UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POSGRADO

NUEVO ENFOQUE EN EL DISEÑO Y ENTRENAMIENTO DE REDES NEURONALES PARA LA CLASIFICACION

POR

M.C. Francisco Román Angel Bello Acosta

TESIS

CON OPCION AL GRADO DE
DOCTOR EN INGENIERIA
CON ESPECIALIDAD EN INGENIERIA DE SISTEMAS

SAN NICOLAS DE LOS GARZA, N. L. ENERO DE 2001

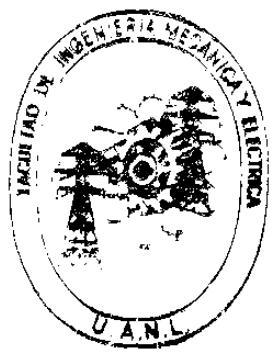
F.R.A.B.A.

NUEVO ENFOQUE EN EL DISEÑO Y ENTRENAMIENTO DE REDES NEURONALES PARA LA CLASIFICACION

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POSGRADO



NUEVO ENFOQUE EN EL DISEÑO Y ENTRENAMIENTO DE REDES NEURONALES PARA LA CLASIFICACION

POR

M.C. Francisco Román Angel Bello Acosta

TESIS

CON OPCION AL GRADO DE

DOCTOR EN INGENIERIA

CON ESPECIALIDAD EN INGENIERIA DE SISTEMAS

SAN NICOLAS DE LOS GARZA, N. L. ENERO DE 2001

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POSGRADO

Los miembros del comité de tesis recomendamos que la tesis **"Nuevo enfoque en el diseño y entrenamiento de redes neuronales para la clasificación"** realizada por el M.C. Francisco Román Angel Bello Acosta sea aceptada para su defensa con opción al grado de Doctor en Ingeniería con especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Asesor
Dr. José Luis Martínez Flores

Revisor
Dr. Oscar Leonel Chacón Mondragón

Revisor
Dr. Rafael Colás Ortiz

Revisor
Dr. Igor Litvinchev

Revisor
Dr. Noé García Sánchez

Vo. Bo.
M.C. Roberto Villarreal Garza
División de Estudios de Posgrado

San Nicolás de los Garza; Nuevo León. Enero de 2001

AGRADECIMIENTOS

Deseo expresar mi sincero agradecimiento al Dr. José Luis Martínez Flores, asesor de esta investigación, por su valiosa ayuda y orientación para la culminación de este trabajo.

Agradezco a los Doctores Rafael Colás Ortiz, Igor Litvinchev y Noé García Sánchez, revisores de esta tesis, por sus acertadas recomendaciones y especialmente al Dr. Oscar Leonel Chacón Mondragón por su paciente revisión y el interés que puso en este trabajo y en mi persona.

Por último, a las autoridades de la Universidad Autónoma de Nuevo León, al Programa de Apoyo a la Investigación Científica y Tecnológica, a la Facultad de Ingeniería Mecánica y Eléctrica y muy particularmente a su División de Estudios de Posgrado, muchas gracias por el apoyo recibido a lo largo de mis estudios doctorales en esta institución.

RESUMEN

Los modelos de redes neuronales artificiales se están aplicando con gran éxito en la solución de una amplia variedad de problemas complejos, tales como, el reconocimiento de formas y de voz, el control de robot, el filtrado de señales, etc. A pesar de esto, existen una serie de problemas no resueltos en el aprendizaje de las redes neuronales que limitan, en cierto modo, una mayor aplicación de este tipo de metodología. En este trabajo de investigación se tratan tres de los problemas fundamentales en el aprendizaje de redes neuronales artificiales de propagación hacia adelante: 1) Determinar la cantidad de neuronas en la capa oculta para que una red neuronal pueda clasificar un conjunto de patrones, 2) Obtener un algoritmo de entrenamiento, con menor cantidad de operaciones de punto flotante que las diferentes variantes del algoritmo de retropropagación del error y 3) Desarrollar un algoritmo de entrenamiento para redes neuronales clasificadoras de memoria entera.

Para dar solución al primer problema, a partir del análisis de los espacios de patrones y de pesos, se diseñó un algoritmo que determina la cantidad de neuronas en la capa oculta para clasificar un conjunto no contradictorio de patrones en dos clases. Este algoritmo determina, en cada iteración, un hiperplano que separa la mayor cantidad de patrones de una misma clase.

Para resolver el segundo problema, se diseñó un algoritmo para entrenar una red con funciones continuas de activación con un número de operaciones muy inferior al de las diferentes variantes del algoritmo de retropropagación del error. Este algoritmo toma como memoria inicial, la que se obtuvo de la red del problema anterior y es aplicable al filtrado de señales, donde el objetivo es minimizar una función de error cuadrático.

Por último, se estudia el problema de entrenamiento de una red neuronal de memoria entera para clasificar un conjunto de patrones en dos clases. Aquí, utilizando

los procedimientos se desarrollaron para el diseño de la red, se obtienen diferentes hiperplanos separadores y después de realizar un proceso de discretización de los coeficientes, se selecciona el hiperplano que separa la mayor cantidad de patrones de una misma clase. La importancia de este algoritmo de entrenamiento está en que las redes de memoria entera son muy económicas en sus diferentes formas de implementación.

INDICE

Capítulo **Página**

CAPITULO 1

INTRODUCCION

1.1 Antecedentes.

Debido a la complejidad de los problemas actuales de la ciencia y la técnica se están desarrollando de forma acelerada nuevos modelos matemáticos y métodos de las ciencias de la computación que se basan en el comportamiento de sistemas biológicos. Estos métodos son capaces de manejar incertidumbres que aparecen cuando nos enfrentamos a problemas reales y además pueden ofrecer soluciones robustas y de fácil implementación.

Los modelos de redes neuronales artificiales (RNA) están inspirados en el funcionamiento del cerebro humano y son capaces de dar solución a una extensa variedad de problemas complejos. Estos modelos están formados por una gran cantidad de unidades de procesamiento (neuronas), interconectadas entre sí y que operan masivamente en paralelo.

Las ideas primarias sobre los modelos de RNA aparecen en 1936 en los estudios de A. Turing, que es el primero en estudiar el cerebro como una forma de ver el mundo de la computación, aunque los primeros teóricos de la computación neuronal son considerados W. McCulloch y W. Pitts, quienes en 1943 desarrollan

una teoría sobre la forma de trabajar de las neuronas y modelan una red neuronal simple mediante circuitos eléctricos [107].

Los sucesos de mayor importancia en los primeros años de desarrollo de la teoría de las RNA se pueden resumir en los siguientes [104]:

- En 1958 F. Rosenblatt introduce un nuevo enfoque al problema de reconocimiento de patrones en sus trabajos sobre el *Perceptrón*, que es considerada como la red neuronal más antigua.
- En 1960 B. Widrow y M. Hoff introducen el algoritmo de los mínimos cuadrados y lo usan para formular *Adaline*, que es la primera red neuronal aplicada a un problema real. La diferencia fundamental entre el *Perceptrón* y *Adaline* radica en el procedimiento de aprendizaje.
- En 1967 S. Grossberg desarrolla una red neuronal, denominada *Avalancha*, que es utilizada para resolver actividades tales como reconocimiento continuo del habla y aprendizaje del movimiento de los brazos de un robot.

Esta primera etapa de desarrollo se caracteriza fundamentalmente por ver a las redes neuronales como unos modelos matemáticos "maravillosos" con los que se podía resolver una gran cantidad de problemas difíciles.

Por estos años comienzan a aparecer algunos problemas, tales como el problema de clasificación para el "or-exclusivo" y el problema de asignación de crédito, que no podían ser resueltos por los modelos existentes de RNA. Esto conlleva a la realización de un análisis más profundo sobre el funcionamiento de estos modelos, surgiendo así una serie de críticas que culminan en 1969 cuando M. Minsky y S. Papert [76] publican el libro "*Perceptrons*", donde ofrecen un análisis matemático detallado del *Perceptrón*, criticándolo fuertemente y considerando que la extensión a perceptrones multicapa era completamente estéril.

A partir de la aparición de este libro disminuye considerablemente la producción científica sobre redes neuronales, la mayoría de los investigadores se orientan hacia la Inteligencia Artificial y solo continúa trabajando en el tema un pequeño grupo, que por lo general elude utilizar el término de red neuronal.

En 1982 aparecen dos trabajos científicos que hacen resurgir el interés por las redes neuronales.

J. Hopfield [32] presenta un trabajo en el que describe con claridad y rigor matemático una red que es una variación del asociador lineal. Además, mostró cómo pueden trabajar tales redes y qué se puede hacer con ellas.

T. Kohonen [51] publica un artículo sobre un modelo de red neuronal con capacidad de formar mapas de características de manera similar a como ocurre en el cerebro humano, basándose en el principio de formación de mapas topológicos para establecer características comunes entre la información de entrada a la red. Una explicación más detallada sobre este trabajo puede encontrarse en la referencia [50].

A pesar de que estos trabajos no son sobre redes neuronales de propagación hacia adelante, sí permiten que se vuelva a retomar este tipo de modelo, comenzando así una nueva etapa de desarrollo de la Teoría de Redes Neuronales Artificiales.

Desde los primeros años de desarrollo se había notado, que si un problema no podía ser resuelto por una red neuronal de propagación hacia adelante, entonces era necesario agregar capas de neuronas ocultas, pero no se tenía fundamentación teórica sobre esto y además no existían algoritmos de entrenamiento.

En ese momento, los problemas fundamentales a los que se enfrentan los investigadores son:

1. Diseñar algoritmos de entrenamiento para redes multicapas.
2. Determinar la cantidad de capas ocultas de neuronas en la red.
3. Determinar la cantidad de neuronas en cada capa oculta.

El primero de estos problemas es resuelto en 1986, cuando aparece publicado un artículo por Rumelhart, Hinton y Williams [82], donde reportan un algoritmo de aprendizaje para redes multicapas, conocido como retropropagación del error (*backpropagation*). Además también fue obtenido de forma independiente por otros dos investigadores al mismo tiempo, Parker [80] y LeCun [67].

Es necesario señalar [103] que en 1974 Werbos, en su tesis doctoral realizada en la Universidad de Harvard, describe este algoritmo para modelos de redes más generales, donde las redes neuronales pueden ser consideradas como un caso

particular. Desafortunadamente este trabajo de Werbos fue desconocido para la comunidad científica por más de una década.

El primero en dar una solución al segundo problema es Cybenko, quien demostró rigurosamente que una sola capa de neuronas ocultas es suficiente para poder aproximar cualquier función continua soportada sobre un hipercubo unitario. Este resultado se conoce en el entorno de las RNA como el *Teorema de Aproximación Universal* y fue publicado, por primera vez en 1988, como un reporte técnico de la Universidad de Illinois y un año después en la referencia [15].

En 1989 son reportado otros dos trabajos sobre las capacidades de aproximación de las redes de propagación hacia adelante: Funahashi [25] y Hornik, Stinchcombe y White [33].

El teorema de aproximación universal es un teorema de existencia y es considerado como el resultado teórico de mayor importancia para este tipo de redes.

No obstante, este teorema deja abiertos dos problemas de gran importancia para la implementación de sistemas de RNA de propagación hacia adelante :

1. Cantidad de neuronas en la capa oculta
2. Acotamiento de los pesos de la red.

En el problema de mantener la propiedad de aproximador universal con una cantidad de neuronas limitada y pesos acotados se está trabajando actualmente en diferentes institutos de investigación a nivel mundial y aunque se han obtenido algunos resultados particulares [3,21,35,38,39,55,59,60,61,95], el problema general aún no está resuelto.

En 1992 se reportan dos resultados que garantizan que la superficie de error no presente mínimos locales. Estos resultados representan condiciones suficientes para que el algoritmo de entrenamiento sea convergente.

El primero es presentado por X. Yu [105] y plantea que son suficientes $T-1$ neuronas en la capa oculta, pero este valor resulta demasiado grande en la solución de problemas prácticos, por lo que esto representa una cota superior para el número de neuronas en la capa oculta.

El otro resultado es dado por Gori y Tesi [27], pero las condiciones que plantean son muy difíciles de verificar en la práctica. Otra forma de abordar este problema ha sido mediante algoritmos constructivos [10,11,12,23,71,80], pero solo se pueden aplicar a redes con funciones discretas de activación.

Para determinar el número de neuronas en la capa oculta, aún se sigue utilizando un procedimiento de ensayo y error [44,29], es decir, se construye una red de gran dimensión y se van eliminando neuronas de la capa oculta mientras se mantenga la convergencia del algoritmo de entrenamiento, o bien se construye una red pequeña y se le van agregando neuronas hasta que se alcance la convergencia.

En el problema del acotamiento de los pesos de la red también se esta trabajando actualmente y se han obtenido algunos resultados teóricos [35,38,53,59,62,97], pero los mayores esfuerzos están orientados hacia redes con memoria discreta y en particular las que utilizan memoria entera [22,46,71,72,77,84,86].

Cuando la memoria de la red está formada por valores enteros pequeños, estos modelos resultan muy atractivos, ya que sus diferentes formas de implementación son sumamente económicas.

El caso ideal es cuando los pesos toman solamente valores del conjunto $\{-1, 0, 1\}$, que son las llamadas redes libres de multiplicación [47], ya que para calcular la entrada neta a cada neurona, no es necesario realizar operaciones de multiplicación, solo sumas y restas. Además, en este caso los pesos de la red pueden ser almacenados mediante cadenas binarias de longitud 2.

Otro caso de gran interés es cuando los pesos están restringidos a tomar valores enteros del intervalo $[-3; 3]$, ya que se requieren solo de 3 bits para su almacenamiento y además proporcionan mayor diversificación para los hiperplanos separadores [48].

En estos dos tipos de redes los principales resultados han sido obtenidos por Khan [46] en su tesis doctoral presentada en 1996.

