30

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

CAMPUS ARAGON

REDES NEURONALES APLICADAS AL RECONOCIMIENTO DE HUELLAS DACTILARES, UTILIZANDO EL ALGORITMO DE BACKPROPAGACION

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N

MERCEDES JIMENEZ BARBOSA
ESTELA MARTINEZ CRUZ

1996

TESIS

COMPLETA

DEDICATORIA

A mi Dios. A mi padre con respeto y amor, por el apoyo que me ha mostrado y del sentido de responsabilidad que ha sabido inculcarme.

A mi madre con admiración, por su cariño, ternura y valioso consejos que me han ayudado a ser fuerte.

Gracias a ambos por dedicarme su tiempo, sus cuidados, su ejemplo desde siempre y que me han ayudado a levantarme en mis tropiezos y seguir adelante. Les agradezco esta valiosa herencia que me han proporcionado y por culminar conmigo este primer gran objetivo de mi vida.

A mis hermanos por su apoyo y comprensión, ya que este Titulo es resultado del cariño que tenemos en la familia.

Los quiero.

Mercedes.

A ti Estela por realizar este trabajo juntas y brindarme tu apoyo y comprensión.

Mercedes.

Este trabajo lo dedico a mis padres:

Quienes me dieron todo lo que estaba en sus posibilidades darme y me quieren a pesar de mis defectos y errores.

Para ellos que siguen creyendo en mí. Con respecto y cariño.

A mis hermanos, ya que sin su apoyo no hubiese podido seguir adelante.

Estela

Agradecimientos Especiales

A todas aquellas personas que nos brindaron su ayuda desinteresadamente y que contribuyeron a la elaboración de este trabajo con sus aportaciones, consejos y estímulos, aún sin conocernos ampliamente.

A todas ellas nuevamente GRACIAS.

Parábola de la educación

Iba un hombre caminando por el desierto cuando oyó una voz que le dijo:

Levanta unos guijarros, mételos a tu bolsillo y mañana te sentirás a la vez triste y contento.

Aquel hombre obedeció. Se inclinó, recogió un puñado de guijarros y se los metió en el bolsillo.

A la mañana siguiente, vió que los guijarros se habían convertido en diamantes, rubíes y esmeraldas.

Y se sintió feliz y triste.

Feliz, por haber recogido los guijarros; triste, por no haber recogido más.

Lo mismo ocurre con la educación.

William Cunningham

PROLOGO

Cuando nuestro asesor de tesis nos propuso realizar nuestro trabajo sobre redes neuronales, nos pareció una buena idea, aunque presentaba algunos inconvenientes. Primeramente la poca bibliografía existente, ya que considerábamos reciente el tema. Segundo la bibliografía existente estaba en inglés.

Debido a estos inconvenientes inicialmente tomamos la decisión de realizar nuestros trabajo 100% teórico, ya que tendríamos que emplear mucho tiempo en la búsqueda de bibliografía y en la traducción de dicha información.

Con el paso del tiempo nos dimos cuenta que el tema ya tenía antecedentes, además que si existía bibliografía suficiente para llevar a cabo nuestro trabajo. Tuvimos que emplear bastante tiempo en la traducción de dicha información, pero nos fuimos adentrando tanto en el tema, que reconsideramos nuevamente los alcances de nuestro trabajo y decidimos realizar una pequeña aplicación con redes neuronales, ya que no representaba mucha dificultad la implementación del algoritmo.

Y así fue, la implementación del algoritmo no representó gran dificultad. Lo que si presentó dificultad fue la conversión de los datos de la huella escaneada a un formato que pudiese ser leído por el algoritmo. Pero esta dificultad fue solucionada gracias a un compañero que nos proporciono un programa que realizaba dicha tarea. Lo demás fue más sencillo.

INDICE

Pág.

INTRODUCCION

La Inteligencia Artificial es la ciencia que trata de la comprensión de la inteligencia y el diseño de máquinas inteligentes, es decir del estudio y la simulación de las actividades intelectuales del hombre, tales como manipulación, razonamiento, percepción, aprendizaje, creación, etc.

La inteligencia artificial estudia las máquinas que ejecutan tareas propias del ser humano en las que se manifiesta la inteligencia, también estudia la modelización del pensamiento.

