regla de Hebb.

Las asociaciones o memorias de la red se almacenan en las matrices W y W^T . Para recuperar una asociación basta con presentar parcial o totalmente el vector A en la salida de la primera capa.

La red a través de W produce un vector de salida B en la capa segunda. Este vector opera sobre la matriz traspuesta W^T produciendo una réplica próxima al vector de

entrada A en la salida de la capa primera. Cada paso en el lazo de la red produce que los vectores de salida de ambas capas sean cada vez más próximas a la memoria o asociación almacenada. Este proceso se lleva a cabo hasta que se alcance un punto estable llamado resonancia en el que los vectores pasan hacia adelante y hacia atrás reforzando las salidas sin modificarlas.

La relación estrecha entre la red BAM y la red Hopfield se manifiesta en el caso de que la matriz sea cuadrada y simétrica ($W = W^T$). Para este caso si las capas primera y segunda tienen el mismo conjunto de neuronas la red BAM queda reducida a la red autoasociativa de Hopfield.

RED ADAPTIVE RESONANCE THEORY

8

- 8.1. Introducción red Adaptive Resonance Theory
- 8.2. Arquitectura red A.R.T.
- 8.3. Modo de Operación
- 8.4. Entrenamiento de la red A.R.T.

TEMA 8.- RED ADAPTIVE RESONANCE THEORY

8.1.- INTRODUCCIÓN RED ADAPTIVE RESONANCE THEORY

La *Adaptive Resonance Theory* (A.R.T.) tiene sus orígenes a mediados de los años 60 y tanto la síntesis como la extensión de las ideas de Grossberg (1976) constituyen la base de la resonancia adaptativa. Esta red presenta algunas características basadas en las neuronas biológicas y en particular resuelve satisfactoriamente el dilema estabilidad-plasticidad característico del cerebro humano. ¿Cómo el cerebro se muestra flexible para almacenar nuevas memorias que le llegan, y por otra parte es capaz de retener las memorias ya almacenadas sin borrarlas?

En aplicaciones reales las redes están expuestas a un entorno constantemente cambiante; un mismo vector de entrenamiento de entrada puede que nunca se presente dos veces. Así en estas circunstancias ocurre que algunas redes no aprenden nada y están continuamente modificando sus pesos sin alcanzar el objetivo.

Las redes ART presentan la plasticidad o flexibilidad necesaria para aprender nuevos patrones y evitan las modificaciones en los patrones aprendidos previamente.

8.2.- ARQUITECTURA RED A.R.T.

La red ART tiene dos modelos, ART1 que acepta solo vectores de entradas binarias, y ART2 que admite también entradas continuas (analógicas). Se presenta el modelo primero que es más sencillo e ilustra los principales aspectos de la red ART.

El núcleo de la red ART consiste en dos capas interconectadas como se muestra en la Figura (8.1) y en una serie de bloques que realizan funciones de control requeridas en las fases de entrenamiento y clasificación.

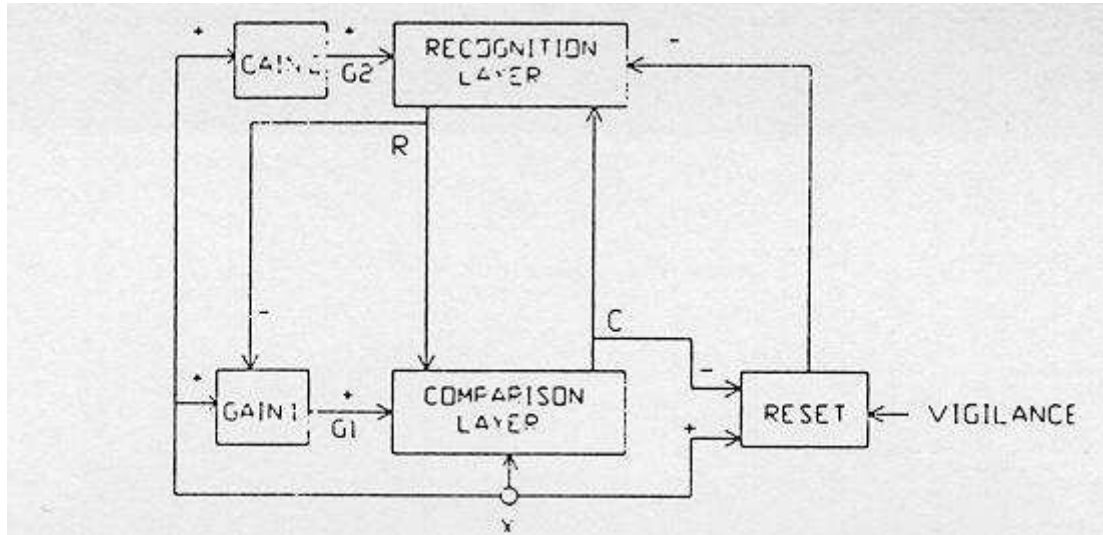


Figura (8.1) - Red ART Simplificada.

Esta red es un clasificador de vectores; acepta vectores de entrada y los clasifica en una de las categorías posibles en función del patrón almacenado al que más se aproxime. La capa de Reconocimiento es la responsable de indicar la categoría. En el caso de que el vector de entrada no se aproxime suficientemente a ningún patrón almacenado se crea una nueva categoría almacenando un patrón idéntico a la entrada. Una vez encontrado un patrón que se parezca al vector de entrada dentro de una tolerancia especificada (parámetro de vigilancia), se ajusta o se entrena dicho patrón para hacerlo más parecido todavía al vector de entrada.

Capa de Comparación.-

La Figura (8.2) muestra la capa de comparación simplificada. Inicialmente el vector binario de entrada X atraviesa la capa sin cambio alguno y pasa a convertirse en el vector C. En una fase posterior la capa de Reconocimiento genera el valor R modificando a continuación el vector C.

Cada neurona de esta capa tienen tres entradas, el vector de entrada X, la suma ponderada del vector R y la ganancia G1; el valor de activación de estas neuronas es de valor uno si al menos dos de las entradas tienen valor uno (regla de "dos tercios").

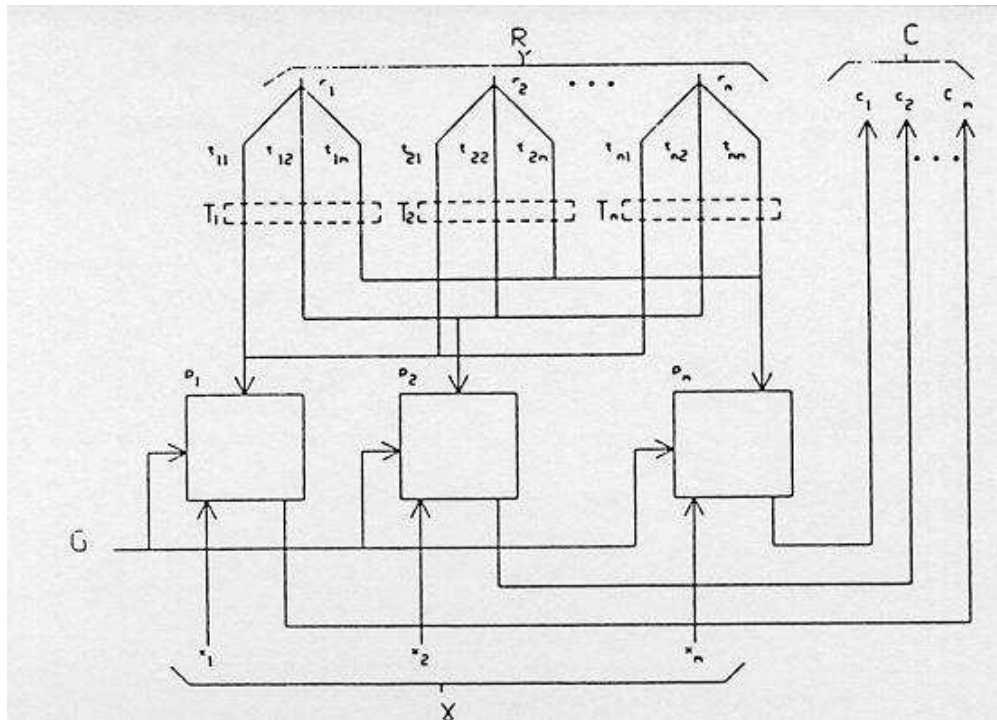


Figura (8.2) - Capa de Comparación Simplificada.