Otro problema que ha llamado la atención de los investigadores en RNA es que el algoritmo de retropropagación del error resulta demasiado costoso, desde el punto

de vista computacional, a pesar de todas las mejoras implementadas por diferentes autores [14,19,27,28,31,40,81,88,99,106].

El resultado más significativo en este sentido fue reportado por M. Biancini y M. Gori [7] en 1996. Ellos reducen el cálculo del gradiente del orden $O(M^2.T)$ al orden $O(M.T)$, donde M es el total de pesos de la red y T es la cantidad de patrones en el conjunto de aprendizaje.

1.2 Justificación del trabajo de investigación.

Debido a sus fundamentos, los modelos de RNA presentan varias características semejantes a las del cerebro. Esos modelos son capaces de aprender de la experiencia, de generalizar de casos anteriores a casos nuevos, de extraer características esenciales a partir de datos que contienen información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas, por lo que este tipo de metodología se está aplicando en múltiples áreas.

A pesar de que los modelos de RNA se están aplicando con gran éxito en la solución de problemas reales de alta complejidad, todavía existen una serie de problemas teóricos y prácticos no resueltos.

Los tres problemas fundamentales en el aprendizaje de redes neuronales de propagación hacia adelante son:

1. Diseño de la red neuronal.
2. Desarrollo de algoritmos eficientes de entrenamiento desde el punto de vista computacional.
3. Diseño y entrenamiento de redes neuronales de memoria entera.

El problema del diseño de la red neuronal consiste en determinar la cantidad de neuronas en la capa oculta para que el algoritmo de entrenamiento sea convergente, ya que la cantidad de neuronas en las capas de entrada y de salida es fija para cada problema particular.

En el caso de redes neuronales para la clasificación, este problema está estrechamente relacionado con el problema de determinar el mayor subconjunto

linealmente separable de un conjunto de entrenamiento, el cual es de la clase NP-completo [90].

El segundo problema está relacionado con las deficiencias del algoritmo de retropropagación del error [17,18,3052,54,65,66,69,83,94], las cuales son :

- La arquitectura de la red neuronal es determinada por ensayo y error.
- Alto número de operaciones, tanto en su fase hacia adelante en el cálculo de las salidas, como en su fase hacia atrás en la modificación de los parámetros de la red.
- La cantidad de ciclos de entrenamiento depende de la memoria inicial, la cual se selecciona de forma aleatoria.

La importancia del problema de diseño y entrenamiento de redes neuronales de memoria entera se debe al bajo costo de producción de este tipo de tecnología. Tanto el problema de diseño, como el de entrenamiento de la red cuando la memoria es entera están catalogados como NP-completos [46].

A partir del análisis de estos problemas definimos la orientación de nuestro trabajo de investigación hacia el desarrollo de métodos y algoritmos de entrenamiento para modelos de RNA de propagación hacia adelante que determinen la cantidad de neuronas en la capa oculta, que sean eficientes computacionalmente y que obtengan una memoria con valores enteros pequeños.

1.3 Objetivos de la investigación.

Del análisis de los problemas fundamentales existentes en el aprendizaje de redes neuronales para la clasificación, planteamos los siguientes objetivos de investigación.

1. Desarrollar un algoritmo para determinar la cantidad de neuronas en la capa oculta de una red neuronal de propagación hacia adelante.
2. Diseñar una nueva forma de entrenar una red con menor número de operaciones que la mejor variante del algoritmo de aprendizaje existente (*backpropagation*).

3. Diseñar un método de entrenamiento para redes neuronales de propagación hacia adelante con memoria entera.

1.4 Metodología de investigación.

Para llevar a cabo el trabajo de investigación se propuso la siguiente metodología a seguir :

1. Investigación bibliográfica sobre los problemas del aprendizaje en redes neuronales de propagación hacia adelante.
2. Análisis de los problemas de clasificación de patrones, de separabilidad lineal y de transformación de conjuntos de patrones en conjuntos linealmente separables. Esto permitirá desarrollar un método que determine el número de neuronas en la capa oculta de redes neuronales de propagación hacia adelante.
3. Análisis del problema de aproximación de funciones y su aplicación en el aprendizaje en redes neuronales de propagación hacia adelante.
4. Estudio de los fundamentos teóricos y análisis riguroso del problema de aprendizaje en redes neuronales de propagación hacia adelante, así como de su algoritmo de entrenamiento. Esto permitirá proponer un método más eficiente en relación con la cantidad de operaciones a realizar en el proceso de aprendizaje.
5. Análisis de la propiedad de aproximación universal en redes neuronales con memoria entera y de métodos de programación entera para diseñar un algoritmo de entrenamiento para redes con memoria entera.
6. Conclusiones de la investigación y recomendaciones para futuros trabajos en esta línea de investigación.

1.5 Estructura de la tesis.

El trabajo de tesis está compuesto por 6 capítulos. El capítulo 2 es de marco teórico y se da una descripción de los conceptos, métodos y algoritmos que se referencian en este trabajo de tesis.

En el capítulo 3 se estudia el problema de diseño de redes neuronales para la clasificación. Aquí, a partir del estudio del problema de clasificación y reconocimiento de patrones y del análisis de la estructura de los espacios de patrones y de pesos, se desarrolla un algoritmo para determinar la cantidad de neuronas en la capa oculta.

En el capítulo 4 se analiza el problema de la eficiencia del algoritmo de entrenamiento y se propone una nueva forma de entrenar redes neuronales para la clasificación que toma como memoria inicial la que se obtuvo de la solución del problema de diseño de la red. El funcionamiento de este algoritmo se basa en explotar las propiedades de las funciones continuas de activación. Con esta nueva forma de entrenamiento se logra reducir significativamente la cantidad de operaciones del proceso de aprendizaje de la red, así como la cantidad de parámetros a modificar. Además, la memoria inicial no se toma de forma aleatoria.

El capítulo 5 está dedicado a los problemas de diseño y entrenamiento de redes neuronales de memoria entera para la clasificación. En este capítulo los pesos de la red estás restringidos a tomar valores enteros del intervalo [-3, 3] y los métodos propuestos son modificaciones de los algoritmos desarrollados en los capítulos 3 y 4.

El capítulo 6 es el de conclusiones y recomendaciones. Aquí se resaltan los principales logros y limitaciones de los resultados obtenidos en este trabajo de investigación, así como se dan una serie de recomendaciones para trabajos futuros, en relación con la generalización de los resultados obtenidos.

CAPITULO 2

MARCO TEORICO

2.1 Introducción.

Uno de los principales objetivos y preocupaciones de los científicos a lo largo de la historia ha sido conseguir diseñar y construir máquinas capaces de realizar procesos con cierta inteligencia. De los intentos realizados en este sentido se han llegado a definir las líneas fundamentales para la obtención de máquinas inteligentes.

A pesar de disponer de herramientas y lenguajes de programación diseñados expresamente para el desarrollo de máquinas inteligentes, existe un problema de fondo que limita enormemente los resultados que se puedan obtener y es que estas máquinas se implementan sobre procesadores basados en una filosofía de funcionamiento, que se apoya en una descripción secuencial del proceso de tratamiento de la información. El desarrollo de estos procesadores no deja de seguir la línea de que, una máquina puramente mecánica es capaz de realizar tareas mecánicas, de forma increíblemente rápida, pero es incapaz de obtener resultados aceptables

cuando se trata de tareas sencillas para un ser humano como reconocimiento de formas, de voz, etc.

El desarrollo de la lógica formal permitió contar con una notación precisa para representar aseveraciones relacionadas con todo lo que existe en el mundo, así como sus relaciones mutuas.

Para crear sistemas inteligentes, la Inteligencia Artificial se esfuerza por elaborar programas que puedan describir un problema en notación lógica y encontrarle solución.

Actuar racionalmente implica actuar de manera que se logren los objetivos deseados, con base en cierto supuesto. Un agente es algo que es capaz de percibir y actuar. De acuerdo con este enfoque, la Inteligencia Artificial se considera como el estudio y construcción de agentes racionales.

Un agente es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores. El objetivo fundamental es diseñar y construir agentes racionales, es decir, agentes que logren un buen desempeño en su ambiente.

Uno de los objetivos fundamentales de los Sistemas de Redes Neuronales Artificiales es también el diseño y construcción de agentes racionales, pero lo hacen de una forma completamente diferente. Las redes neuronales se pueden enmarcar dentro de las denominadas redes de autoproceso, que están formadas por nodos procesadores de información de cuyas interacciones locales depende el comportamiento global del sistema.

2.2 Neuronas biológicas y sus modelos artificiales.

La teoría y la modelación de redes neuronales artificiales está inspirada en la estructura y funcionamiento del sistema nervioso.

La *neurona* o célula nerviosa es la unidad funcional básica de los tejidos del sistema nervioso, incluido el cerebro. Las neuronas están formadas por el cuerpo de célula o *soma*, donde se aloja el núcleo de la célula. Del cuerpo de la célula salen

ramificaciones conocidas como *dendritas*, y sale también una más larga denominada *axón*. Como se muestra en la figura 1.

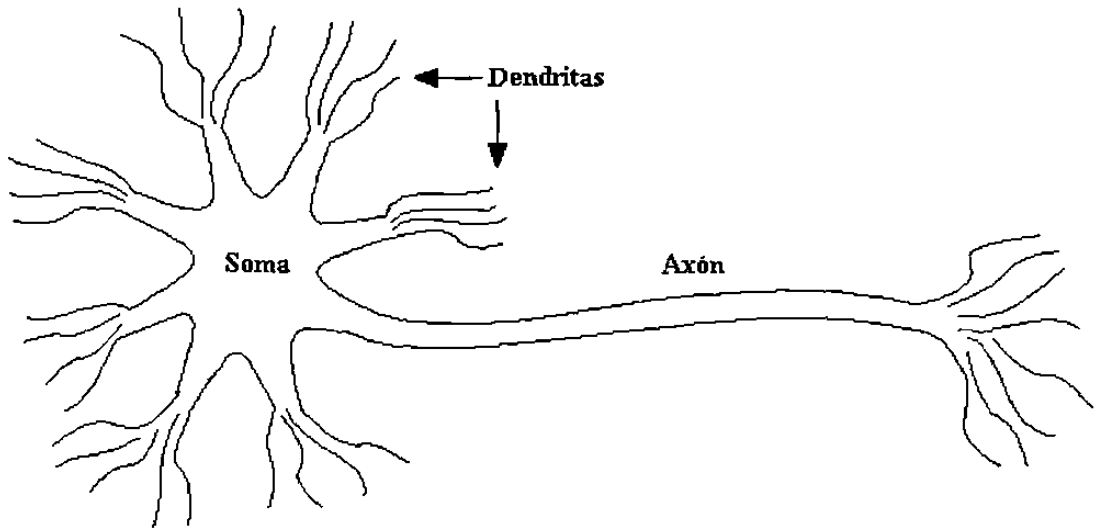


Figura 1 : Partes de la neurona.

Las dendritas se ramifican tejiendo una tupida red alrededor de la célula, mientras que el axón tiene una longitud, por lo general, de 1 cm (100 veces el diámetro del cuerpo de la célula) y en casos extremos llega a medir hasta un metro. Finalmente, el axón también se ramifica en filamentos, mediante los cuales establece conexión con las dendritas y cuerpos de otras células. Esta unión o conexión se le conoce como *sinapsis*.

Cada neurona establece sinapsis con una cantidad variable de otras neuronas, que oscila desde una docena hasta ciento de miles.

Las señales se propagan de neurona a neurona mediante una complicada reacción electroquímica. Las sinapsis liberan sustancias químicas transmisoras y entran a la dendrita con la cual se eleva o se reduce el potencial eléctrico del cuerpo de la célula. Una vez que el potencial eléctrico rebasa cierto valor de umbral, se envía al axón un impulso eléctrico o potencial de acción. El impulso se difunde a través de las ramas del axón y finalmente llega a la sinapsis y libera transmisores en los cuerpos de otras células.

Las sinapsis que aumentan el potencial se conocen como excitadoras, y las que lo disminuyen se denominan inhibidoras. La característica más importante de las conexiones sinápticas es que muestran plasticidad, es decir, alteraciones a largo plazo de la intensidad de las conexiones como respuesta al patrón de estimulación. Las neuronas establecen nuevas conexiones con otras neuronas, y en ocasiones con grupos completos de neuronas capaces de migrar de un sitio a otro.

Se considera que los mecanismos anteriores constituyen el fundamento del aprendizaje en el cerebro.

Este modelo del sistema nervioso parte de que las neuronas se comunican entre sí por medio de impulsos eléctricos y que forman una red neuronal que tiene una estructura compleja de interconexiones. La entrada a la red proviene de receptores sensitivos que están en contacto con el mundo exterior. Estos sensores envían estímulos en forma de impulsos eléctricos que llevan la información a la red de neuronas. Como resultado del procesamiento de la información en el sistema nervioso central, los efectores controlan y dan respuesta en forma de diversas acciones.

De esta forma este sistema está formado por los receptores, la red neuronal y los efectores en el control del organismo y sus acciones, como se muestra en la figura 2.

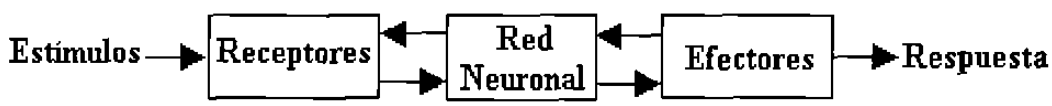


Figura 2: Flujo de información en el sistema nervioso.

Las redes neuronales artificiales son modelos que intentan reproducir el comportamiento del cerebro humano. Cada modelo neuronal consiste de un elemento de procesamiento con conexiones de entrada sinápticas y una salida simple.