Algunas de las ramas más importantes de la inteligencia artificial son las siguientes:

- Robótica.- Es la rama encargada del diseño y desarrollo de máquinas capaces de realizar aquellos procesos mecánicos y repetitivos de los cuales es capaz el hombre.

- Sistemas Expertos.- Son la rama creada del diseño y desarrollo de sistemas que intentan simular el comportamiento de un experto en una área determinada, utilizando para ello conocimientos hipotéticos fundamentados en la experiencia.

- Procesamiento del lenguaje natural (PLN).- Estudia el proceso de comunicación entre el hombre y las computadoras mediante algún tipo de lenguaje natural propio del hombre (comunicación oral, o escrita). Es un problema que involucra un estudio sintáctico, semántico y pragmático de las expresiones del hombre.

- Visión por ordenar o visión por computadora.- Consiste en la identificación, inspección, localización y verificación de objetos por computadora.

- Aprendizaje Automático.- Es la rama cuyo estudio se enfoca al aprendizaje de programas de computadora de manera automática.

- Tratamiento Inteligente de la Información.- Estudia las maneras más inteligentes de manipular la información.

- Programación Automática.- Es la rama que estudia la generación automática de programas que satisfagan una amplia variedad de problemas.

- Redes Neuronales.- Es la rama que estudia conjuntos de algoritmos que tienen la finalidad de resolver tareas de tipo cognoscitivo, como lo es el aprendizaje.

El objetivo de este trabajo es ampliar la visión sobre las redes neuronales en el campo del reconocimiento de patrones. Al mismo tiempo pretende dejar las bases necesarias para futuros trabajos en relación a las redes neuronales y sus aplicaciones en el reconocimiento de patrones. Hoy en día estos temas son de gran auge en los países desarrollados, pero en México se han desarrollado de manera esporádica.

Este trabajo de Tesis comprende cuatro capítulos, los cuales están desarrollados de la siguiente manera:

El *Capitulo I* establece los antecedentes, la definición, los componentes y el funcionamiento básico de las redes neuronales, así como sus diferentes tipos existentes.

En el *Capitulo II* se describe en forma breve las propiedades de las redes, así como dos grandes áreas que comprenden el reconocimiento de patrones el óptico y el de voz, para ello se presentan ejemplos con sus respectivos funcionamientos básicos.

En el *Capitulo III* se describen y comparan algunos algoritmos utilizados por las redes neuronales en el reconocimiento de patrones. Se hace una mejor descripción del método del algoritmo de backpropagación que es el que se utilizó para la solución del problema.

En el *Capitulo IV* se analiza y plantea la solución al problema que es el reconocimiento de una huella digital. Se describen los pasos que se realizaron y la justificación de cada uno de ellos, así mismo se detalla en programa que fue creado en lenguaje C.

Esperamos que este trabajo sirva como una fuente de información para aquellas personas interesadas en el tema.

CAPITULO I

LAS REDES NEURONALES

INTRODUCCION

Un cerebro humano esta construido de aproximadamente 2^{10} elementos inteligentes a los cuales se les denomina en términos biológicos Neuronas, capaces de comunicarse mediante una red de conexión compuesta de axones y sinápsis; una neurona contiene una cantidad de aproximadamente 2^{10} sinápsis. Dentro de la medicina, una hipótesis con respecto al modelo de un sistema nervioso biológico, dicta que las neuronas transmiten la información entre ellas mediante impulsos eléctricos.

La entrada de una red neuronal biológica esta compuesta por un conjunto de sensores que desempeñan la función de receptores los cuales proporcionan los estímulos provenientes del exterior al interior de la red; dichos estímulos están representados por impulsos eléctricos que se encargan de transmitir la información hacia el sistema nervioso central, provocando así, efectos de respuesta humana, la cual puede estar expresada en una gran variedad de acciones.

La información es manejada en tres etapas, la cuales son: el procesamiento, la evaluación y la comparación con la información que se encontraba previamente almacenada dentro del sistema nervioso central. En los momentos en que son requeridas, el sistema nervioso central genera señales de control que son transmitidas hacia los órganos motores, con el fin de tener un monitoreo de los mismos. Este rastreo se lleva a cabo mediante los enlazadores de retroalimentación que se encargan de verificar esta acción.