Capa de Reconocimiento.-

Esta capa realiza la clasificación del vector de entrada. Como se muestra en la Figura (8.3) cada neurona tiene asociado un vector de pesos B .

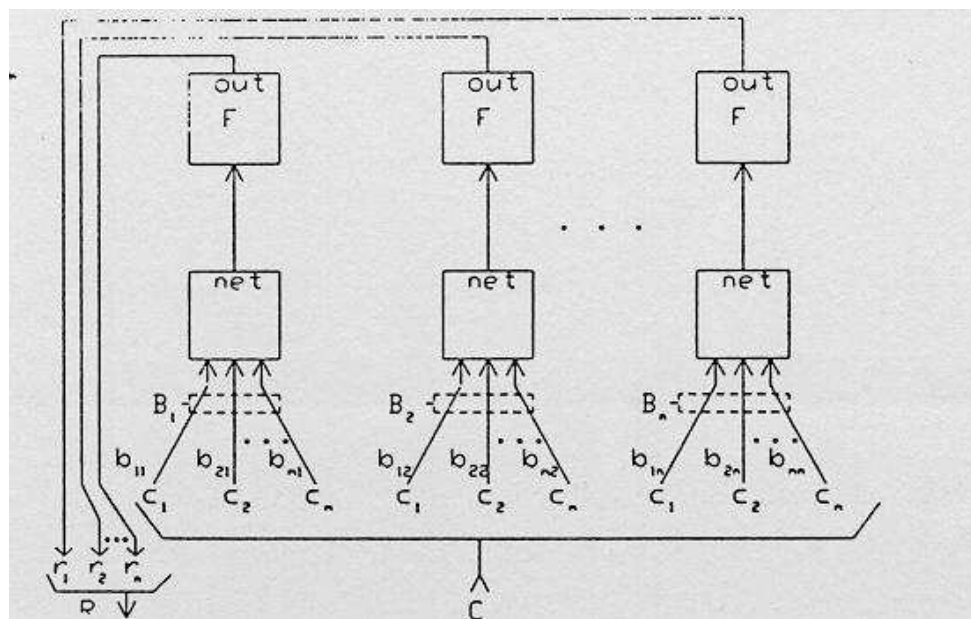


Figura (8.3) - Capa de Reconocimiento Simplificada.

En esta capa sólo se activará la neurona que tenga el vector de pesos B más próximo al vector de entrada. Las demás neuronas quedarán inhabilitadas por las conexiones laterales de inhibición existentes entre ellas.

Reset.-

Este módulo mide la similitud entre los vectores X y C . Si difieren en una cantidad mayor que el parámetro de vigilancia se activa la señal de reset para deshabilitar la neurona activada en la capa de Reconocimiento.

Módulos de Ganancia $G1$ y $G2$.-

Estos módulos presentan su salida con valor uno si alguna componente del vector X es uno, excepto en el caso de que si además R tiene algún componente a uno el módulo $G1$ presentará la salida con valor cero.

8.3.- MODO DE OPERACIÓN

El modo de operación de una red ART tiene cinco fases: Inicialización, Reconocimiento, Comparación, Búsqueda y Entrenamiento.

La fase de inicialización asigna los valores iniciales de los pesos B_j , T_j y del parámetro de vigilancia ρ antes de realizar la fase de entrenamiento.

La aplicación de un vector X a la entrada da inicio a la fase de reconocimiento. Al comienzo el vector C es idéntico a X . El reconocimiento se realiza como el producto escalar propio de cada neurona de la capa de reconocimiento cuya entrada presente valor uno. La existencia de la inhibición lateral garantiza la activación de la neurona que presente el mayor valor del producto escalar.

En este punto la señal de realimentación de la capa de reconocimiento provoca que $G1$ sea cero, y por consiguiente la comparación sólo se realizará en las neuronas cuyos componentes de los vectores P y X tengan el valor uno.

Además el bloque de Reset compara el vector C y X , y provoca una señal de reset a aquellas neuronas de la capa de reconocimiento que no superen el umbral del parámetro de vigilancia. Si la similitud S de la neurona ganadora es mayor que el parámetro de vigilancia entonces no es necesario la fase de búsqueda. En caso contrario es necesario buscar entre los patrones almacenados cuál de ellos se aproxima más al vector de entrada, o determinar qué neurona representará dicha entrada.

El entrenamiento es el proceso de presentar secuencialmente un conjunto de vectores a la entrada de la red y ajustar los pesos para que vectores similares activen la misma neurona de la capa de reconocimiento. Este tipo de entrenamiento es no supervisado.

8.4.- ENTRENAMIENTO DE LA RED A.R.T.

El entrenamiento además de ajustar los pesos B y T para que al aplicar a la entrada un patrón se active la neurona de la capa de Reconocimiento asociada a un patrón almacenado similar, evita la destrucción parcial o total de los patrones previamente almacenados, desapareciendo por tanto cualquier inestabilidad temporal.

Una entrada que nunca haya sido mostrada a la red no encontrará un patrón almacenado similar dentro del parámetro de vigilancia y por consiguiente provocará que una neurona libre almacene este nuevo patrón. Si la entrada fuera suficientemente parecida a algún patrón almacenado no provocará que sea almacenado como un nuevo patrón sino que modificará los pesos de una neurona para que el parecido sea mayor.

La Figura (8.4) muestra una sesión de entrenamiento de una red ART. A la red se le muestra letras que están representadas como pequeños cuadrados de una rejilla de 8x8. En la parte izquierda se representa el conjunto de vectores de entrada y en la parte derecha se representan los patrones almacenados.

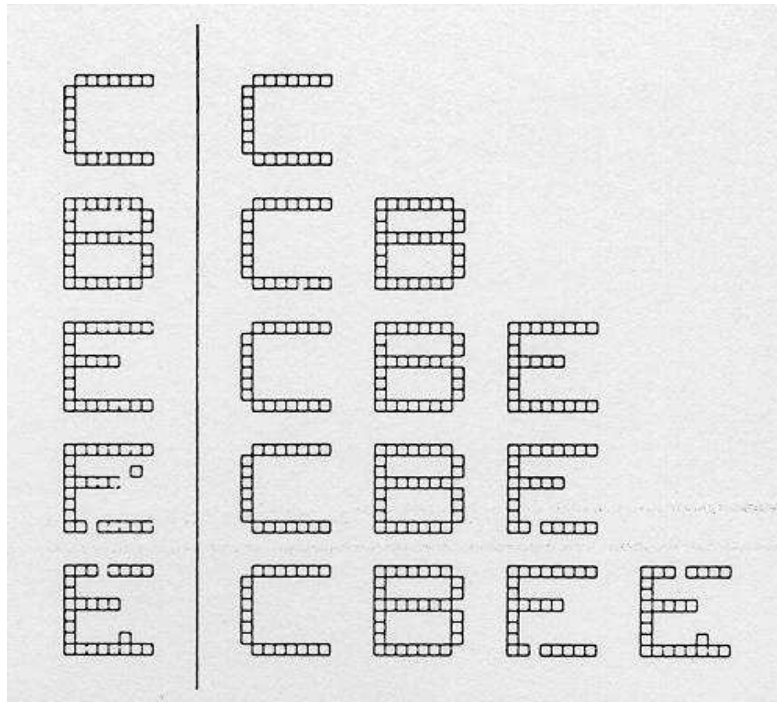


Figura (8.4) – Entrenamiento de una red ART.

APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES

9

- 9.1. Introducción
- 9.2. Diseño de una Red para una Aplicación
- 9.3. Ejemplos de Aplicaciones

TEMA 9.- APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES

9.1.- INTRODUCCIÓN

En este tema se recogen varios ejemplos de aplicaciones de las redes neuronales artificiales consideradas como sistemas que resuelven eficazmente problemas de emparejamiento, clasificación y complemento de vectores patrones.

Entre las áreas de aplicación de las redes se encuentran entre otras las siguientes: Análisis Financiero; Procesado de Imágenes en el ámbito de la Medicina, Industria y Defensa; Diagnóstico Médico y Comercial; Robótica y Control; Reconocimiento y Síntesis de Voz; Clasificación de Datos provenientes de sensores; Compresión y Codificación de Información.

No obstante conviene matizar la palabra Aplicaciones y diferenciar entre aplicaciones candidatas, aplicaciones en desarrollo y aplicaciones ya demostradas. Dicho de otro modo es necesario puntualizar el estado de desarrollo, de realización y de comprobación de las redes neuronales utilizadas en cada aplicación.

Las aplicaciones candidatas son aquellos problemas que en principio podrían ser resueltos con este tipo de tecnología que ofrecen las redes neuronales artificiales. Las aplicaciones en desarrollo son aquellas en las que se han realizado los estudios oportunos del problema y se dispone de un prototipo de red al que se le ha entrenado para resolver una versión simplificada del problema. Por último las aplicaciones demostradas son redes que de hecho ya están siendo utilizadas para resolver un problema real.

9.2.- DISEÑO DE UNA RED PARA UNA APLICACIÓN

A la hora de diseñar nuestra red neuronal para resolver un problema concreto es conveniente disponer de una herramienta software de diseño de ANN. Con una herramienta de éstas basta con pensar en términos de redes y no en programación de algoritmos en lenguajes de alto nivel. De esta manera todo el esfuerzo se debe dirigir al diseño de la arquitectura o estructura de la red y en la selección de los datos del conjunto de entrenamiento y de test.

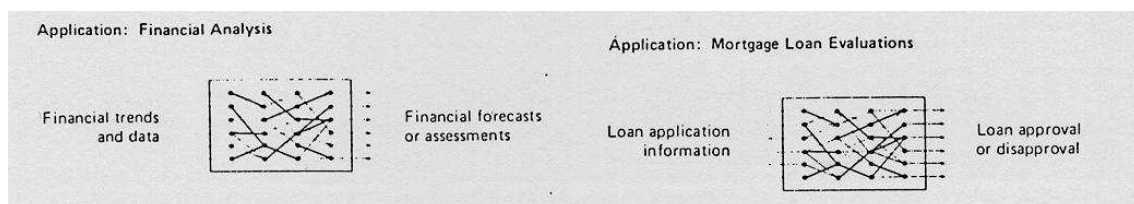
El diseñador construye con el software apropiado la red especificando el número de capas, de neuronas y los tipos de conexiones. Define los ficheros o conjuntos de datos de entrada y salida, y debe elegir los parámetros de los cálculos internos de la red. Además el diseñador puede seleccionar diferentes funciones de transferencia y procesamiento de las neuronas, así como construir variaciones de los modelos estándar.

En la fase de entrenamiento se debe especificar el número de iteraciones y la planificación de los cambios de los parámetros de aprendizaje. Generalmente esta fase requiere varias sesiones y la experimentación de diferentes parámetros de aprendizaje, diferentes vectores de entrada o diversas estrategias de entrenamiento permiten obtener conclusiones definitivas para la solución más eficaz de una aplicación.

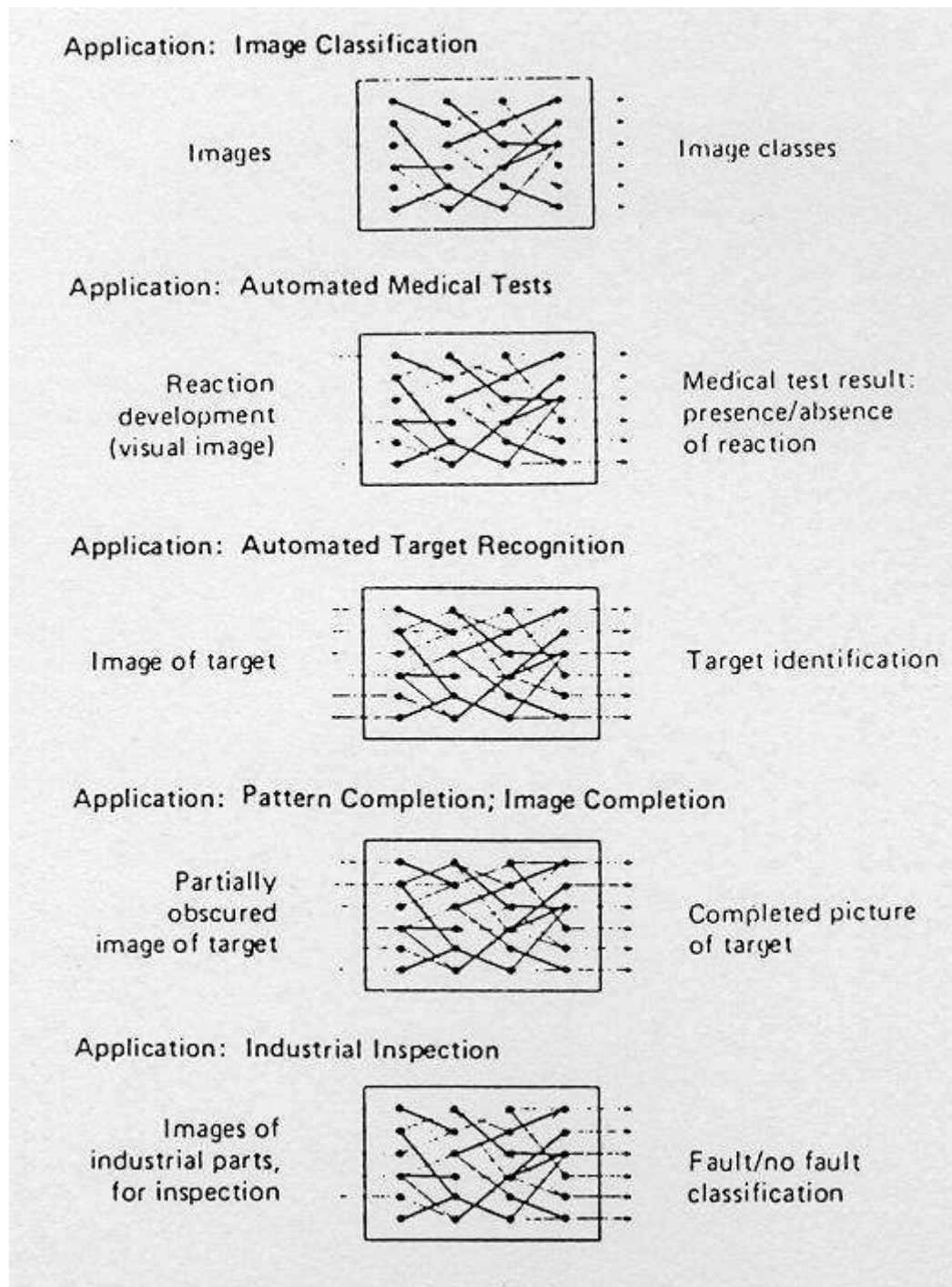
Afortunadamente la disposición de este tipo de software apropiado por el diseñador de ANN permite que el diseñador no se preocupe de los aspectos computacionales y disponga de todo el tiempo tanto para la elección de la arquitectura como para la selección y preprocesado de los datos presentados a la red. Este último aspecto es uno de los factores más influyentes en el éxito del diseño y realización de una red para una aplicación.