En la figura 3 se muestra un modelo general de un elemento de procesamiento (PE). La i-ésima señal de entrada que recibe este elemento de procesamiento se representa por x_i. El flujo de señal de una neurona de entrada es considerado unidireccional y esta indicado por el sentido de la flecha. Cada conexión tiene asociada una magnitud llamada peso o intensidad de conexión y se denota por w_i.

Cada PE determina un valor de entrada neta basándose en todas las conexiones de entrada. Por lo general, se calcula el valor de entrada neta mediante la suma ponderada de las entradas por sus pesos correspondientes y se denota por *net* es decir,

$$net = \sum_{i=1}^{n} w_i x_i = W^t X \tag{2.1}$$

donde $W = \left[w_1, w_2, \cdots, w_n\right]^t$ es el vector de los pesos, y $X = \left[x_1, x_2, \cdots, x_n\right]^t$ es el vector de entradas.

Nótese, además, que el umbral se puede considerar como un peso más para una señal de entrada igual a -1 y que por comodidad se asume que es igual a x_n.

Una vez que la entrada neta ha sido calculada, se puede calcular el valor de salida aplicando la función de activación $o = f(net)$

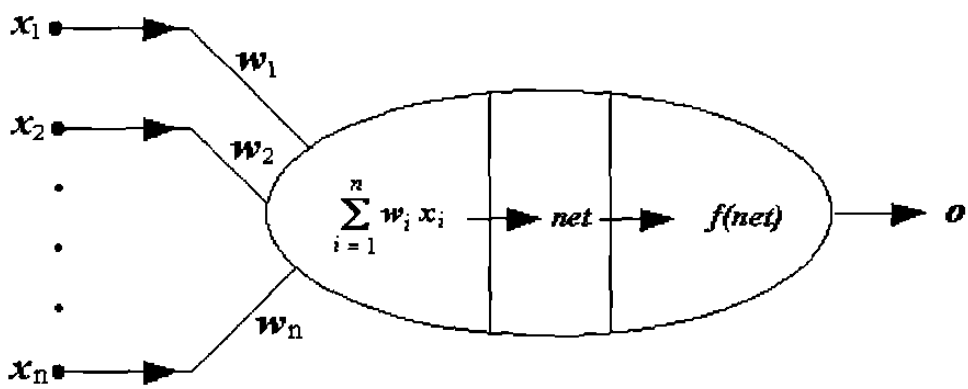


Figura 3 : Simbología para las neuronas artificiales

Las funciones de activación más usadas son

$$f(net) = \frac{2}{1 + e^{-\lambda net}} - 1 \quad , \tag{2.2}$$

$$f(net) = \begin{cases} 1, & net \geq 0 \\ -1, & net < 0 \end{cases} \tag{2.3}$$

Las funciones de activación (2.2) y (2.3) son conocidas como función continua bipolar (función sigmoide) y función binaria bipolar (función signo). Note que cuando $\lambda \to \pm\infty$, la función continua tiende a sgn(net).

Otras funciones que también se usan como funciones de activación son

$$f(net) = \frac{1}{1+e^{-\lambda net}} \tag{2.4}$$

$$f(net) = \begin{cases} 1, & net \geq 0 \\ 0, & net < 0 \end{cases} \tag{2.5}$$

que se denominan función continua unipolar y función binaria unipolar respectivamente.

La mayoría de los modelos neuronales emplean uno de estos tipos de funciones de activación

Dada una capa de m neuronas, sus valores de salida $o_1, o_2, \cdots, o_m$ se pueden agrupar en un vector de salida

$$O = [o_1, o_2, \cdots, o_m]' \tag{2.6}$$

donde o_i es la señal de salida de la neurona i.

El dominio del vector O está definido en un espacio m-dimensional por

$$(-1,\ 1)^m = \{O \in R^m,\ o_i \in (-1,\ 1)\},\ i = 1,2,\cdots,m$$

para la función bipolar continua, o por

$$(0,\ 1)^m = \{O \in R^m,\ o_i \in (0,\ 1)\},\ i = 1,2,\cdots,m$$

para la función unipolar continua. Es evidente que el dominio del vector O, en este caso, es el interior de un cubo m-dimensional.

Para valores de salida binarios o_i, el dominio de O en el espacio m-dimensional está dado por

$$\{-1,\ 1\}^m = \{O \in R^m,\ o_i \in \{-1,\ 1\}\},\ i = 1,2,\cdots,m$$

para la función bipolar discreta, o por

$$\{0,\ 1\}^m = \{O \in R^m,\ o_i \in \{0,\ 1\}\},\ i = 1,2,\cdots,m$$

para la función unipolar discreta.

En este caso el dominio del vector O está formado por los vértices de un cubo m-dimensional.

2.3 Clasificación y Reconocimiento de Patrones.

En la etapa de funcionamiento de redes neuronales, el proceso cálculo de una salida O para una entrada X dada, es conocido como *recordar* y el objetivo de esta fase es restaurar la información. *Recordar* corresponde a descifrar el contenido almacenado que puede haber sido introducido a la red previamente.

El término de *patrón* es utilizado para referirse a los elementos del conjunto de entradas que se le presentan a la red en la etapa de entrenamiento. Más aún, un patrón debe ser una descripción cuantitativa de un objeto, evento o fenómeno.

Supongamos que un conjunto de patrones puede ser almacenado en la red. Existen diferentes formas de asociación entre la entrada y la salida, como se muestra en la figura 4.

Autoasociación : A la red se le presenta una entrada y ella responde con miembro del conjunto almacenado que más se parece a esta entrada. Este proceso generalmente es usado para reconstruir una determinada información de entrada que se presenta incompleta o distorsionada.

Heteroasociación : es cuando la red almacena parejas de datos $(A_1, B_1);(A_2, B_2);\cdots;(A_n, B_n)$ de forma tal que cuando se presente una entrada ella la asocia al patrón almacenado mas parecido A_i y responde con la salida correspondiente B_i.

Clasificación : Es cuando el conjunto de patrones de entrada es dividido en clases o categorías. En este caso al presentar una entrada, la red debe dar como respuesta a qué clase pertenece. Generalmente las clases son expresadas por vectores de salida de valores discretos y se usan funciones de activación binarias.

Reconocimiento : En este caso también el conjunto de patrones de entrada es dividido en clases o categorías, pero la nueva entrada que se le presenta a la red no es exactamente igual a ningún miembro del conjunto de patrones de entrada, por lo que la red lo asocia a una clase donde se encuentre el patrón que más se le asemeje.

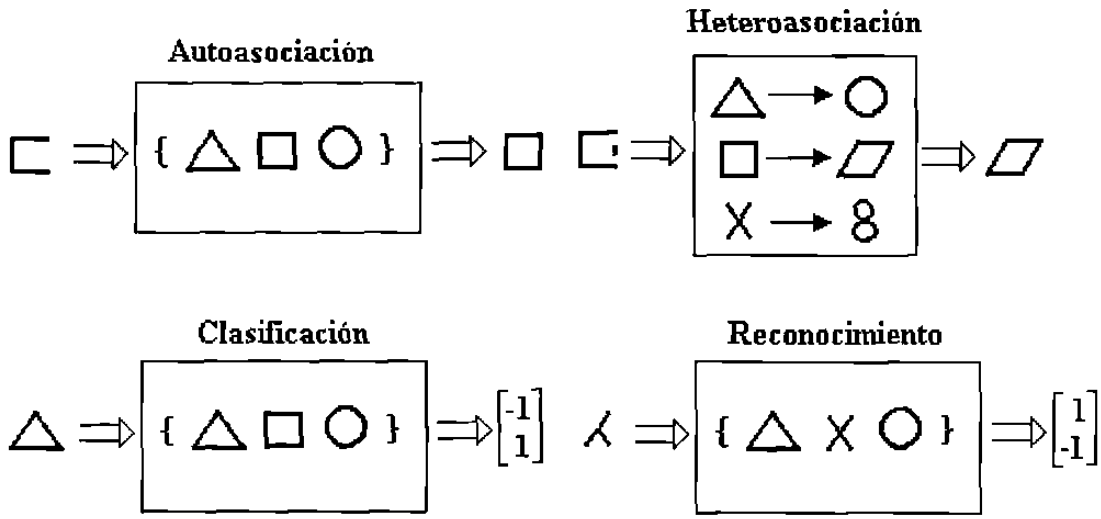


Figura 4 : Tipos de asociación.

La principal función de un sistema de clasificación es decidir a qué clase pertenece la entrada que se presente. Conceptualmente, el problema puede se descrito como una transformación de conjuntos o funciones desde el espacio de entrada al espacio de salida, que es llamado espacio de clasificación.

El objetivo de la clasificación de patrones es asignar un objeto físico, evento o fenómeno a una de las clases o categorías preestablecidas. El problema de clasificación de patrones puede ser considerado como uno de discriminación de datos de entrada dentro de una población de objetos, mediante la búsqueda de atributos invariantes entre los miembros de la población.

El estudio extensivo de procesos de clasificación ha conducido al desarrollo de modelos matemáticos abstractos que proporcionan las bases para el diseño de clasificadores [8,24,68,92,98].

En la figura 5 se muestra un diagrama de bloques para sistemas de reconocimiento y clasificación.

Los sistemas de clasificación contienen un traductor de entradas que provee al extractor de características de patrones de entrada. Generalmente, las entradas al extractor de características son conjuntos de vectores de datos que pertenecen a cierta categoría. El extractor de características reduce la dimensión de los datos, por lo que el espacio de características es de menor dimensión que el espacio de patrones. El

vector de características contiene solamente las características esenciales del vector de patrones que le permita mantener la probabilidad de hacer una clasificación correcta.

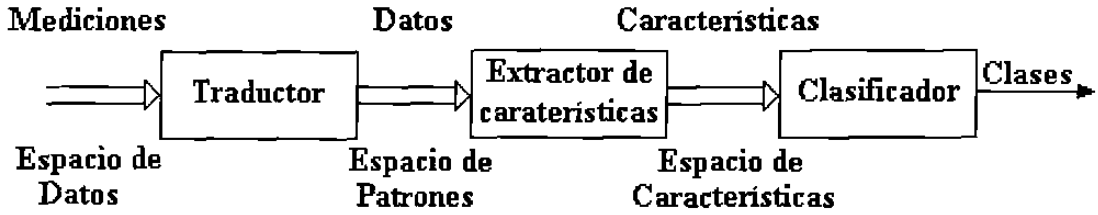


Figura 5 : Diagrama de bloques para sistemas de reconocimiento y clasificación.

La entrada es representada como un vector X y la clasificación a la salida del sistema es obtenida por un clasificador implementado por una función de decisión $i_0(X)$ que puede tomar uno de los valores discretos 1,2,...,R , donde la respuesta representa la categoría a la cual puede ser asignado el patrón, como se muestra en la figura 6. Es decir

$$i_0 = i_0(X) \tag{2.7}$$

donde $X = \left[x_1, x_2, \cdots, x_n\right]^t$

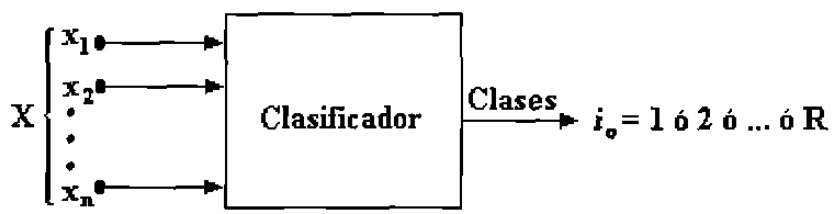


Figura 6 : Esquema de un clasificador multicategoría.

La función de clasificación (de decisión) de la ecuación (2.7) representa una transformación o aplicación de un vector n-dimensional X a una de las categorías $i_0(X)$.

La clasificación también puede ser descrita convencionalmente en forma geométrica. Un patrón puede ser representado por un punto en un espacio euclidiano n-dimensional $\mathbf{R}^n$, denominado espacio de patrones. Los puntos en este espacio corresponden a los elementos del conjunto de patrones que son vectores n-

dimensionales. Un clasificador de patrones aplica conjuntos de puntos del espacio $\mathbf{R}^n$ en el espacio de uno de los números $i_0(X) = 1, 2, \cdots, R$ como describe la función de decisión (2.7).

Las regiones denotadas por C_j son llamadas regiones de decisión y las fronteras que separan una región de las otras se denominan superficies de decisión. En un espacio $\mathbf{R}^n$ las superficies de decisión son hipersuperficies de n-1 dimensiones.

Durante la etapa de clasificación, para determinar la pertenencia a una categoría, el clasificador necesita basarse en la comparación de los cálculos para el patrón de entrada X de R funciones de discriminación $g_1(X), g_2(X), \cdots, g_R(X)$. Las funciones de discriminación toman valores escalares y un patrón pertenece a la i-ésima categoría si y solo si se cumple que

$$g_i(X) > g_j(X), \quad \forall i, j = 1, 2, \cdots, R; \quad i \neq j \tag{2.8}$$

Esto significa que dentro de la región C_j la i-ésima función de discriminación toma el mayor valor. Esta propiedad de la función de discriminación $g_i(X)$ de tomar el valor máximo para un patrón que pertenezca a la clase i es fundamental y es usado para seleccionar formas específicas de las funciones $g_i(X)$.

Las funciones de discriminación $g_i(X)$ y $g_j(X)$ para regiones de decisión contiguas C_i y C_j definen las superficies de decisión entre patrones de las clases i y j en el espacio $\mathbf{R}^n$.

El diagrama de bloques para un clasificador básico lo podemos adoptar como el que se muestra en la figura 7.

Para un patrón dado el i-ésimo discriminador calcula el valor de la función $g_i(X)$ que se denomina simplemente discriminante. El selector del máximo implementa la condición (2.8) y selecciona la mayor de todas las entradas produciendo una respuesta igual al número de categoría $i_0(X)$.