En general la estructura del sistema nervioso, se asemeja en muchas de sus características a las que posee un sistema de control infinito.

En ella, existe un fenómeno que se representa cuando se realiza la unión entre el axón de una neurona y la dendrita de la siguiente, al que se le conoce con el nombre de sinápsis, (aunque en realidad, durante este fenómeno el axón y la dendrita no se tocan). El impulso que se provoca debido a esta "unión" sólo puede atravesar en el sentido axón-dendrita.

La sinápsis sirve de válvula para impedir el flujo inverso de impulsos. La transmisión a través de una sinápsis es considerablemente más lenta que la transmisión a lo largo de un nervio. Cabe mencionar que la conexión sináptica afecta únicamente a áreas limitantes de las neuronas que participan.

Basado en lo anterior, una red neuronal artificial se constituye de patrones de entrada y salida, de elementos de procesamiento (EP), de conexiones de peso y operaciones preestablecidas.

1.1. ANTECEDENTES

Los primeros estudios llevados a cabo sobre redes neuronales artificiales datan de la década de los 40's, los cuales se le atribuyen a los investigadores Warren McCulloch y a Walter Pitts quienes propusieron su teoría general acerca de los sistemas de procesamiento de información. Dicha teoría describe el modelo de redes construidas a base de interruptores binarios o también conocidos como elementos de decisión que eran capaces de realizar operaciones lógicas así como funciones de carácter aritmético-lógico, debido a estas características se les acredito el nombre de *Neuronas*. McCulloch y Pitts (1943) publicaron el primer estudio sistemático de redes neuronales artificiales. En trabajos posteriores, (Pitts y McCulloch 1947) exploraron paradigmas de redes para reconocimiento de patrones y además de rotación y translación de objetos. Mucho de su trabajo envolvía el modelo de neurona simple. La unidad multiplica cada entrada x por un peso w, y suma las entradas pesadas. Si ésta suma es mayor que el umbral predeterminado, la salida es 1, de otra manera es 0. Estos sistemas (y sus variaciones) han sido llamados colectivamente Perceptrones.[1] En general, consisten de una sola capa de neuronas artificiales conectadas por pesos a un conjunto de entradas aunque más complicadas redes compartan el mismo nombre.

Mc Culloch y Pitts en su teoría manejan la diferencia que existe entre una red neuronal artificial y una computadora tradicional, la cual consiste en que en la primera, las instrucciones de un programa no se ejecutan de forma secuencial como ocurre en las computadoras tradicionales, sino en forma paralela.

[1] Un PERCEPTRON está constituido por dos niveles que se encuentran separados y que representan al nivel de entrada y salida respectivamente, los elementos de nivel de salida reciben las señales provenientes de los elementos que se encuentran en la entrada, pero nunca en sentido contrario; los elementos que se encuentran en un sólo nivel no se comunican entre sí, por lo que se dice que el flujo de información es estrictamente direccional.

Para el año de 1949 Donald Hebb propone una regla para comprender la conexión que existe entre las neuronas y que hasta la fecha se le conoce como la "Regla de Aprendizaje Hebbiana"[2], esta regla afirma que toda información proveniente del exterior puede almacenarse en el interior de las conexiones, además postula la técnica de aprendizaje que habría de servir como base para futuros desarrollos sobre este campo. La regla de aprendizaje de Hebb fue una de las primeras que aportó gran ayuda a la teoría de la red neuronal artificial.

En el transcurso de la década de los 50's, el investigador Marvin Minsky comenzó sus trabajos sobre las primeras neurocomputadoras, para finalmente en 1954 construir y aprobar modelos que había propuesto, en los que se adaptaron conexiones del tipo automático, y que servirían como base para el modelo denominado PERCEPTRON.

En los 60's los perceptrones crearon un gran interés y optimismo Rosemblatt (1962) demostró un teorema notable acerca del aprendizaje del perceptron. Widrow (Widrow 1961, 1963; Widrow y Angell 1962, Widrow y Hoff 1960) efectuó una serie de demostraciones convincentes de sistemas similares a los perceptrones, y los investigadores en todo el mundo explotaron entusiatamente el potencial de éstos sistemas. La euforia inicial fué reemplazada por desilusión ya que los perceptrones fallaban en ciertas tareas simples de aprendizaje. Minsky (Minsky y Papert 1969) analizaron éste problema con gran rigor y probaron que existen severas restricciones en lo que un perceptron unicapa puede representar, y de aquí, de lo que puede aprender.