9.3.- EJEMPLOS DE APLICACIONES

- Análisis Financiero



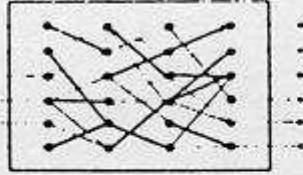
- Procesado de Imágenes



- Diagnóstico

Application: Diagnosis

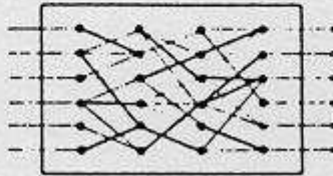
Diagnostic
test results-
data and
sensor readings



Diagnosis,
fault evaluation

Application: Dermatology Diagnosis

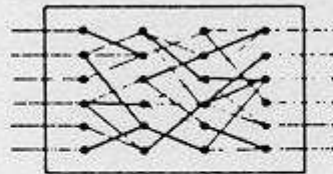
Dermatology
symptoms



Diagnosis of
skin disease

Application: Jet Engine Fault Diagnosis

Sensor data
from jet
engine



Identification
of fault type

Application: Automotive Control System Diagnostics

Automotive
engine sensor
data and
observations

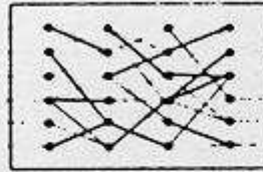


Identification
of fault type

- Control y Robótica

Application: Broom Balancing

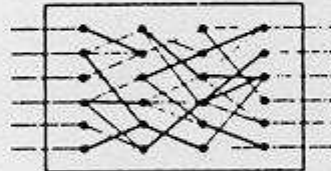
Current broom
position and
recent history