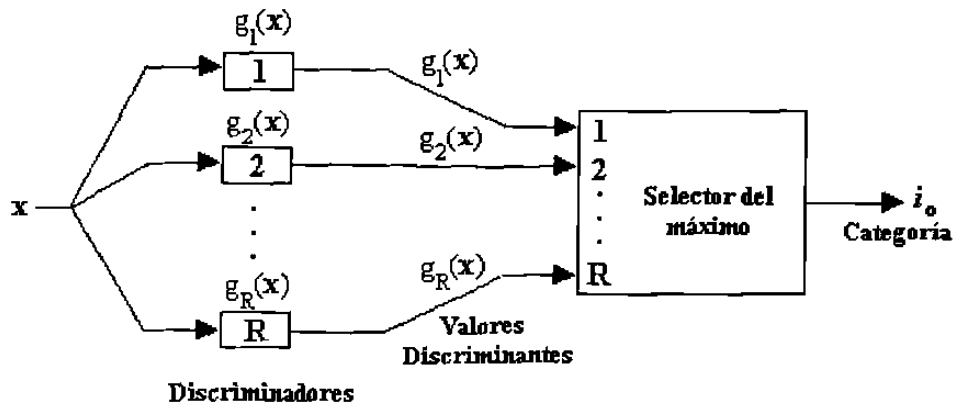


Figura 7: Diagrama de bloque de un clasificador básico

En el caso que $R = 2$, el clasificador es denominado dicotomizador y en este caso la condición (2.8) puede ser reducida a la inspección del signo de la siguiente función de discriminación

$$g(X) = g_1(X) - g_2(X) \tag{2.9}$$

Por lo que aquí la regla general (2.9) puede ser reescrita como

$$\begin{aligned} &g(X) > 0, \quad \text{si } X \in C_1 \\ &g(X) < 0, \quad \text{si } X \in C_2 \end{aligned} \tag{2.10}$$

La evaluación de esta condición es más fácil de implementar en la práctica que la condición del máximo. Para construir un dicotomizador simple puede ser usada una unidad lógica de umbral (TLU) simple como la que se muestra en la figura 8.

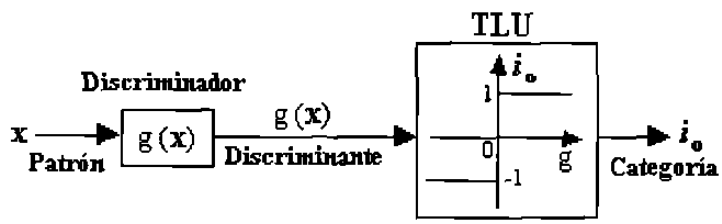


Figura 8 : Dicotomizador.

Una TLU puede ser considerada como una versión binaria de una neurona, el la cual los pesos son introducidos como en un perceptrón binario. Las respuestas 1, -1 de la TLU pueden ser interpretadas como indicaciones de las categorías 1 y 2 respectivamente. La TLU simple implementa la función signo definida como

$$i_0(X) = \text{sgn}(g(X)) = \begin{cases} 1, & g(X) > 0 \\ -1, & g(X) < 0 \end{cases}$$

El diseño de clasificadores se puede basar por completo en el cálculo de las fronteras de decisión que se derivan de los patrones y de su pertenencia a determinada clase.

Un clasificador eficiente, que tenga un diagrama de bloques como el que se mostró en la figura 7, puede ser descrito, en general, por funciones de discriminación que dependan de forma no lineal de las entradas $x_1, x_2, \cdots, x_n$. Pero, el uso de funciones de discriminación no lineales puede ser eludido mediante el diseño de clasificadores de propagación hacia adelante que sean multicapas.

En el caso de la clasificación lineal, la superficie de decisión es un hiperplano. En la figura 9 se muestra una función discriminante lineal en el caso bidimensional.

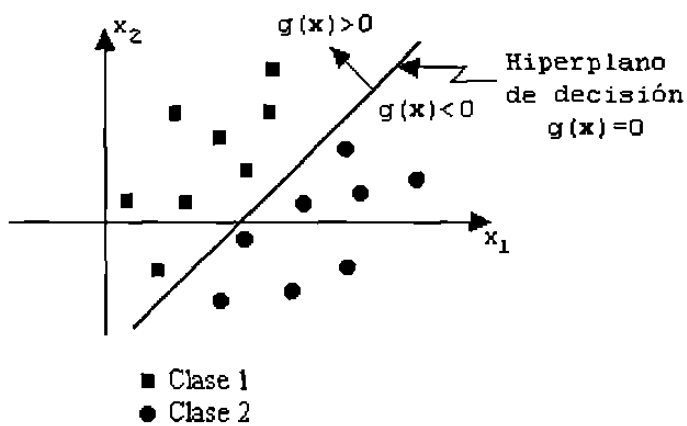


Figura 9 : Ilustración de una función discriminante lineal bidimensional.

La forma lineal de las funciones discriminantes también puede ser usada para la clasificación en más de dos categorías. En el caso de R categorías separables dos a dos existen a lo sumo $R(R\text{-}1)/2$ hiperplanos de decisión. Para un número de categorías elevado, algunas de las regiones de decisión C_i , C_j pueden no ser contiguas, por lo que se eliminan los hiperplanos de decisión.

El diagrama de bloques para un clasificador lineal se muestra en la figura 10 que se puede ver como un caso especial del diagrama de la figura 7.

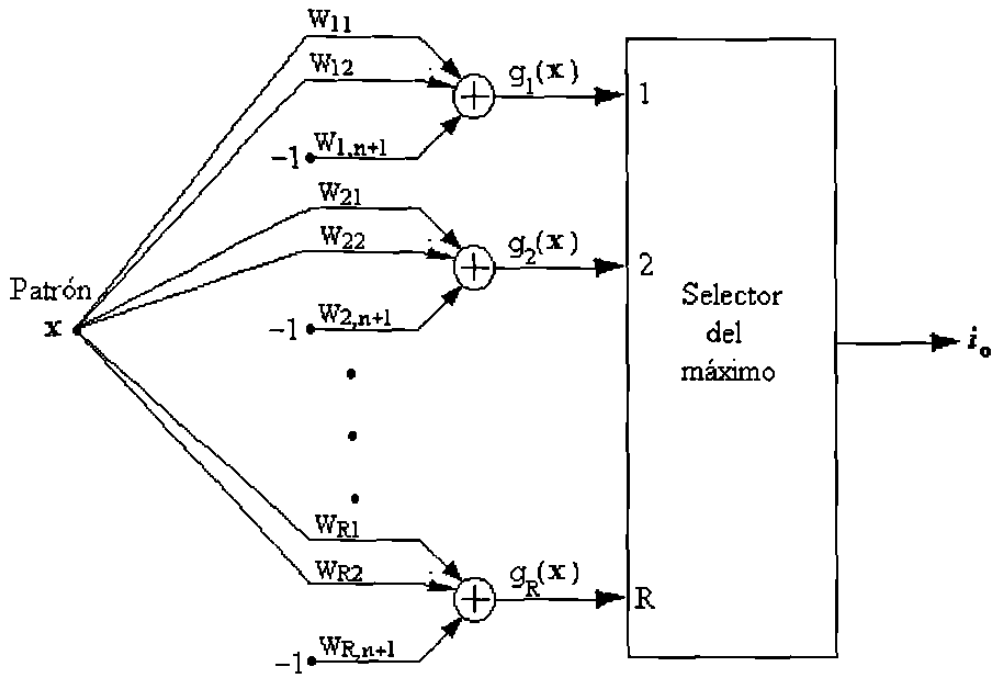


Figura 10 : Clasificador lineal.

El clasificador lineal está compuesta por R nodos de cálculo de producto escalar y un selector de máximo. Durante la clasificación, se calculan R funciones discriminantes $g_i(X)$ para el patrón de entrada X, estos valores se dan como entrada al selector del máximo que da como resultado el número de clase para el cual el determinante tuvo mayor valor.

Para introducir la noción de patrones linealmente separables, se asume que el conjunto de patrones $\mathbf{X}$ se puede dividir en los subconjuntos $\mathbf{X}_1$, $\mathbf{X}_2$,..., $\mathbf{X}_R$ respectivamente. Si un clasificador lineal puede clasificar los patrones de $\mathbf{X}_i$ como pertenecientes a la clase "i" para $i = 1, 2, \cdots, R$, entonces el conjunto de patrones $\mathbf{X}$ es linealmente separable.

Usando las propiedades de las funciones discriminantes lineales, la separabilidad lineal se puede formular más formalmente. Si existen R funciones lineales de X tales que

$$g_i(X) > g_j(X), \quad \forall X \in \mathbf{X}_i: i, j = 1, 2, \cdots, R\ ;\ i \neq j,$$

entonces los conjuntos de patrones $\mathbf{X}_i$ son linealmente separables.

2.4 Aprendizaje en redes neuronales de propagación hacia adelante.

2.4.1 Aprendizaje para clasificadores lineales.

Cuando se analiza el problema de clasificación de patrones, resulta interesante el estudio de aquellos clasificadores, que sus capacidades de decisión son generadas por patrones de entrenamiento mediante aprendizaje, entrenamiento o algoritmos iterativos.

La clasificación de un dato es aprendida gradualmente mediante la inspección repetida y clasificación de ejemplos.

Cuando el tipo de función discriminante ha sido seleccionado, el algoritmo de aprendizaje da como resultado la solución para los coeficiente, inicialmente desconocidos, de la función discriminante, que se obtiene a partir del conjunto de patrones de entrenamiento.

Para el estudio de clasificadores entrenables (adaptativos) se asume que :

1) El conjunto de patrones de entrenamiento es conocido, así como la clasificación de todos sus elementos, por lo que el entrenamiento es supervisado.
2) Las funciones discriminantes tienen una forma lineal y solo sus coeficientes son ajustados en el proceso de entrenamiento.

Bajo estas suposiciones, un clasificador entrenable puede ser implementado por el aprendizaje mediante ejemplos. El interés, por lo tanto, está enfocado hacia vectores de datos de entrada para los cuales se conoce su clasificación correcta, y a los que se denominan prototipos de clase.

El problema de clasificación consistirá entonces en determinar las superficies de decisión en un espacio n-dimensional a partir de la correcta clasificación de los prototipos y que permita con un grado de confianza realizar correctamente el reconocimiento y la clasificación de patrones desconocidos que no hayan sido usados en el entrenamiento. La única limitación que se tiene para que los patrones

desconocidos sean reconocidos es que tengan el mismo formato que se usó en los patrones de entrenamiento.

2.4.2 Aprendizaje como aproximación.

En general, el aprendizaje es un cambio permanente y relativo en el comportamiento basado en la experiencia. En redes neuronales, el aprendizaje es un proceso más directo y se puede entender como una relación causa-efecto que puede ser vista como una relación que transforma las entradas en las salidas para un conjunto de ejemplos de pares entrada-salida.

Una fundamentación clásica a este problema proviene de la Teoría de Aproximación. Esta teoría consiste en aproximar una función continua multivariable $h(X)$ por otra función $H(W, X)$, donde $X = \left[x_1, x_2, \cdots, x_n\right]^t$ es el vector de las entradas y $W = \left[w_1, w_2, \cdots, w_n\right]^t$ es el vector de los parámetros (pesos). En este sentido a las redes neuronales se les puede ver como un sistema que puede aprender la aproximación de relaciones.

Con este enfoque, el aprendizaje consiste en hallar el vector W que de la mejor aproximación posible de $h(X)$ para un conjunto de ejemplos de entrenamiento $\mathbf{X}$. La selección de la función $H(W, X)$ para representar a $h(X)$ es conocido como un problema de representación. Una vez que haya sido seleccionada $H(W, X)$, el algoritmo de aprendizaje de la red es aplicado para encontrar los parámetros óptimos W^*.

Una formulación más precisa del problema de aprendizaje puede ser establecida como los cálculos que involucran a W^* de forma tal que

$$\rho\left[\mathrm{H}(\mathrm{W}^*, \mathrm{X}), h(X)\right] \leq \rho\left[\mathrm{H}(\mathrm{W}, \mathrm{X}), h(X)\right]$$

donde $\rho[H(W, X)$, $h(X)]$ es una función distancia y representa una medida cualitativa de aproximación entre $H(W, X)$ y $h(X)$. Cuando el ajuste se determina sobre la base del cuadrado de las diferencias para un conjunto de ejemplos de entrenamiento $\mathbf{X}$, entonces la distancia tiene la forma de la suma de los errores cuadrados.

2.4.3 Regla general de aprendizaje en redes de propagación hacia adelante.

Una neurona es considerada como un elemento adaptativo. Sus pesos son modificables en dependencia de la señal de entrada que recibe, de su valor de salida y de la respuesta del supervisor. En el caso del aprendizaje no supervisado la modificación de los pesos se basa solamente en la señal de entrada y/o en los valores de salida.

En el proceso de aprendizaje se modifican las componentes w_{ij}, que están sobre las conexiones de la *j*-ésima entrada con la *i*-ésima neurona de cada vector de pesos W_i. En general las entradas pueden ser las salidas de otras neuronas o entradas externas.

En la figura 11 se muestra el entrenamiento de una neurona.