En esta misma década un nuevo proyecto fue introducido, el cual fue llamado ADALINE que se deriva del inglés ADAptive LINEar Combiner, junto con una nueva regla de aprendizaje que lleva como nombre Regla de Aprendizaje Widrow-Hoff[3]. Algunas de las primeras aplicaciones que se hicieron con los modelos ADELINE y su extensión MADALINE que significa varias ADALINE (Many ADALINE) incluyeron el reconocimiento de patrones, pronósticos del tiempo y controles adaptivos.

En tanto en Japón, Sun Ichi Amari (1972, 1977) propone un estudio acerca del aprendizaje en redes neuronales construidas de elementos preestablecidos. Por su parte, Kunihico Fukushima desarrolla una arquitectura de red neuronal artificial a la cual llamó NEOCOGNITRONS[4], esta arquitectura fue diseñada para el renonocimiento de patrones de imágenes; estas redes realizaban una simulación de una imagen tomada mediante un dispositivo que simulaba una retina artificial, para posteriormente procesarla, utilizando para ello niveles bidimensionales de neuronas.

---

[2] Hebb, D. *"Organization of Behavior"* New York: Wiley, 1949.

[3] Widrow, B. and. M. Hoff. *"Adaptive Switching Circuits"*. in WESCON Convention Records, Vol. 4, 1960.

[4] Fukushima, K. *"Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition"*. Neural Networks, Vol. 1, 1988.

Durante este período, Little y Gordon Shaw así como John Hopfield estudiaron el proceso en el que una red neuronal logra almacenar y extraer información (sistema recursivo); además, Little apunta las similitudes que existen entre la red neural del tipo propuesto por Mcculloch - Pitts y los sistemas denominados Spins o Sistemas de Instantes Magnéticos Elementales o modelos Ising.[5]

En los años de 1968 y 1987 surgieron nuevamente programas dedicados a la investigación de redes neuronales artificiales, a tal grado de que llegaron a propagarse gran número de conferencias así como publicaciones de revistas enfocadas al campo de la neurocomputación.

A partir de 1986 cuando AT & T fabrica los primeros modelos de memorias neuronales en circuitos integrados, a partir de ello se observa un crecimiento muy acelerado en estas estructuras. Hoy en día los sistemas neuronales artificiales fabricados con tecnología VLSI (Very Large Scale Integrated) han acaparado el mercado de la Electrónica.

Durante este crecimiento se han creado expectativas acerca de las futuras aplicaciones que se le darán a los sistemas neuronales dentro de las cuales destacan la visión electrónica, hablar, toma de decisiones, y razonamiento, así como la intervención en los procesadores de señales tales como filtros, los detectores y los sistemas de control de calidad.

En una red neuronal artificial típica, se observa que cada uno de sus niveles se compone de un conjunto de elementos de procesamiento, cada uno de estos elementos reúne valores que provienen de los conectores de entrada, realizando así operaciones matemáticas predeterminadas para finalmente producir un valor de salida.

Las funciones básicas de las redes neuronales artificiales son las siguientes:

- Clasificación: Consiste en que un vector de entrada es transmitido a la red, para el cual se produce una salida que es asociada a una clase representativa definida con anterioridad.

- Igualación de vectores: En esta función al igual que en la anterior, se introduce un vector de entrada, para el cual se produce el vector de salida correspondiente.

- Complementación de vectores: A la entrada de la red se introduce un vector incompleto, obteniendo a la salida un vector complementado con partes que se carecían a la entrada.

[5] En donde las localidades que se encuentran en el enrejado únicamente pueden tomar dos diferentes orientaciones, como pueden ser hacia arriba o hacia abajo. En analogía con una red neuronal biológica, se identifica a cada localidad como una neurona artificial y se asocia la orientación hacia arriba con el estado activo y la orientación hacia abajo con el estado inactivo.

- Eliminación de ruido: Esta función consiste en eliminar el ruido que presenta un vector a la entrada de la red, obteniendo a la salida un vector similar pero libre de ruido.