Next movement
of cart

Application: Robot Control

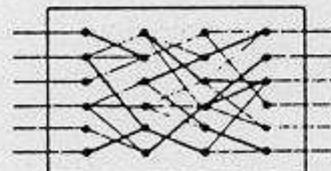
Visual sensors



Robot arm
commands

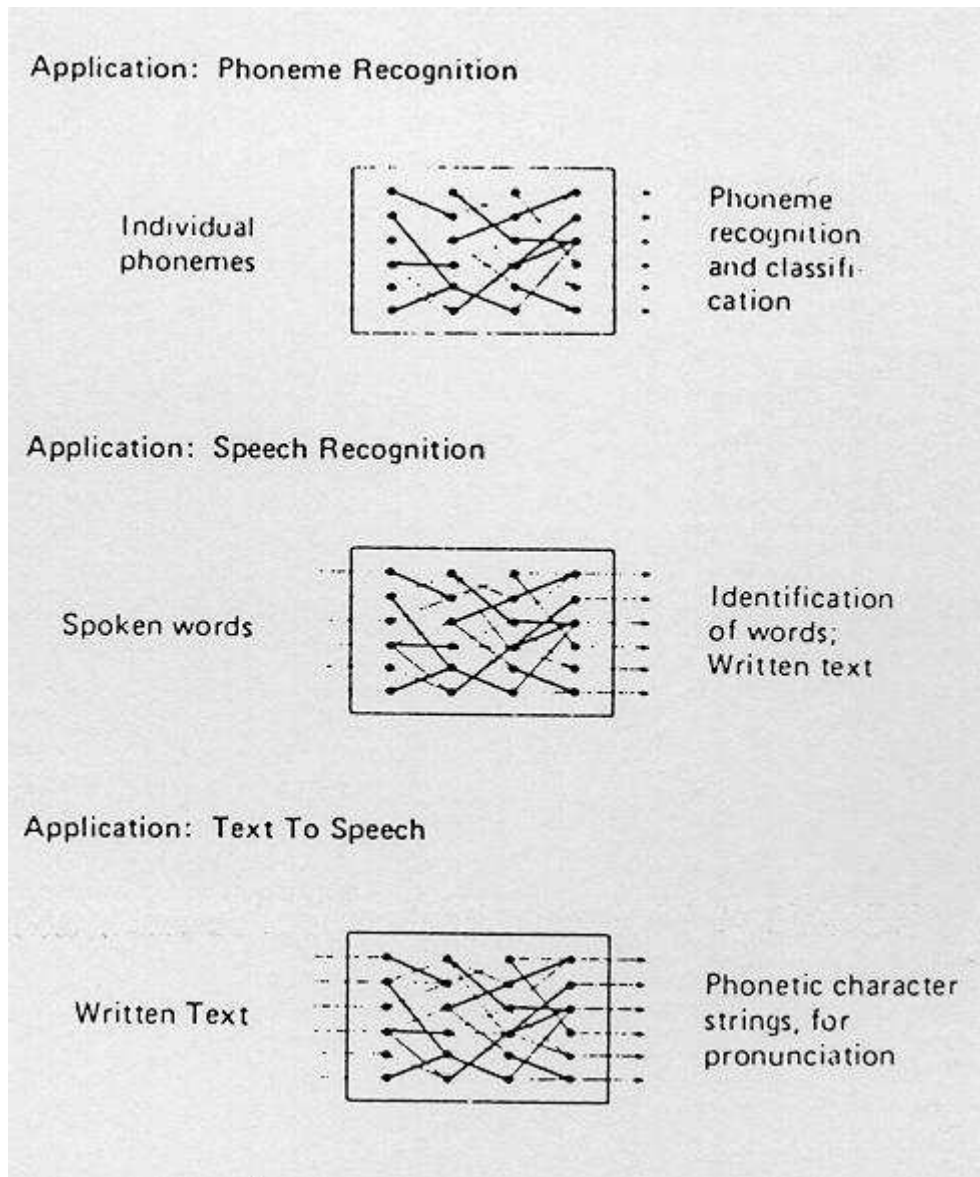
Application: Robot Control

Desired target
position

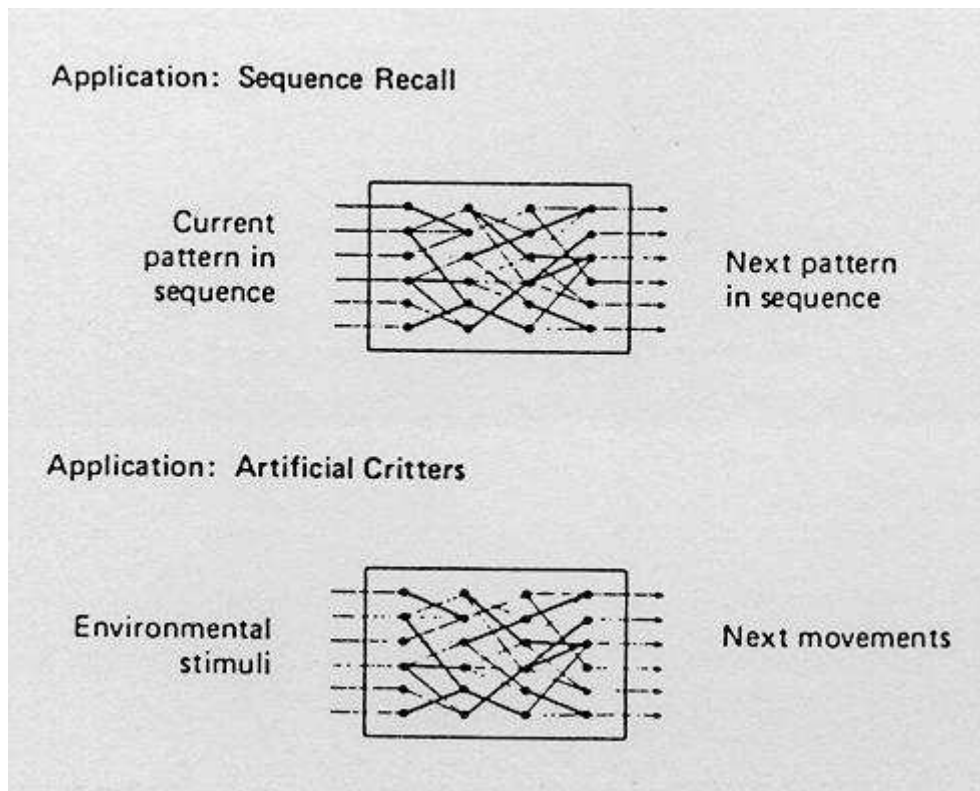


Robot arm
commands

- Procesado de Voz



- Otras Aplicaciones



LÓGICA DIFUSA Y REDES NEURONALES ARTIFICIALES

10

- 10.1. Introducción
- 10.2. Estructura General de un Sistema basado en Lógica Borrosa
- 10.3. Sistemas Neuro-Difusos

TEMA 10.- LÓGICA DIFUSA Y REDES NEURONALES ARTIFICIALES

10.1.- INTRODUCCIÓN

La Lógica Difusa creada por el matemático Zadeh en 1965 ha emergido en los últimos años con un considerable éxito dentro del contexto del Control Inteligente. No todos los problemas de control pueden ser formulados mediante los métodos utilizados en el control convencional; para resolverlos de forma sistemática se han desarrollado un número de métodos que de forma colectiva se les llama Metodologías de Control Inteligente. Entre las áreas de investigación más relevantes del Control Inteligente se encuentran las Redes Neuronales Artificiales y la Lógica Difusa.

La proliferación de artículos, congresos, aplicaciones y productos sobre esta nueva tecnología exige a todo investigador tener el conocimiento suficiente de ella para saber cuando es apropiado la utilización de dicha lógica. De forma resumida se puede decir que la Lógica Difusa conviene utilizarla cuando se produce alguna de las siguientes condiciones: las variables de control son continuas, no existe un modelo matemático del proceso o es difícil decodificarlo, o bien el modelo es complejo y difícil de evaluarlo en tiempo real. También es recomendable cuando la aplicación requiere utilizar sensores y microprocesadores simples o cuando se dispone de una persona experta que puede especificar en reglas el comportamiento del sistema.

Desde el punto de vista de aplicación esta nueva tecnología conviene considerarla como un método alternativo serio a la lógica clásica. Ello requiere un cambio de mentalidad y saber que la precisión no siempre es imprescindible. Basta para ello pensar en la mayoría de las acciones cotidianas que realizamos a lo largo del día, como conducir el

coche, realizar cualquier tipo de deporte, cocinar, etc. donde no es necesario conocer con precisión la velocidad del coche, el impulso del balón, o el tiempo de cocción para poder llevar acabo con éxito las diferentes acciones.

Teoría de Conjuntos Borrosos

En la teoría clásica de conjuntos cualquier elemento perteneciente a un Universo X pertenece o no pertenece a un subconjunto A incluido en X , sin que exista otra posibilidad al margen de esas dos.

La pertenencia o no pertenencia de un elemento arbitrario a un subconjunto A viene dada en la mayoría de los casos por la verificación o no de un predicado que caracteriza a A y da lugar a una bipartición del universo del discurso X .

La función de pertenencia representa numéricamente la pertenencia o no pertenencia de un elemento a un conjunto A . Esta función expresada en la ecuación (10.1) asigna a cada elemento x del discurso un número, 1 ó 0, según x pertenezca o no pertenezca al conjunto A .

$$\mu_A : X \rightarrow \{0,1\} \quad \mu_A(x) = \begin{cases} 1 & \text{si } x \text{ pertenece a } A \\ 0 & \text{si } x \text{ no pertenece a } A \end{cases} \quad \text{ec. (10.1)}$$

Esta función de pertenencia desempeña un papel clave en el estudio de la teoría de Conjuntos Borrosos. Una nueva definición matemática de la función pertenencia caracterizará a los conjuntos difusos que presentan un predicado menos preciso y más genérico.

La mayoría de las veces los conjuntos clásicos se definen mediante un predicado que da lugar a una perfecta bipartición del universo del discurso X . Sin embargo, el razonamiento humano utiliza frecuentemente predicados de los cuales no resulta una bipartición del universo. Así por ejemplo, en el universo del discurso X , el formado por todos los hombres de una ciudad se puede definir un subconjunto A como aquel que está formado por todos los hombres "altos". El predicado utilizado para caracterizar a los elementos de este subconjunto no separa el universo X en dos partes bien diferenciadas; ¿quiénes constituyen el subconjunto de hombres altos? ¿las personas que miden más de 1,80 metros? ¿Se puede decir que una persona que mida un centímetro menos que ese umbral es baja?

La forma adecuada de definir con claridad este tipo de problema es considerar que la pertenencia o no pertenencia de un elemento x al conjunto A no es absoluta sin gradual. El conjunto A constituye un conjunto borroso y su función de pertenencia asignará valores comprendidos en el intervalo cerrado $[0,1]$ en vez de sólo dos valores distintos, 0 y 1 que realizaba la función pertenencia de los conjuntos clásicos. La ecuación (10.2) expresa la nueva definición de la función pertenencia.

$$\mu_A : X \rightarrow [0,1] \quad \text{ec. (10.2)}$$

Normalmente la función pertenencia se define de forma heurística o arbitraria adquiriendo cualquier tipo de forma gráfica como refleja la Figura (10.1), destacando entre ellas las funciones que tienen forma triangular, trapezoidal y sigmoidal.

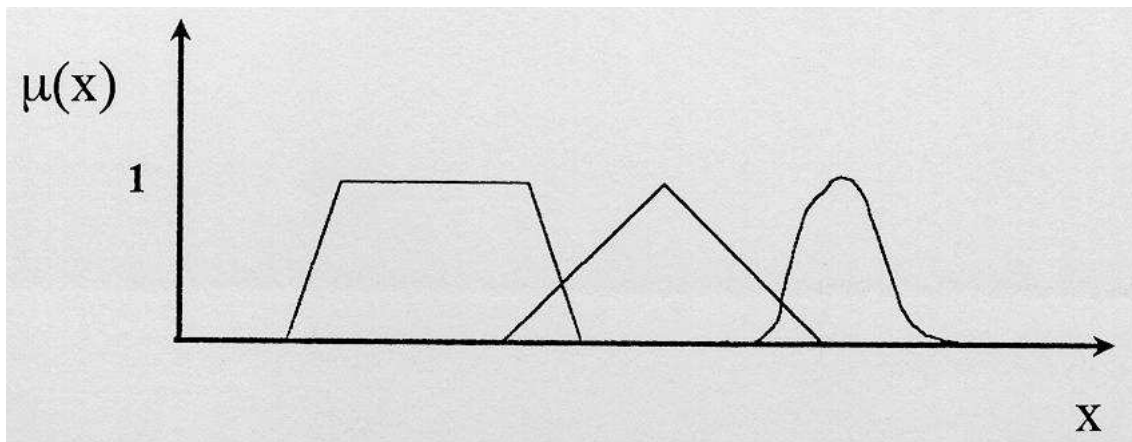


Figura (10.1) - Tipos de función de pertenencia de los conjuntos borrosos.