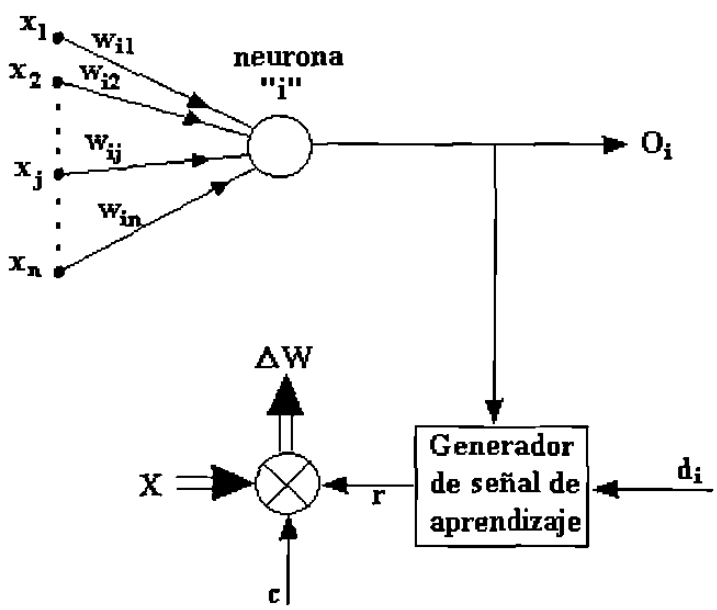


Figura 11: Esquema del aprendizaje supervisado en una neurona.

Para el estudio de redes neuronales se toma la siguiente regla general de aprendizaje :

" El vector de pesos $W_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^t$ se incrementa en proporción al producto de la entrada X y de la señal de aprendizaje *r* ".

Esta señal de aprendizaje es en general una función de W_i, de X y en algunos casos de la señal del supervisor d_i .

$$r = r\left(W_i, X_i, d_i\right)$$

El incremento del vector de pesos W_i es producido por el paso de aprendizaje en el instante de tiempo t de acuerdo a la regla de aprendizaje

$$\Delta W_i(t) = c\, r\left[W_i(t), X(t), d_i(t)\right] X(t) \tag{2.11}$$

donde c es un número positivo denominado constante de aprendizaje y determina la velocidad del aprendizaje.

De (2.11) se tiene que

$$W_i(t+1) = W_i(t) + c\, r\left[W_i(t), X(t), d_i(t)\right] X(t)$$

Para sistemas discretos en el tiempo se utilizan superíndices para especificar el paso de aprendizaje, es decir, para el k-ésimo paso se tiene

$$W_i^{k+1} = W_i^k + c\, r\left[W_i^k, X^k, d_i^k\right] X^k$$

por lo que el aprendizaje toma la forma de una secuencia discreta de modificaciones de los pesos.

2.4.4 Reglas clásicas de aprendizaje en redes de propagación hacia adelante.

Las reglas de aprendizaje más comunes [2,5,29,107] para las redes neuronales de propagación hacia adelante son:

- **Regla de aprendizaje del perceptrón.**

El análisis del diseño de clasificadores de patrones está basado en el cálculo de las superficies de decisión a partir de los prototipos o grupos de patrones muestrales. Los coeficientes de las funciones discriminantes lineales son los componentes del vector de pesos y pueden ser determinados a partir de la información que se tiene del conjunto de patrones sobre la pertenencia de sus elemento a clases o categorías.

Los vectores de patrones de muestrales $X_1, X_2, \cdots, X_T$ forman una secuencia de entrenamiento y se les presentan al clasificador conjuntamente con la respuesta de

salida correcta. El clasificador modifica los parámetros mediante un aprendizaje iterativo supervisado.

El aprendizaje del clasificador se basa en la experiencia lograda por la comparación de la respuesta correcta con la respuesta obtenida y su estructura usualmente es ajustada después de cada respuesta incorrecta. El ajuste se realiza sobre la base del valor del error generado.

El dicotomizador consta de n+1 pesos y de una TLU como elemento de decisión binaria. La entrada a este TLU es la suma ponderada de los componentes del vector de entrada. Ver figura 12.

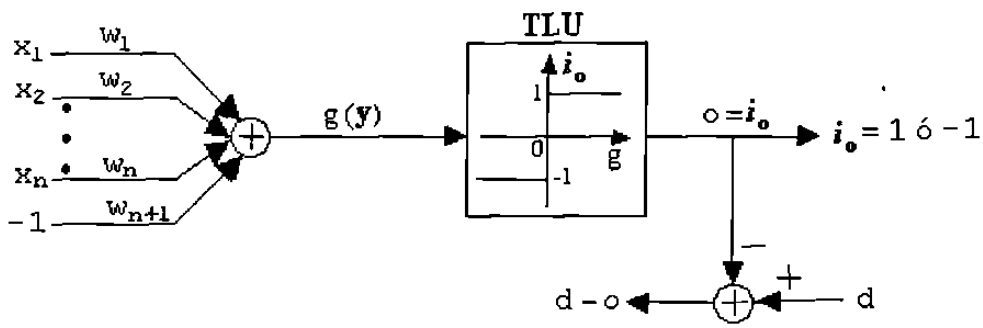


Figura 12 : Dicotomizador lineal con TLU.

Al presentarle al dicotomizador un patrón, la información sobre el error puede ser usada para adaptar los pesos.

Por lo que puede verse que esto es lo mismo que la regla de aprendizaje para el perceptrón discreto.

Para esta regla la señal de aprendizaje es la diferencia entre la salida deseada y la salida obtenida de la neurona al aplicarle una entrada *X*. Además, aquí el aprendizaje es supervisado. Ver figura 13.

En este caso la señal de aprendizaje tiene la forma

$$r = d_i - o_i$$

donde $o_i = \text{sgn}\left(W_i^t \cdot X\right)$ y d_i es la respuesta deseada.

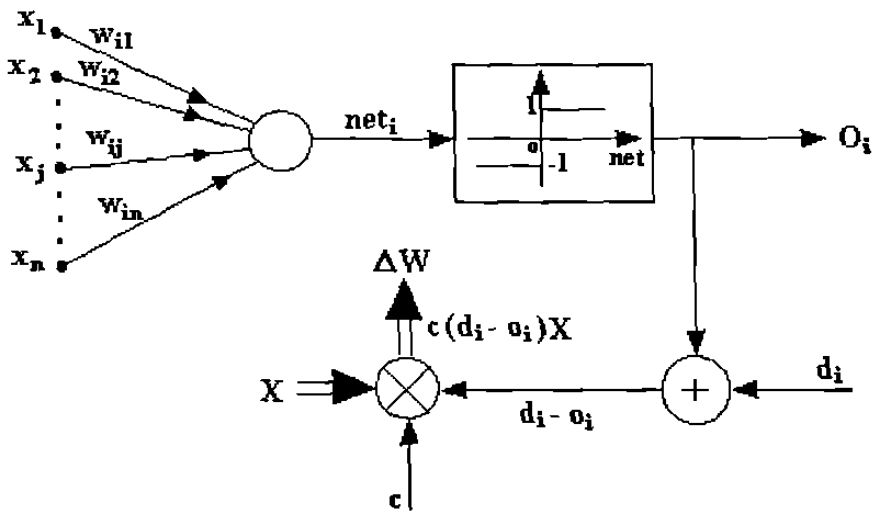


Figura 13 : Aprendizaje en un perceptrón discreto.

El ajuste de los pesos en esta regla de aprendizaje se realiza mediante la expresión

$$\Delta W_i = c\left[d_i - \text{sgn}(W_i^t X)\right] X$$

y cada elemento del vector de pesos se ajusta por

$$\Delta w_{ij} = c\left[d_i - \text{sgn}(W_i^t X)\right] x_j \qquad (2.12)$$

Esta regla es aplicable solo para neuronas de respuesta binaria y las relaciones (2.12) expresan la regla para el caso bipolar binario. Bajo esta regla los pesos son ajustados solo en caso que o_i sea incorrecta. Obviamente, como la respuesta deseada es 1 ó -1, el ajuste de los pesos se reduce a

$$\Delta W_i = \pm 2cX$$

donde el signo "+" es aplicable cuando $d_i = 1$ y $o_i = -1$ y el signo "-" cuando $d_i = -1$ y $o_i = 1$.

- **Regla delta de aprendizaje.**

Esta regla es aplicable solamente para las funciones de activación continuas bipolar y unipolar definidas anteriormente y para un aprendizaje supervisado. En esta regla la señal de aprendizaje se denomina delta y está definida por :

$$r = \left[d_i - f\left(W_i' X\right)\right] f'\left(W_i' X\right)$$

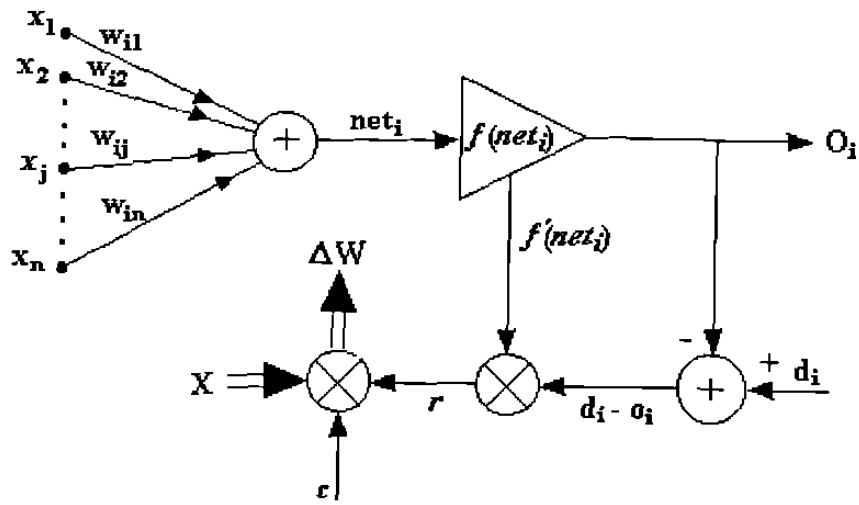


Figura 14 : *Aprendizaje en un Perceptrón continuo.*

El término $f'\left(W_i' X\right)$ es la derivada de la función de activación, calculada para $net = W_i' X$.

Esta regla de aprendizaje puede ser obtenida directamente de la condición del error mínimo cuadrático entre d_i y o_i.

Calculando el vector gradiente respecto a W_i del error cuadrático definido por

$$E = \frac{1}{2}(d_i - o_i)^2$$

o a su expresión equivalente

$$E = \frac{1}{2}\left[d_i - f\left(W_i' X\right)\right]^2$$

obtenemos el vector gradiente del error.

$$\nabla E = -\left[d_i - o_i\right] f'\left(W_i' X\right) X \qquad (2.13)$$

Los componentes del vector gradiente serán

$$\frac{\partial E}{\partial w_{ij}} = -\left[d_i - o_i\right] f'\left(W_i' X\right) \mathrm{x}_j$$

Como se requiere la minimización del error cuadrático, para realizar el cambio de los pesos, se toma la dirección negativa del gradiente, es decir,

$$\Delta W_i = -\eta \nabla E \tag{2.14}$$

donde η es una constante positiva.

Entonces de (2.13) y (2.14) se obtiene

$$\Delta W_i = \eta\left[d_i - o_i\right] f'\left(W_i^t X\right) X \tag{2.15}$$

y para cada componente del vector de pesos

$$\Delta w_{ij} = \eta\left[d_i - o_i\right] f'\left(W_i^t X\right) x_j \text{ , } j = 1,2,\cdots,n$$

Aplicando a este caso la formula general de aprendizaje definida por (2.11) se tiene que el ajuste de los pesos se realiza mediante

$$\Delta W_i = c\left[d_i - o_i\right] f'\left(W_i^t X\right) X$$

y comparándola con la expresión (2.15) se ve que son idénticas ya que c y η son constantes arbitrarias.

Para el entrenamiento de este tipo de redes es necesario inicializar el vector de los pesos.

Esta regla requiere del cálculo de $f'(net)$ en cada paso. Para este propósito se usa el resultado que $f'(net) = \frac{1}{2}\left(1 - o_i^2\right)$, que es válido para la función de activación continua bipolar.

Este resultado es de gran utilidad ya que expresa la pendiente de la función de activación en términos de la señal de salida de la neurona.

Como este método está basado en el movimiento del vector de pesos sobre la superficie de los pesos en la dirección negativa del gradiente de error, requiere de pequeños valores de c, el cual es seleccionado de forma empírica y ajustado en el proceso de aprendizaje.

- **Regla de aprendizaje del perceptrón discreto multicategoría.**

Para poder entrenar un clasificador multicategoría es necesario asumir que cada clase es linealmente separable respecto a las restantes clases, esto es equivalente a que existan R funciones discriminantes lineales tales que

$$g_i(X) > g_j(X), \quad \forall X \in \mathbf{X}_i: \; i, j = 1, 2, \cdots, R \; ; \; i \neq j$$

Por comodidad, el umbral de la i-ésima neurona es denotado por $w_{i,n+1}$ y es agregado a los vectores de pesos, también a los patrones de entrada se le agrega una componente $x_{i,n+1} = -1$, obteniéndose los vectores ampliados

$$\overline{W_i} = \left[w_{i1}, w_{i2}, \cdots, w_{in}, w_{i,n+1} \right]^t,$$

$$Y_i = \left[x_{i1}, x_{i2}, \cdots, x_{in}, -1 \right]^t$$

En el perceptrón discreto multicategoría, el selector del máximo puede ser sustituido por R unidades lógicas de umbral, que son más fáciles de implementar. En la figura 15 se muestra la estructura de un perceptrón discreto multicategoría.