- Optimización: A los vectores de entrada que presentan problemas en sus valores iniciales, la red proporciona como resultado un conjunto de variables que nos da una solución al problema.

- Control: Teniéndose en la entrada vectores que representan el estado actual de un controlador, así como la respuesta deseada del mismo, la red da como salida una secuencia adecuada de señales las cuales crearán la respuesta correcta.

Entre las características importantes y representativas de las redes neuronales artificiales se encuentran las siguientes:

- Cada elemento de procesamiento que conforma la red actúa de forma independiente.

- Necesariamente los elementos de procesamiento requieren de recibir información, para realizar correctamente su función, además, un elemento no requiere conocer el estado de algún otro elemento.

- El trabajar con un gran número de conexiones trae como consecuencia un valor de salida redundante y elevado, pero también facilita la representación mediante una mejor distribución.

1.2. DEFINICION DE UNA RED NEURONAL ARTIFICIAL

En el campo de los sistemas inteligentes se ha manejado un sin número de definiciones referentes a las redes neuronales artificiales, de entre las cuales se encuentran las siguientes:

> "Las redes neuronales artificiales pueden ser vistas como una caja negra la cual acepta entradas y produce salidas"[6].

Este concepto hace referencia de manera general de lo que es una red neuronal, a la que se considera como un conjunto de elementos que se encuentran alojados en el interior de una caja, la cual recibe información que procesa para finalmente producir salidas.

[6] Patrick K. Simpson *"Foundations of Neural Networks"*

"Una red neuronal artificial es un conjunto de algoritmos que tienen como fin el de resolver tareas cognoscitivas, como lo es el aprendizaje y la optimización".[7]

Este concepto se inclina hacia la relación que existe entre las funciones que realizan las redes neuronales artificiales y el cerebro.

Una de las definiciones más aceptables sobre las redes neuronales artificiales es la que maneja Drew Van Camp:

"Una red neuronal artificial es un modelo de computadora inspirado en la estructura y estudio de las neuronas reales. Como el cerebro, estas redes pueden reconocer patrones, reorganizar datos y lo más interesante, aprender"[8] .

Agregando a todos los conceptos, los investigadores llegan a un punto en común, que las redes neuronales artificiales están hechas de elementos llamados unidades, las cuales representan a los cuerpos de las neuronas biológicas y realizan las funciones correspondientes de los axones y las dendritas; estas unidades se conectan mediante enlazadores provocando el fenómeno sináptico. Semejante a la fuerza de conexión que existe en una sinápsis, el enlazador multiplica la salida de una unidad por un FACTOR DE PESO, para posteriormente pasarlo a otra unidad y sumarlo con los valores provenientes de otros enlazadores.

Un fenómeno importante es el disparo o excitación de una neurona, que análogamente, en las redes neuronales artificiales ocurre cuando el valor total de la suma de los enlazadores de entrada, excede a un valor ya preestablecido, para entonces, los factores de peso que se encuentran en los enlaces sufren modificaciones; y a todo este proceso se le conoce como aprendizaje de la red.

1.3. COMPONENTES DE UNA RED NEURONAL ARTIFICIAL

Las redes neuronales artificiales tienen en su estructura componentes básicos, los cuales son las unidades, pesos, conexiones y elementos de procesamiento, además dependiendo de la aplicación a la que se destina la red, se presenta la necesidad de utilizar otros componentes, como son las funciones preestablecidas, vectores de entrada y vectores de salida.

[7] David S. Touretzky. *"Aduanced in Neural Información Process ing System"* Vol. II.
[8] Drew Van Camp. *"The amateur Scientist"*. Scientific Americanan pag. 125.

Para ilustrar los componentes básicos de una red neuronal artificial, consideremos la fig. 1.

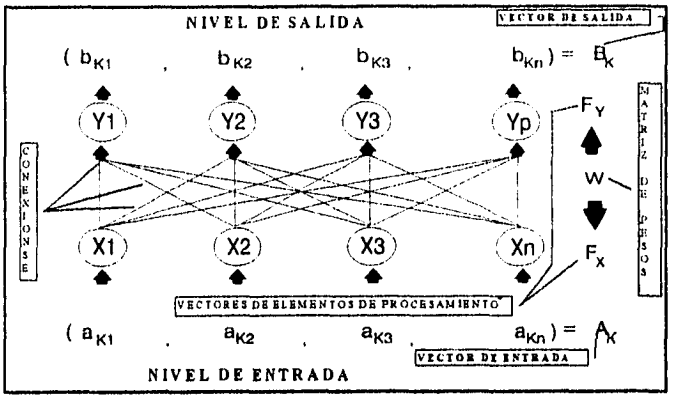


Fig. 1.1. Estructura de una red neuronal artificial de 2 niveles.