Los conjuntos borrosos presentan operaciones típicas como son intersección, unión y complemento; de esta forma se pueden realizar operaciones entre los conjuntos borrosos del mismo modo que se realizan entre los conjuntos clásicos.

La lógica clásica presenta un procesamiento de inferencia basado en la comparación del antecedente (hecho observado) con el condicionante de la regla. Si se verifica éste se infiere de manera inmediata el correspondiente consecuente. Si la verificación no es exacta no se puede inferir consecuente alguno.

Sin embargo la lógica borrosa presenta un razonamiento aproximado, es decir, el razonamiento trata con conceptos difusos o poco precisos desde el punto de vista de la lógica clásica. Con este tipo de razonamiento es posible inferir un consecuente aunque el antecedente no verifique la regla de forma completa. El consecuente obtenido será un concepto también borroso con su correspondiente función de pertenencia.

10.2.- ESTRUCTURA GENERAL DE UN SISTEMA BASADO EN LÓGICA BORROSA

Uno de los aspectos más interesantes que presenta el diseño de sistemas basados en lógica borrosa es su sencillez; no es necesario partir con un modelo matemático del sistema a controlar sino que basta con tener una idea general de cómo funciona el sistema. Para ello debe definirse los rangos de las variables de entrada y de salida, las funciones de pertenencia asociadas a cada una de ellas y el conjunto o base de reglas que describen el sistema.

Otro aspecto que interviene de forma notable en la estructura de este tipo de sistemas es la masiva, por no decir total, caracterización de los hechos observables a través de la

medición de las variables de los sistemas en términos de valores correspondientes a conjuntos clásicos. Este es el motivo por el que es necesario realizar en la entrada del sistema una transformación de los valores numéricos en valores de lógica difusa y una posterior conversión de los valores difusos en valores numéricos en la salida del mismo.

Además es obvio que debe existir un motor de inferencia que realice las reglas siguiendo un procedimiento de inferencia borroso.

Desde el punto de vista de Control Inteligente los sistemas de lógica borrosa son sistemas que determinan las señal de salida o de control mediante razonamiento borroso a partir de los valores del conjunto de señales de entrada. La Figura (10.2) presenta el diagrama de bloques de un sistema basado en lógica borrosa.

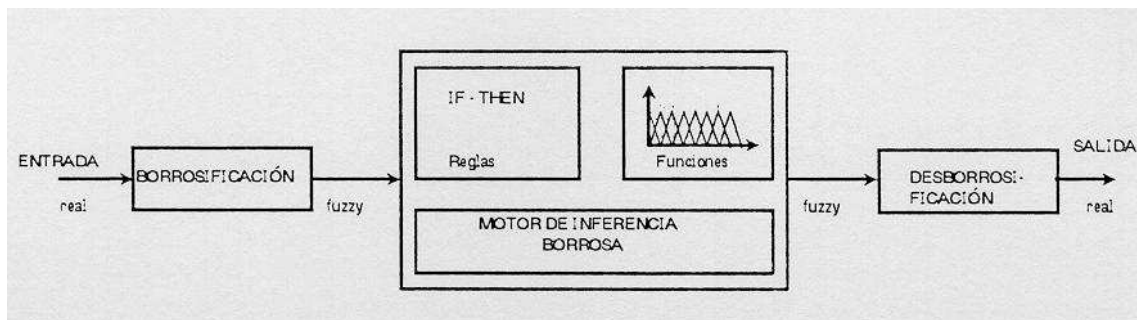


Figura (10.2) - Sistema basado en Lógica Borrosa.

La primera fase o bloque, borrosificación, asigna a las variables numéricas de entrada los grados de pertenencia a las diferentes clases o conjuntos borrosos. Por ejemplo, la variable de entrada temperatura puede ser, dependiendo del valor numérico, perteneciente con diferentes grados a las clases frío, templado y caliente.

La segunda fase aplica el conjunto o base de reglas tipo IF-THEN a las variables de entrada. Los resultados de las diferentes reglas se agrupan para obtener la salida borrosa. El grado de pertenencia correspondiente a una función lógica AND difusa de una regla se obtiene tomando el mínimo de los grados de pertenencia de los antecedentes de dicha regla. El grado de pertenencia de la combinación de los mismos tipos de resultado de diferentes reglas se obtiene eligiendo el máximo grado de pertenencia de los mismos.

Por último, la tercera fase, Desborrosificación, convierte el conjunto borroso de salida en un valor numérico necesario para el mecanismo de control. Por ello se utilizan diferentes métodos destacando entre ellos el método de máxima pertenencia y el método de centro de gravedad.

Ejemplo de Control con Lógica Borrosa

Problema del Péndulo Invertido

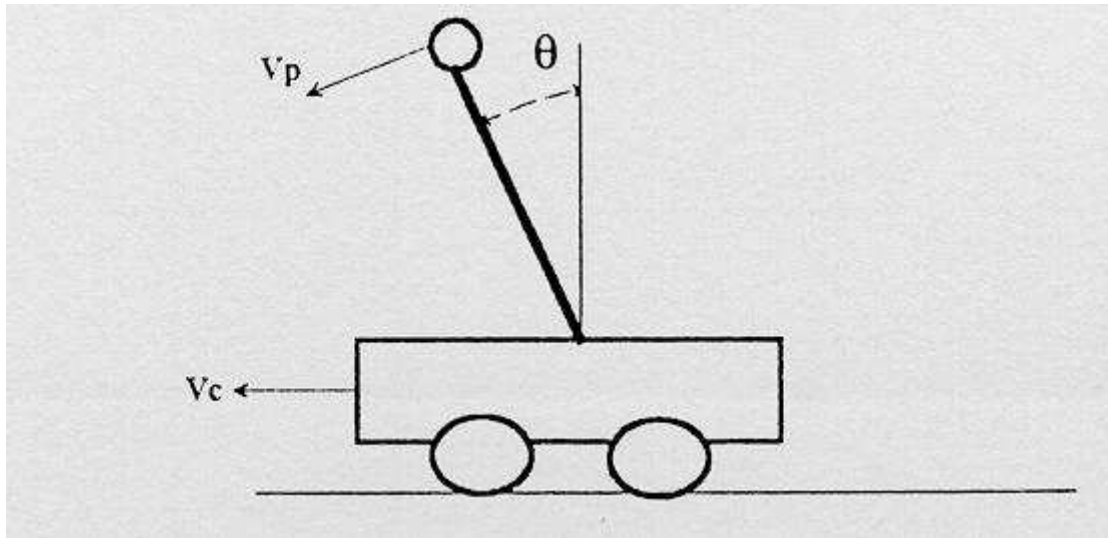


Figura (10.3) - Sistema de Péndulo Invertido.