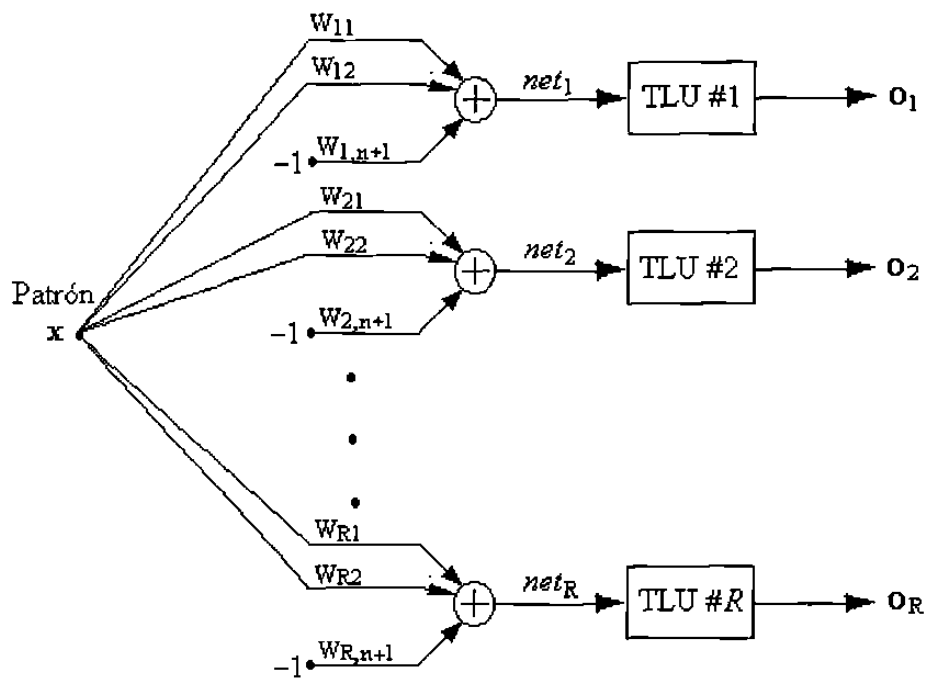


Figura15 : Estructura de un perceptrón discreto multicategoría.

En este caso la salida deseada correspondiente al patrón X_i será un vector $\mathbf{d}_i = [d_{i1}, d_{i2}, \cdots, d_{iR}]^t$, donde las componentes d_{ij} cumplen que

$$d_{ij} = \begin{cases} 1, \text{ si } \mathbf{X}_i \in C_k, j = k \\ -1, \text{ si } \mathbf{X}_i \in C_k, j \neq k \end{cases}$$

En esta regla de entrenamiento, el objetivo es que el vector de salida de la red $\mathbf{o}_i = [o_{i1}, o_{i2}, \cdots, o_{iR}]^t$, para cada patrón de entrenamiento X_i, coincida con el vector de salida deseada correspondiente, por tanto, la red se considera entrenada si se cumple que

$$o_{ij} = \begin{cases} 1, \text{ si } \mathbf{X}_i \in C_k, j = k \\ -1, \text{ si } \mathbf{X}_i \in C_k, j \neq k \end{cases},$$

lo que significa que $\mathbf{d}_i - \mathbf{o}_i = [0,0,\cdots,o]^t \in \mathbf{R}^R$.

Aquí, el problema de aprendizaje puede ser formulado como:

Dado un el conjunto de entrenamiento $\{(\mathbf{X}_1, d_1), (\mathbf{X}_2, d_2), \cdots, (\mathbf{X}_T, d_T)\}$, donde $\mathbf{X}_i = (x_{i1}, x_{i2}, \cdots, x_{in}) \in \mathbf{R}^n$, $i = 1, \cdots, T$ representan a los patrones de entrenamiento y $d_i = (d_{i1}, d_{i2}, \cdots, d_{iR})$, $i = 1, \cdots, T$ es el vector de salida deseada correspondiente al patrón X_i, siendo

$$d_{ij} = \begin{cases} 1, \text{ si } \mathbf{X}_i \in C_k, j = k \\ -1, \text{ si } \mathbf{X}_i \in C_k, j \neq k \end{cases},$$

determinar una matriz de pesos

$$\overline{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & w_{1,n+1} \\ w_{21} & w_{22} & \cdots & w_{2n} & w_{2,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{R1} & w_{R2} & \cdots & w_{Rn} & w_{R,n+1} \end{bmatrix}$$

de forma tal que $E = \frac{1}{2}\sum_{i=1}^{T}\sum_{j-1}^{R}\left(d_{ij} - o_{ij}\right)^2 = 0$,

donde $o_{ij} = \text{sgn}(net_{ij})$, $i = 1, \cdots, T$; $j = 1, \cdots, R$

Si el patrón está mal clasificado, la modificación de los pesos se realiza según la expresión

$$\overline{W}_j^{(k+1)} = \overline{W}_j^{(k)} + \frac{c}{2}\left(d_{ij} - o_{ij}\right)Y_i \quad , \; i = 1,\cdots,T \; ; \; j = 1,\cdots,R$$

- **Regla de aprendizaje del perceptrón continuo multicategoría.**

Este tipo de redes difiere del caso discreto en que utiliza funciones continuas de activación, por lo que las componentes del vector de salida, en el caso de la función bipolar, cumplen $|o_{ij}| < 1$. Ver figura 16.

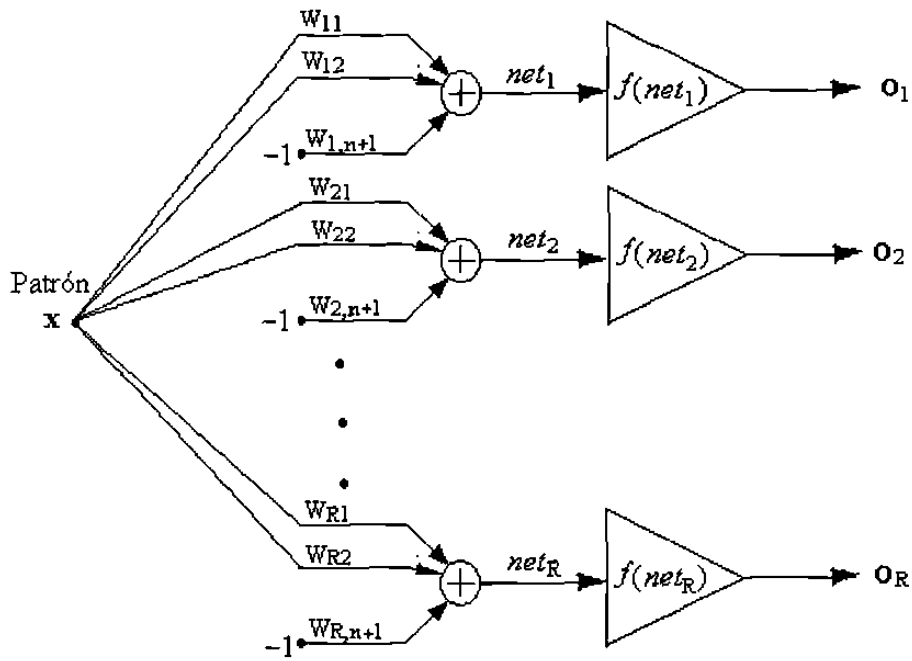


Figura 16 : Estructura de un perceptrón continuo multicategoría.

Aquí, la función de error tiene la misma forma que en el caso discreto, es decir

$$E = \frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{R}\left(d_{ij} - o_{ij}\right)^2 \; ,$$

pero $o_{ij} = \dfrac{2}{1+e^{-\lambda net_{ij}}} - 1 \quad , \; i = 1,\cdots,T \; ; \; j = 1,\cdots,R$, en el caso bipolar y

$o_{ij} = \dfrac{1}{1+e^{-\lambda net_{ij}}} \quad , \; i = 1,\cdots,T \; ; \; j = 1,\cdots,R$, en el caso unipolar.

En esta regla de aprendizaje, los pesos se modifican según el método del descenso acelerado, después que se le presenta un patrón a la red.

Aquí, el problema de aprendizaje puede ser formulado como:

Dado un el conjunto de entrenamiento $\{(\mathbf{X}_1,d_1),(\mathbf{X}_2,d_2),\cdots,(\mathbf{X}_T,d_T)\}$, donde $\mathbf{X}_i=(x_{i1},x_{i2},\cdots,x_{in})\in\mathbf{R}^n$, $i=1,\cdots,T$ representan a los patrones de entrenamiento y $d_i=(d_{i1},d_{i2},\cdots,d_{iR})$, $i=1,\cdots,T$ es el vector de salida deseada correspondiente al patrón X_i, siendo

$$d_{ij}=\begin{cases}1, \text{ si } \mathbf{X}_i\in C_k, j=k\\ -1, \text{ si } \mathbf{X}_i\in C_k, j\neq k\end{cases},$$

determinar una matriz de pesos

$$\overline{W}=\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & w_{1,n+1}\\ w_{21} & w_{22} & \cdots & w_{2n} & w_{2,n+1}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ w_{R1} & w_{R2} & \cdots & w_{Rn} & w_{R,n+1}\end{bmatrix}$$

de forma tal que $E=\frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{R}(d_{ij}-o_{ij})^2<E_{max}$,

donde $o_{ij}=\frac{2}{1+e^{-\lambda net_{ij}}}-1$, $i=1,\cdots,T$; $j=1,\cdots,R$ y E_{max} es el error de aproximación.

Los componente de la matriz de pesos se modifica según la expresión

$$\Delta w_{ij}=-c\frac{\partial E}{\partial w_{ij}}$$

Como los pesos se modifican al presentar a la red cada patrón, entonces la función de error puede ser escrita como $E=\sum_{i=1}^{T}E_i$, donde $E_i=\frac{1}{2}\sum_{j=1}^{R}(d_{ij}-o_{ij})^2$ es el error asociado al patrón X_i.

En este caso los pesos se modifican según la expresión

$$\Delta\overline{W}_j=c(d_{ij}-o_{ij})f'(\overline{W}_j{}^t Y_i)Y_i$$

donde

$$\overline{W}_i=[w_{i1},w_{i2},\cdots,w_{in},w_{i,n+1}]^t,$$

$$Y_i = \left[x_{i1}, x_{i2}, \cdots, x_{in}, -1\right]^t$$

- **Regla de aprendizaje para redes multicapas.**

Cuando el conjunto de patrones no es linealmente separable, una red neuronal con una sola capa de neuronas no es capaz de realizar correctamente la tarea de clasificación. En estos casos son necesarias superficies de decisión no lineales para separar el conjunto de patrones, por lo que se implementan redes con capas de neuronas ocultas.

En la figura 17 se muestra una red neuronal multicapa con dos capas ocultas de neuronas.

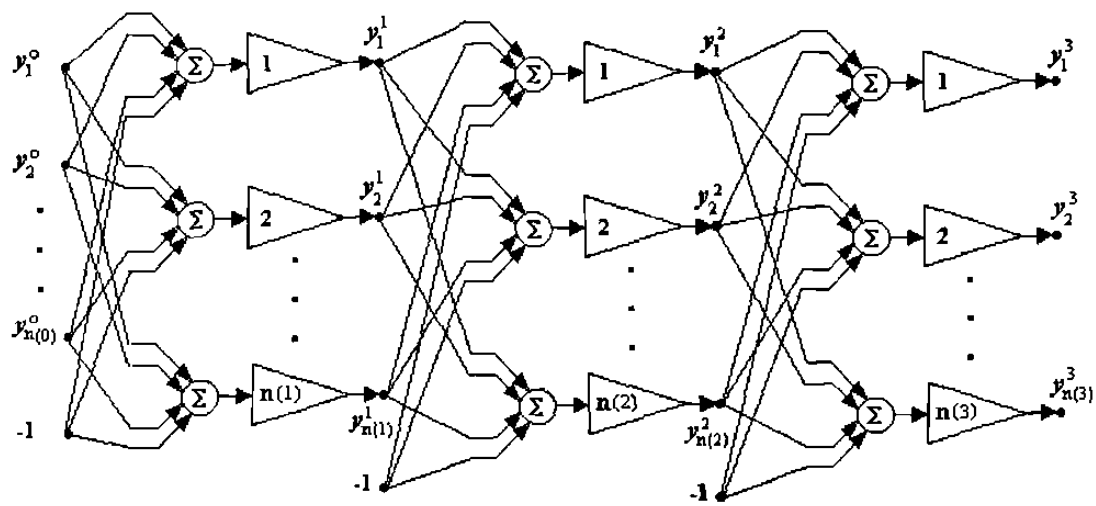


Figura 17 : Red Neuronal con dos capas ocultas.

Hasta el momento no se ha encontrado ninguna generalización de la regla de aprendizaje del perceptrón para redes multicapas, solo existen algunos algoritmos constructivos de aprendizaje que van agregando neuronas durante el proceso de entrenamiento, que están basados en la regla de aprendizaje del perceptrón discreto o en alguna de sus modificaciones.

Para el entrenamiento de redes multicapas de propagación hacia adelante se utiliza un algoritmo, conocido como el algoritmo de retropropagación del error (*backpropagation*) y es una generalización de la regla delta de aprendizaje.

El algoritmo de retropropagación del error es un algoritmo iterativo, basado en la técnica del descenso acelerado y su objetivo de entrenamiento consiste en minimizar determinada función de error.

Supongamos que se tiene un conjunto de aprendizaje no contradictorio de la forma $\{(X_1,\mathbf{d}_1),(X_2,\mathbf{d}_2),\cdots,(X_T,\mathbf{d}_T)\}$,

donde $X_i=(x_{i1},x_{i2},\cdots,x_{in})^t\in R^n$, $(i=1,...,T)$ representan los de patrones de entrenamiento y $\mathbf{d}_i\in R^L$ es la salida deseada correspondiente al patrón X_i , siendo además

$$d_{ij}=\begin{cases}1,\ \text{si}\ \mathbf{X}_i\in C_k, j=k\\ -1,\ \text{si}\ \mathbf{X}_i\in C_k, j\neq k\end{cases}$$

Para analizar el funcionamiento del algoritmo de retropropagación del error en el entrenamiento de una red con $L-1$ capas de neuronas ocultas $(L\geq 2)$, se introducen las siguientes notaciones:

- $n(l)$ a la cantidad de neuronas en la capa l, $(0\leq l\leq L)$.
- y_{ij}^l a la salida de la neurona j en la capa l para el vector X_i.
- w_{kj}^l al peso de la conexión entre la neurona k de la capa l y la neurona j de la capa $l-1$
- net_{ij}^l a la entrada a la neurona j en la capa l, $(1\leq l\leq L)$ para el vector X_i.
- e_{ij} al error generado por el vector X_i en la neurona j de la capa de salida.

Según la notación utilizada, entonces

$n(0)=n$

$y_{ij}^0=x_{ij}$, $(j=1,\cdots,n(0))$

$y_{i,n(0)+1}^0=-1$

Cada iteración del algoritmo consta de dos faces, una hacia adelante y otra hacia atrás.

En la fase hacia adelante se le presenta a la red un patrón y esta calcula la salida y el error correspondientes.

En esta primera fase se calcula sucesivamente para $l=1,\cdots,L$

$$net_{ij}^{l} = \sum_{k=1}^{n(l-1)} w_{jk}^{l} y_{ik}^{l-1},\ \left(i = 1,\cdots,n(l-1)+1\ ;\ j = 1,\cdots,n(l)\right)$$

$y_{ij}^{l} = f(net_{ij}^{l})$, donde f es la función de activación continua bipolar.

Al llegar a la capa de salida se determina el error asociado al patrón X_i como

$$E_i = \sum_{j=1}^{n(L)} \left(e_{ij}\right)^2$$

donde $e_{ij} = d_{ij} - y_{ij}^{L}$

La función del error total de un ciclo de entrenamiento se puede expresar como el cuadrado de la norma euclidiana de la diferencia entre la salida deseada y la salida calculada por la red, es decir,

$$E = \sum_{i=1}^{T} E_i = \sum_{i=1}^{T} \sum_{j=1}^{n(L)} \left(e_{ij}\right)^2$$

En la segunda fase el error es propagado hacia atrás, a través de la red y los pesos son modificados según la dirección negativa del gradiente de la función de error.

En un proceso de propagación hacia atrás, para $l = L, L-1,\cdots,1$, se calcula

$$\delta_{ij}^{l} = \begin{cases} e_{ij} f'(net_{ij}^{l}),\ l = L \\ f'(net_{ij}^{l}) \sum_{k=1}^{n(l+1)} \delta_{ik}^{l+1} w_{kj}^{l+1},\ l = L-1,\cdots,1 \end{cases}$$

y los pesos se modifican según la regla delta generalizada

$$\Delta w_{kj}^{l} = c\delta_{ij}^{l} y_{ij}^{l-1}$$

La presentación completa del conjunto de entrenamiento es conocida como ciclo de entrenamiento y el algoritmo termina cuando al concluir un ciclo de entrenamiento, el error total del ciclo es menor que un error prefijado con antelación.

2.4.5 Algoritmos constructivos de aprendizaje.

Para el entrenamiento de redes multicapas, el algoritmo de aprendizaje utilizado tradicionalmente es el algoritmo de retropropagación del error. Este algoritmo

presenta la desventaja de que presupone conocida la arquitectura de la red, es decir, el número de capas y la cantidad de neuronas por capas.

Los algoritmos constructivos de aprendizaje [10,11,79,100,102] son procedimientos que se usan para diseñar y entrenar redes neuronales multicapas para la clasificación. Estos algoritmos obtienen redes neuronales de arquitectura sub-optimal en el sentido del número de neuronas en las capas ocultas.

En la mayoría de estos algoritmos, el entrenamiento está basado en alguna variante de la regla de aprendizaje del perceptrón discreto y su funcionamiento consiste en ir agregando neuronas a la red hasta que se logre que la igualdad a cero de la función de error para todos los patrones del conjunto de entrenamiento.

Es conocido que cuando el conjunto de patrones no es linealmente separable, es imposible realizar la clasificación correcta del conjunto de entrenamiento sin agregar capas de neuronas ocultas.

El principio general de funcionamiento de los algoritmos constructivos de aprendizaje es determinar, en cada iteración del algoritmo, un vector de pesos y un valor de umbral que proporcione el valor mínimo de la función de error, el cual es igual a cero si el conjunto de entrenamiento es linealmente separable.

Entre los algoritmos constructivos, los más eficientes son [10,11]:

- Algoritmo de bolsa con mecanismo de reten (*Pocket algorithm with ratchet modification*).
- Algoritmo del perceptrón térmico (*Thermal perceptron algorithm*).
- Procedimiento de corrección baricéntrica (*Barycentric correction procedure*).

Esta comprobado [90] que el problema de determinar el mayor subconjunto linealmente separable de un conjunto de entrenamiento es NP-completo. De aquí que todos estos algoritmos constructivos son procedimientos heurísticos que en cada iteración tratan de determinar, con una complejidad polinomial, el mayor subconjunto linealmente separable.

El algoritmo de bolsa con mecanismo de reten utiliza la regla del perceptrón para la modificación de los pesos y guarda en un vector W_{pocket} el vector de pesos que proporciona el menor valor de la función de error. En cada iteración se compara el

valor de la función de error para el vector de pesos W calculado con el valor para W_{pocket} y si este valor es menor, se reemplaza W_{pocket} por W. Está comprobado [26] que este algoritmo converge al menor valor de la función de error.

El algoritmo del perceptrón térmico es utilizado para controlar la modificación de los pesos durante el proceso de entrenamiento. En el algoritmo clásico de entrenamiento del perceptrón, cuando el conjunto de patrones no es linealmente separable, pueden ocurrir cambios bruscos en los pesos, que producen fluctuaciones severas en la función de error y entorpecen el proceso de clasificación.

Para estabilizar el proceso de aprendizaje se introduce el siguiente factor amortiguador en la ecuación de modificación de los pesos

$$W \leftarrow W + c\left[d_i - o_i)\right] X_i \exp\left(-\frac{|W^t . X_i|}{Q}\right)$$

El valor de Q se le da un valor de Q_0 al comienzo del entrenamiento y gradualmente se aproxima a cero a medida que progresa el aprendizaje.

Este factor amortiguador introducido en la regla de modificación de los pesos no permite cambios bruscos en los pesos al final del entrenamiento.

El procedimiento de corrección baricéntrica es un algoritmo eficiente para entrenar una unidad lógica de umbral. En este procedimiento los patrones son separados en dos subconjuntos S^+ y S^-. El baricentro de cada subconjunto se define como la media ponderada de los patrones multiplicados por su correspondiente coeficiente de peso. El vector de pesos $W = \left(w_1, w_2, \cdots, w_n\right)^t$ es determinado como la diferencia entre los baricentros de los dos subconjunto de patrones y el valor de umbral w_{n+1} es seleccionado de forma tal que minimice la función de error. Inicialmente a cada patrón se le asocia un coeficiente de peso igual 1.

Si el conjunto de patrones es lincalmcnte separable, este procedimiento determina, de forma mas eficiente que los dos algoritmos anteriores, el hiperplano que separa al conjunto de patrones en las dos clases o categorías.

2.5 Teorema de Aproximación Universal.

Después de encontrar un algoritmo de entrenamiento para redes multicapas, el problema fundamental que enfrentaron los investigadores en redes neuronales fue determinar la menor cantidad de capas de neuronas ocultas para que este algoritmo fuera convergente.

En 1989 se da solución a este problema mediante la demostración de un teorema que es conocido como el *Teorema de Aproximación Universal*. Este teorema está considerado como el resultado teórico de mayor importancia para redes neuronales de propagación hacia adelante y fue reportado en tres trabajos diferentes: Cybenko [15], Funahashi [25] y Hornik, Stinchcombe y White [33].

El teorema puede ser formulado como:

Teorema: Sea $\varphi(.)$ una función continua, monótona creciente y acotada. Denotemos por $\mathbf{I}_n$ el hipercubo unitario n-dimensional $[0, 1]^n$ y por $C(\mathbf{I}_n)$ el espacio de las funciones continuas sobre $\mathbf{I}_n$. Entonces, dados una función $f \in C(\mathbf{I}_n)$ y un $\varepsilon > 0$, existen un entero m y conjuntos de constantes reales α_i, θ_i y w_{ij}, donde $i = 1,\cdots,m$; $j = 1,\cdots,n$ tales que se puede definir

$$F(x_1,\ldots,x_n) = \sum_{i=1}^{m} \alpha_i \varphi\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i\right)$$

como una aproximación de la función f, que cumple

$$\left|F(x_1,\ldots,x_n) - f(x_1,\ldots,x_n)\right| < \varepsilon \qquad \text{para todo} \quad (x_1,\ldots,x_n) \in \mathbf{I}_n .$$

El teorema de aproximación universal es un teorema de existencia y establece que una sola capa oculta es suficiente para que una red neuronal multicapa calcule una aproximación uniforme para un conjunto de entrenamiento dado, representado por el conjunto de entradas $(x_1,\ldots,x_n)$ y una salida deseada $f(x_1,\ldots,x_n)$.

La demostración de este teorema está fundamentada en el teorema de aproximación por superposición de Kolmogorov. En el entorno de las redes neuronales, estos dos resultados han sido ampliamente discutidos y analizados por

diferentes autores [34,36,37,41,42,43,45,56,57,58,74,75,85,96], además, han sido aplicados a redes neuronales con otros tipos de funciones de activación [20,31,78].

El teorema de aproximación universal no impone ninguna restricción sobre la cantidad neuronas en la capa oculta ni sobre la magnitud de los valores de los pesos de la red. De aquí que deja abiertos los problemas de si se mantiene la propiedad de aproximación universal, si se imponen restricciones a la cantidad de neuronas en la capa oculta y/o a los pesos de la red.

2.6 Conclusiones.

En este capítulo se da una breve introducción a la teoría de las redes neuronales para la clasificación.

El objetivo de este capítulo es describir y explicar los conceptos, métodos y procedimientos que, a nuestro juicio, son necesarios para una mejor comprensión de este trabajo de tesis.

Comenzamos con la explicación de la teoría del funcionamiento del cerebro en que se fundamentan los modelos de redes neuronales artificiales y de como esta teoría es simplificada para conformar los modelos artificiales.

Luego tratamos el problema de clasificación de patrones y como este puede ser abordado mediante los modelos de redes neuronales de propagación hacia adelante.

Más adelante, tratamos el problema del aprendizaje de las redes neuronales de propagación hacia adelante, describiendo las diferentes reglas clásicas de aprendizaje, así como los algoritmos constructivos de aprendizaje de mayor eficiencia computacional, reportados en la literatura.

Al final del capítulo incluimos el teorema de aproximación universal, debido a su importancia dentro de la teoría de las redes neuronales de propagación hacia adelante, ya que garantiza la existencia de la solución del problema de entrenamiento.

CAPITULO 3

DISEÑO DE REDES NEURONALES DE PROPAGACION HACIA ADELANTE PARA LA CLASIFICACION

3.1 Introducción.

Uno de los problemas fundamentales en las aplicaciones de los modelos de redes neuronales multicapas de propagación hacia adelante es el problema del diseño de la red, es decir, la cantidad de neuronas en cada capa.

La cantidad de neuronas en las capas de entrada y de salida está determinada por la naturaleza del problema a resolver, por lo que el problema de diseño de la red se reduce a determinar cuántas neuronas deben tener las capas ocultas.

El teorema de aproximación universal [15,25,33] garantiza que una sola capa oculta es suficiente para que los modelos de redes neuronales de propagación hacia adelante puedan aproximar uniformemente cualquier función continua soportada sobre un hipercubo unitario, pero deja abierto el problema de la cantidad de neuronas necesarias en esta capa oculta para que se alcance esta propiedad.

Este teorema, también es aplicable a redes neuronales con funciones discretas de activación en las neuronas, que se utilizan para la clasificación y/o reconocimiento de patrones [7,27,105].

En otros trabajos posteriores [27,105] aplicados al problema de clasificación de patrones, se demuestra que con $(T-1)$ neuronas en la capa oculta, donde T es la cardinalidad del conjunto de entrenamiento, la superficie de error no presenta mínimos locales, por lo que el algoritmo de aprendizaje es convergente. Pero en la práctica este valor $(T-1)$ resulta demasiado grande y además, en la resolución de diferentes problemas prácticos, se ha comprobado que el algoritmo de entrenamiento es convergente con menor número de neuronas en la capa oculta.

Los métodos más usuales, que aparecen en la literatura para determinar la cantidad de neuronas en la capa oculta, son los llamados procedimientos de depuración y crecimiento [29,44], que operan por ensayo y error. Esto conlleva a que el proceso de entrenamiento de la red sea muy costoso computacionalmente.

En los procedimientos de depuración se construye una red de gran tamaño y se eliminan neuronas mientras se mantenga la convergencia y en los de crecimiento se construye una red pequeña y se agregan neuronas hasta que se alcance la convergencia del algoritmo de entrenamiento. Además, estos métodos utilizan como algoritmo de entrenamiento al algoritmo de retropropagación del error, el cual es criticado en la literatura por su alto costo computacional.

Otra forma de abordar el problema en redes neuronales para la clasificación, ha sido mediante algoritmos constructivos [10,11,12,23,71,80,89,103]. Estos algoritmos tratan de encontrar el mayor subconjunto linealmente separable de un conjunto de entrenamiento dado y su funcionamiento se basa en diferentes modificaciones del algoritmo de aprendizaje del perceptrón discreto.

En este capítulo se trata el problema de diseño de una red neuronal para la clasificación. Este problema se puede enunciar como: *Determinar el número de neuronas en la capa oculta para que una red neuronal, con funciones discretas de activación en las neuronas, clasifique correctamente a un conjunto de patrones de entrenamiento.*

3.2 Análisis del problema de clasificación.

Analicemos el problema de clasificación de un conjunto de patrones en dos clases o categorías C_1 y C_2.