- Variables de Entrada: Ángulo del Péndulo θ
 Velocidad de caída V_p
- Variable de Salida: Velocidad del carro V_c
- Conjuntos Borrosos: NL, NM, NS, ZR, PS, PM, PL
 Función Pertenencia: Triangular

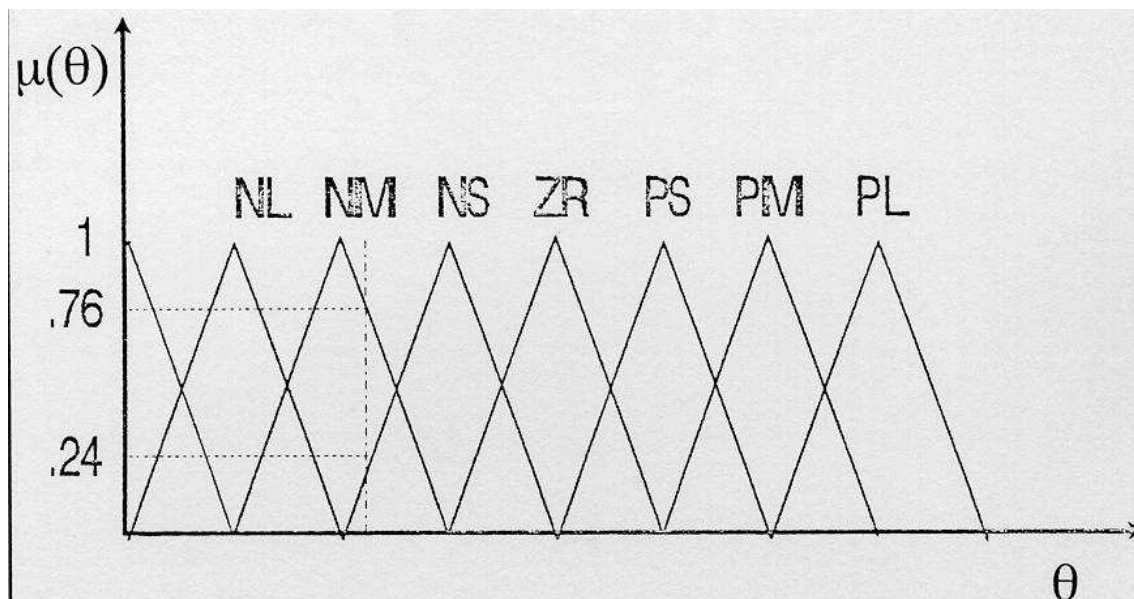


Figura (10.4) - Borrosificación de la variable de entrada θ .

- Conjuntos de Reglas:

1° IF (θ es NM **AND** V_p es NL) THEN V_c es NM
 2° IF (θ es NM **AND** V_p es NS) THEN V_c es NS

 6° IF θ es ZR THEN V_c es ZR

Regla	?	Grado	V_p	Grado	V_c	Grado
1°	NM	0,76	& NL	0,30	NM	0,30
2°	NM	0,76	& NS	0,70	NS	0,70
3°	NS	0,24	& NL	0,30	NM	0,24
4°	NS	0,24	& NS	0,70	NS	0,24

Salida Difusa: combinación de reglas que especifican la misma acción.

1° y 3° \Rightarrow NM 0,30
 2° y 4° \Rightarrow NS 0,70

- Desborrosificación

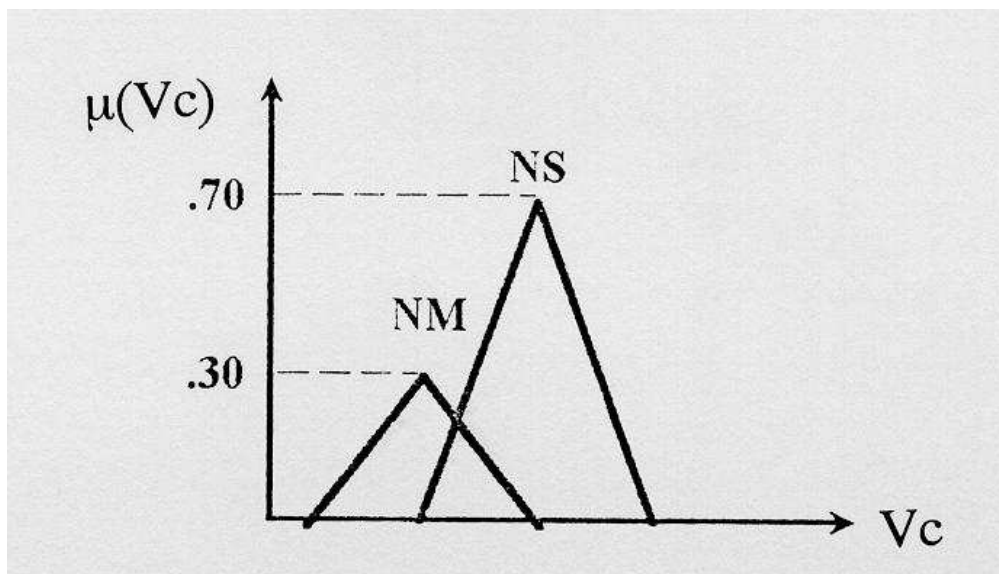


Figura (10.5) -Desborrosificación de la variable de salida V_c .

métodos alternativos: - centro de gravedad
 - máxima pertenencia

10.3.- SISTEMAS NEURO- DIFUSOS

Este apartado presenta una de las características más notables de las llamadas metodologías del Control Inteligente, la colaboración interdisciplinaria de las diferentes tecnologías que las constituyen. En concreto, la colaboración entre las Redes Neuronales Artificiales y la Lógica Difusa puede ser utilizado para mejorar o resolver algunas de las limitaciones que presentan cada una de ellas; estos nuevos sistemas híbridos, llamados sistemas neuro-difusos, desarrollan las propiedades y ventajas propias de cada tecnología en beneficio de la otra tecnología complementaria, obteniendo una mejora importante en el comportamiento global del sistema.

En los sistemas de control basados en lógica difusa no es imprescindible partir de un modelo matemático del mismo, sino que como se ha indicado anteriormente un conocimiento básico de cómo funciona el sistema puede ser suficiente. No obstante la tarea de explicitar este conocimiento mediante un conjunto de reglas tipo IF-THEN no siempre resulta sencilla, ya que la definición precisa de las funciones de pertenencia y de las relaciones entre las variables del sistema conllevan cierta dificultad.

Las Redes Neuronales Artificiales por su propia naturaleza y comportamiento son incapaces de expresar de forma explícita el conocimiento adquirido; pero por el contrario presentan la capacidad de aprender y relacionar las variables del sistema a partir de datos obtenidos en experiencias anteriores.

La Figura (10.6) representa un prototipo de sistema de cooperación de ambas tecnologías. Tiene como propósito controlar una aplicación utilizando la capacidad de aprendizaje de las Redes Neuronales Artificiales y la comprensión clara de los modelos lógicos difusos.