Denotemos por $\{(X_1,d_1),(X_2,d_2),\cdots,(X_T,d_T)\}$ el conjunto de aprendizaje, donde $X_i=(x_{i1},x_{i2},\cdots,x_{in})^t \in R^n$, $(i=1,...,T)$ representan los patrones de entrenamiento y d_i es la salida deseada correspondiente al patrón X_i, siendo además

$$d_i=\begin{cases} 1, \text{ si } X_i \in C_1, \\ -1, \text{ si } X_i \in C_2 \end{cases}$$

Cuando un conjunto de patrones es linealmente separable, entonces existen un vector de pesos $W=(w_1,w_2,\cdots,w_n)^t \in R^n$ y un valor de umbral $w_{n+1} \in R$, tales que para cada patrón de entrenamiento X_i se cumple que

$$\begin{cases} \sum_{j=1}^{n}(x_{ij}.w_j)-w_{n+1}>0, \text{ si } X_i \in C_1 \\ \sum_{j=1}^{n}(x_{ij}.w_j)-w_{n+1}<0, \text{ si } X_i \in C_2 \end{cases} \tag{3.1}$$

Esto significa que existe un hiperplano de la forma $X^t.W-w_{n+1}=0$ que separa el conjunto de patrones en las dos clases C_1 y C_2.

Si se multiplica cada una de las desigualdades en (3.1) por la salida deseada del patrón correspondiente, obtenemos una condición equivalente

$$d_i\left(\sum_{j=1}^{n}(x_{ij}.w_j)-w_{n+1}\right)>0,\ i=1,\cdots,T \tag{3.2}$$

Cada una de estas desigualdades en (3.2) representa un semiespacio abierto en el espacio de pesos y si el conjunto de patrones es linealmente separable, este sistema de T desigualdades representa el interior de un cono poliédrico, que es a su vez, el conjunto de soluciones factibles de este sistema de desigualdades.

En el caso en que el conjunto de patrones no sea linealmente separable, el conjunto de soluciones factibles de este sistema de desigualdades es vacío, lo que

significa que para cualesquiera vector de pesos W y valor de umbral w_{n+1}, siempre existe un grupo de desigualdades en (3.2) que se incumplen.

En este caso, se determinará un hiperplano $X'.W - w_{n+1} = 0$ que separe correctamente la mayor cantidad de patrones, por lo que es necesario determinar un vector W y un umbral w_{n+1} para los cuales se cumplan la mayor cantidad de desigualdades en (3.2).

Este hiperplano divide el conjunto de patrones en dos subconjuntos, los cuales pueden contener patrones de las dos clases, por lo que para cada uno de estos subconjuntos se buscará un hiperplano que dé la mayor cantidad de patrones bien clasificados y se procederá así sucesivamente hasta que sean separados todos los patrones.

En el principio de determinar el hiperplano que separe la mayor cantidad de patrones se basan todos los procedimientos de diseño de redes para la clasificación que aparecen reportados en la literatura. Estos procedimientos tienen la dificultad de no separar ningún patrón, cuando la menor cantidad de patrones mal clasificados coincide con todos los patrones de una misma clase.

Para eliminar esa dificultad, desde el principio, buscará un hiperplano que clasificando correctamente todos los patrones de una misma clase, garantice la mayor cantidad de patrones bien clasificados de la otra clase.

Para determinar este hiperplano, en el sistema de desigualdades (3.2) se introducen T variables auxiliares $z_1, z_2, \cdots, z_T$, de la siguiente forma

$$d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) - z_i = 0 \quad , \quad (i = 1, \cdots, T) \tag{3.3}$$

Si el conjunto de patrones es linealmente separable, entonces existen un vector de pesos $W = \left(w_1, w_2, \cdots, w_n\right)'$ y un valor de umbral w_{n+1}, para los cuales el vector de las variables auxiliares $Z = \left(z_1, z_2, \cdots, z_T\right)$ tiene todas sus componentes positivas.

En caso en que el conjunto de patrones no sea linealmente separable, en el sistema (3.2) existen desigualdades que se incumplen y por tanto, algunas de las variables auxiliares toman obligatoriamente valores menores o iguales que cero.

Con esta forma de introducir las variables auxiliares se logra una doble correspondencia entre patrón bien clasificado y variable auxiliar positiva.

Ahora, el problema se ha transformado en determinar las soluciones $\left(w_1, w_2, \cdots, w_n, w_{n+1}, z_1, z_2, \cdots, z_T\right)$ del sistema de ecuaciones (3.3) que contengan la mayor cantidad de valores positivos correspondientes a las variables auxiliares $z_1, z_2, \cdots, z_T$ y que entre las positivas, estén todas las que pertenecen a una misma clase.

El sistema de ecuaciones (3.3) puede ser escrito como

$$d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right)-w_{n+1}\right)=z_i \quad , \quad (i=1,\cdots,T)\,, \tag{3.4}$$

o en forma matricial,

$$\begin{bmatrix} d_1x_{11} & d_1x_{12} & \cdots & d_1x_{1n} & -d_1 \\ d_2x_{21} & d_2x_{22} & \cdots & d_2x_{2n} & -d_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_Tx_{T1} & d_Tx_{T2} & \cdots & d_Tx_{Tn} & -d_T \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n+1} \end{bmatrix}=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_T \end{bmatrix},$$

teniendo por matriz asociada, la matriz

$$\left[\begin{array}{ccccc|cccc} d_1x_{11} & d_1x_{12} & \cdots & d_1x_{1n} & -d_1 & 1 & 0 & \cdots & 0 \\ d_2x_{21} & d_2x_{22} & \cdots & d_2x_{2n} & -d_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ d_Tx_{T1} & d_Tx_{T2} & \cdots & d_Tx_{Tn} & -d_T & 0 & 0 & \cdots & 1 \end{array}\right]$$

Supóngase que el conjunto de entrenamiento contiene T_1 patrones de la clase 1 y $T-T_1$ patrones de la clase 2 , además que está ordenado de forma tal que primero se encuentran los T_1 patrones de la clase 1.

Entonces para determinar el hiperplano separador hay que resolver uno de los siguientes problemas:

$$\begin{cases} max \sum_{j=T_1+1}^{T} \operatorname{sgn}(z_j) \\ s.\ a \quad d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) = z_i, \quad i = 1,\cdots,T \\ \qquad z_j > 0, \quad j = 1,\cdots,T_1 \end{cases} \tag{P3.1}$$

$$\begin{cases} max \sum_{j=1}^{T_1} \operatorname{sgn}(z_j) \\ s.\ a \quad d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) = z_i, \quad i = 1,\cdots,T \\ \qquad z_j > 0, \quad j = T_1+1,\cdots,T \end{cases} \tag{P3.2}$$

Después de obtener un hiperplano separador, el conjunto de patrones se divide en dos subconjuntos. Si el conjunto de patrones no es linealmente separable, entonces uno de estos subconjuntos contiene patrones de las dos clases, por lo que a este subconjunto se le aplica de nuevo el procedimiento descrito. Este proceso continúa, hasta que se encuentre un subconjunto linealmente separable.

Cada uno de estos hiperplanos separadores representa una neurona de la capa oculta de la red y las componentes del vector W y el valor de w_{n+1}, asociados a cada hiperplano, representan los pesos y el umbral respectivamente, entre la capa de entrada y cada neurona de la capa oculta de la red.

3.3 Cota superior para el número de neuronas en la capa oculta.

El procedimiento de separación de patrones que se propone, se basa en dividir el conjunto de patrones en dos subconjuntos de forma tal que: la intersección de sus envolturas convexas sea vacía, uno de estos subconjuntos contenga patrones de una sola clase y la cardinalidad del subconjunto que contiene patrones de una sola clase sea máxima.

Esto permite determinar un hiperplano separador que divide al conjunto de patrones en dos subconjuntos, donde en uno de ellos solo hay patrones de una misma clase y además la cardinalidad de este subconjunto es máxima.

En el proceso de separación de los patrones se obtiene un árbol como el que se muestra en la figura 18.

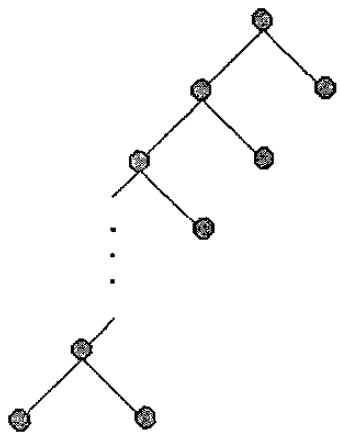


Figura 18: Arbol asociado al proceso de división.

El nodo raíz de este árbol representa al conjunto de patrones de entrenamiento y los restantes nodos representan los subconjuntos en que se va dividiendo este conjunto de patrones. Los nodos hojas del árbol representan los subconjuntos en que se divide al final el conjunto original de patrones y la cantidad de estos nodos hojas, disminuida en una unidad, representa la cantidad de hiperplanos que se utilizaron para separar el conjunto de patrones.

En el peor de los casos, en cada división se logra separar un solo patrón y el último subconjunto a dividir estará formado por dos patrones de diferentes clases, siendo en este caso la cantidad de nodos hoja será igual a T y la cantidad de hiperplanos igual a $T-1$. Este último valor coincide con el valor conocido de la cota superior del número de neuronas en la capa oculta para que la superficie de error no presente mínimos locales.

3.4 Algoritmo de solución.

3.4.1 Transformación de los problemas planteados.

Para facilitar la resolución de los problemas (P3.1) y (P3.2), se realiza un intercambio entre variables libres y variables dependientes del sistema, mediante un proceso de eliminación gaussiana, de forma tal que queden como variables dependientes las componentes del vector de pesos y el umbral.

Nótese que el rango del sistema de ecuaciones homogéneas (3.3) es igual a T por lo que siempre tiene soluciones diferente de la solución trivial, además el sistema fundamental de soluciones consta de $n+1$ soluciones. Esto significa que hay $n+1$ variables libres y T variables dependientes.

Si el rango de la matriz

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & -1 \\ x_{21} & x_{22} & \cdots & x_{2n} & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{T1} & x_{T2} & \cdots & x_{Tn} & -1 \end{bmatrix},$$

formada con los patrones ampliados, es igual a $n+1$, entonces se pueden intercambiar las variables que representan los pesos y el umbral con $n+1$ de las variables auxiliares, obteniendo una matriz de la siguiente estructura

$$\left[\begin{array}{ccccc|cccc} 1 & 0 & \cdots & 0 & & a_{11} & a_{12} & \cdots & a_{1,n+1} & 0 \cdots 0 \\ 0 & 1 & \cdots & 0 & & a_{21} & a_{22} & \cdots & a_{2,n+1} & 0 \cdots 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \vdots \vdots \\ 0 & 0 & \cdots & 1 & & a_{n+1,1} & a_{n+1,2} & \cdots & a_{n+1,n+1} & 0 \cdots 0 \\ 0 & 0 & \cdots & 0 & & a_{n+2,1} & a_{n+2,2} & \cdots & a_{n+2,n+1} & 1 \cdots 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \vdots \vdots \\ 0 & 0 & \cdots & 0 & & a_{T,1} & a_{T,2} & \cdots & a_{T,n+1} & 0 \cdots 1 \end{array}\right]$$

Si $rank\left(X\right) = n_1 < n+1$, entonces al realizar el proceso de intercambio entre variables libres y variables dependientes, solo n_1 de las componentes del vector $\overline{W} = \left(w_1, w_2, \cdots, w_n, w_{n+1}\right)'$ pasan a ser variables dependientes.

En este caso la matriz del sistema (3.4) se transforma a una matriz, donde entre las columnas asociadas a las variables $w_1, w_2, \cdots, w_n, w_{n+1}$, solo existen n_1 columnas formadas por vectores canónicos y las $n+1-n_1$ columnas restantes contienen, en general, n_1 componentes distintas de cero.

Por ejemplo, se obtiene una matriz de la siguiente estructura, aunque el orden de las columnas no es necesariamente el que se presenta.

$$\left[\begin{array}{cccccc|cccc} 1 & 0 & \cdots & 0 & \widetilde{x}_{1,n_1+1} & \cdots & \widetilde{x}_{1,n+1} & a_{11} & a_{12} & \cdots & a_{1n_1} & 0\cdots 0 \\ 0 & 1 & \cdots & 0 & \widetilde{x}_{2,n_1+1} & \cdots & \widetilde{x}_{2,n+1} & a_{21} & a_{22} & \cdots & a_{2n_1} & 0\cdots 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\cdots\vdots \\ 0 & 0 & \cdots & 1 & \widetilde{x}_{n_1,n_1+1} & \cdots & \widetilde{x}_{n_1,n+1} & a_{n_1,1} & a_{n_1,2} & & a_{n_1,n_1} & 0\cdots 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & a_{n_1+1,1} & a_{n_1+1,2} & \cdots & a_{n_1+1,n_1} & 1\cdots 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots\cdots\vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & a_{T1} & a_{T2} & \cdots & a_{Tn_1} & 0\cdots 1 \end{array}\right]$$

Como el objetivo es determinar un vector $\overline{W} = \left(w_1, w_2, \cdots, w_n, w_{n+1}\right)^t \neq \overline{0}$, que le correspondan valores no nulos de las variables auxiliares, entonces se les pueden asignarle valores iguales a cero a todas las componentes del vector $\overline{W}$ que quedaron como variables libres. Esto significa que existen $n+1-n_1$ componentes en los patrones ampliados que no tienen influencia sobre la determinación del hiperplano separador y por tanto en este caso se puede reducir a n_1 la dimensión de los patrones ampliados.

De todo esto se puede concluir que cuando $rank\left(X\right) = n_1 < n+1$, el vector $\overline{W}$ tiene $n+1-n_1$ componentes iguales a cero y para tener uniformidad en el desarrollo del procedimiento, se realizará la asignación $n \leftarrow n_1 - 1$.

Supóngase ahora, por comodidad de notación, que las primeras $n+1$ variables auxiliares $z_1, z_2, \cdots, z_{n+1}$ son ahora las variables libres, entonces el sistema (3.4) se transforma en