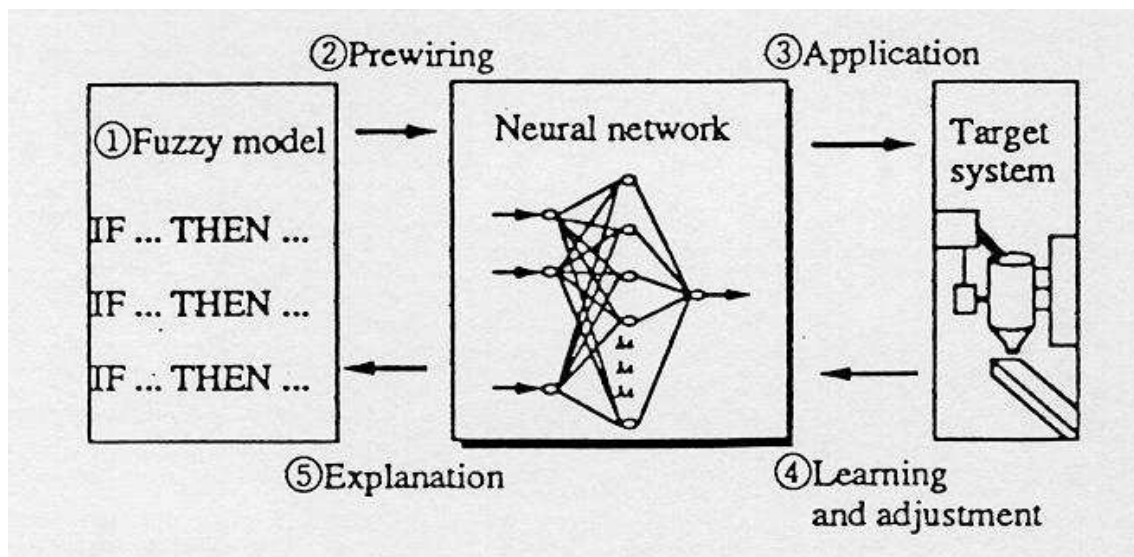


Figura (10.6) - Sistema Neuro-Difuso.

La cooperación de ambas tecnologías se realiza en los siguientes pasos:

a) A partir del conocimiento que tienen los expertos del sistema a controlar se infieren las funciones de pertenencia y las reglas borrosas que definen el modelo borroso del sistema objeto de estudio.

b) Se establecen las conexiones y el valor inicial de los pesos de la red neuronal de acuerdo con el modelo borroso.

c) Se aplica dicha Red Neuronal al sistema objeto de estudio.

d) La red neuronal es entrenada con los datos obtenidos para mejorar su precisión.

e) Después del entrenamiento, las conexiones y los pesos de la red neuronal son interpretados como funciones de pertenencia y reglas difusas. De esta manera queda explicitado el conocimiento adquirido por la red neuronal y el modelo borroso del sistema representa con mayor precisión al sistema real.

Las principales aplicaciones de los cada vez más numerosos sistemas híbridos neuro-difusos se pueden clasificar en los siguientes grupos:

- Mejorar la precisión del Modelo Difuso a partir del aprendizaje de las redes neuronales artificiales.
- Interpretar el conocimiento adquirido por la red neuronal artificial en el formato del modelo difuso.
- Extracción automática de reglas difusas utilizando arquitecturas especiales de redes neuronales artificiales.

BIBLIOGRAFÍA

Libros complementarios del curso

Libros de interés y consultados para la elaboración del curso

BIBLIOGRAFÍA

Libros complementarios del curso

1	<p>Título: Redes de Neuronas Artificiales. Un enfoque Práctico</p> <p>Autor: Pedro Isasi Viñuela , Inés M. Galván León, 2003</p> <p>Editorial: Pearson Prentice Hall</p> <p>ISBN: 84-205-4025-0</p>
2	<p>Título: El cerebro nos engaña</p> <p>Autor: Francisco J. Rubia, 2000</p> <p>Editorial: temas de hoy</p> <p>ISBN: 84-8460-045-9</p>
3	<p>Título: Redes Neuronales y Sistemas Borrosos</p> <p>Autores: Bonifacio Martín del Brío, Alfredo Sanz Molina, 2001</p> <p>Editorial: RA-MA, 2ª Edición 2001, 84-7897-466-0</p>
4	<p>Título: Cerebro y emociones.El ordenador emocional</p> <p>Autor:José A. Jáuregui, 1998</p> <p>Editorial: Maeva</p> <p>ISBN: 84-86478-80-4</p>
5	<p>Título: Redes Neuronales Artificiales y sus Aplicaciones</p> <p>Redes Neuronales Artificiales y sus Aplicaciones</p> <p>Xabier Basogain Olabe</p> <p>Formato Impreso: Publicaciones de la Escuela de Ingenieros, 1998</p> <p>Formato html: Campus Virtual-Material Docente-Curso RNA</p>

Libros de interés y consultados para la elaboración del curso

Nota: la mayoría de las figuras provienen de esta serie de libros.

Aleksander, Igor
Neural Computing Architectures
MIT Press, Cambridge, MA, 1989

Anderson, J.A. & Rosenfeld, E., eds.
Neurocomputing
MIT Press, Cambridge, MA, 1989

Carrascosa, J.L.
Quimeras del Conocimiento. Mitos y Realidades de la Inteligencia Artificial
Ediciones Fundesco, 1992

Dayhoff, J.
Neural Networks Architectures: An Introduction
Van Nostrand Reinhold, New York, 1990

Diederich, J. ed.
Artificial Neural Networks: Concept Learning
Computer Society Press, Los Alamitos, CA, 1990

Fogelman-Soulie, F. ed.
Automata Networks in Computer Science
Princeton Univ. Press, Princeton, NJ, 1987

Grosberg, Stephen
Adaptive Pattern Classification and Universal Recording: I. Parallel Development and coding of Neural Feature Detectors
Biological Cybernetics, Volume 23, pp. 121-134, 1976

Grosberg, Stephen
Studies of Mind and Brain
Reidel, Dordrecht, Holland, 1982

Hebb, D. O.
Organization of behavior
Science Editions, New York, 1949

Hecht-Nielsen, Robert
Counter-Propagation Networks

IEEE First International Conference on Neural Networks, Volume II, pp 19-32,
1987

Hecht-Nielsen, Robert

Neurocomputing

Addison Wesley, Reading, MA, 1990

Hinton, G. & Anderson, J.A. eds.

Parallel Models of Associative Memory (Updated Edition)

Lawrence Erlbaum Associates, Hillsdale, NJ, 1989

Ikerlan

Sistemas de Control basados en Lógica Borrosa Fuzzy Logic

Ref. 600 88

Kohonen, T.

Self-Organization and Associative Memory

Springer-Verlag, New York, 1988

Kosko, B.

Bi-directional associative memories

IEEE Transactions on Systems, Man and Cybernetics 18(1), pp 49-60, 1987

Neural Networks and Fuzzy Systems

Prentice-Hall, 1992

Manual de NeuralWorks

NeuralWare Inc. Technical Publications, Pittsburgh, PA, 1990

Mare, Harston & Pap

Handbook of Neural Computing Applications

Academic Press, San Diego, 1990

McClelland & Rumelhart

Explorations in Parallel Distributed Processing

MIT Press Cambridge, MA, 1988

McClelland & Rumelhart

Parallel Distributed Processing, Vol 2

MIT Press Cambridge, MA, 1986

Mead, Carver

Analog VLSI and Neural Systems

Addison Wesley, Reading, MA, 1989

Minsky, M. L. & Papert, S.A.

Perceptrons (Expanded Edition)

MIT Press Cambridge, MA, 1988

Rumelhart & McClelland eds

Parallel Distributed Processing, Vol 1

MIT Press Cambridge, MA, 1986

Stephen, J.

Neural Network Design and The Complexity of Learning

MIT Press Cambridge, MA, 1990

Simpson, Patrick K.

Artificial Neural Systems

Pergamon Press, New York, 1990

Soucek, B. & Soucek, M.

Neural and Massively Parallel Computers

Wiley-Interscience, New York, 1988

Vemuri, V. eds

Artificial Neural Networks: Theoretical Concepts

Computer Society Press, Los Alamitos, CA, 1990

Waltz, D. & Feldman, J.A. eds.

Connectionist Models and their Implications

Abex, Norwood, NJ, 1988

Wasserman, P.D.

Neural Computing

Van Nostrand-Reinhold, New York, 1989

JOURNALS

IEEE Transactions on Neural Networks

Neural Computation

MIT Press Cambridge, MA

Neural Networks

Pergamon Press, New York