1.3.1. Unidades

Existen tres tipos diferentes de unidades, de las cuales pueden estar compuestas las redes neuronales artificiales: las *unidades de entrada* que son los encargados de llevar la información proveniente del exterior al interior de la red. Las *unidades de salida*, cuya función es la de enviar las señales ya procesadas y visibles a la salida. El tercer tipo de unidades no la contemplan todas las redes en su diseño (como es el caso de las redes de uno y dos niveles), estas son las *unidades ocultas*, que tienen como función la de actuar como intermediario, es decir, reciben las señales provenientes de las unidades de entrada, las procesan y finalmente producen un valor dirigido a las unidades de salida.

1.3.2. Pesos

Para las redes neuronales artificiales, los pesos son físicamente magnitudes eléctrica representadas por valores numéricos, que al ser implementados a los patrones de entrada producen un cambio en dicho patrón.

Para modelar una red neuronal artificial implementándola mediante una compuerta EXOR no es particularmente interesante, debido a que seleccionando cuidadosamente los pesos y entradas de la red se puede asemejar cualquier función lógica. Lo que es realmente interesante de tales redes es que no es necesariamente elegir los pesos y las entradas, ya que estos pueden comenzar con cualquier valor.

Posteriormente, la red comienza a mostrar de forma repetida los vectores de entrada y salida, así como los pesos necesarios para implementarlos a la compuerta EXOR; de esta manera la red se encuentra en proceso de aprendizaje. Por lo que, para un gran conjunto de datos, la red es capaz de reconocer vectores que no haya analizado anteriormente.

1.3.3. Conexiones

Las *Conexiones* sirven para enlazar los nodos o elementos de procesamiento, y permiten el flujo de la información en una sola dirección, siendo esta representada por una flecha; de esta forma, el flujo de la información se lleva a cabo mediante los trazos y llegan hasta los *elementos de procesamiento* que es en dónde se realizan las operaciones.

El propósito de las conexiones dentro de una gráfica dirigida, es la de determinar la dirección del flujo de la información, como se muestra en la fig. 1.1 El flujo de la información que se realiza del nivel Fx a el nivel Fy se hacen mediante las conexiones representadas con la letra W. En los diagrafos de las redes neuronales se les asigna un peso a cada trazo (conexión), que representa el valor de la señal de salida que se trasladará de un nodo (elemento de procesamiento) inferior a un nodo adyacente.

Como se aprecia en la fig. 1.1. se utiliza una flecha doble para indicar el sentido del flujo de la información. Una conexión tiene fundamentalmente dos funciones, por un lado la de indicar la dirección, y por el otro, de modular la cantidad de información que fluye entre los elementos de procesamiento. En tanto, las conexiones denominadas excitatorias son las que contienen pesos con valores positivos, y por el contrario, cuando estos pesos tienen un valor negativo se les denomina como conexiones inhibitorias. Cuando se presenta el caso de que no existe conexión alguna se debe a que el peso de esta conexión presenta un valor de cero o nulo.

Para el diseño de redes neuronales artificiales no es conveniente el uso continuo de conexiones que representen el valor de cero, a excepción de que estas conexiones se encuentren esparcidas y en números pequeños. Durante el funcionamiento de una red neuronal artificial, una de las condiciones más deseables es la de lograr que los elementos de procesamiento mantengan valores parciales internos (valores preestablecidos).

1.3.4. Elementos De Procesamiento (P.E.)

Los elementos de procesamiento son los elementos más importantes y básicos que componen a una red neuronal artificial. Esta importancia se debe a que en el interior de estos elementos se ponen en funcionamiento los componentes restantes de la red, para realizarse las operaciones que nos lleven a los resultados esperados.

Uno de los elementos de procesamiento que se usa con mayor frecuencia se muestra en la fig. 1.2.

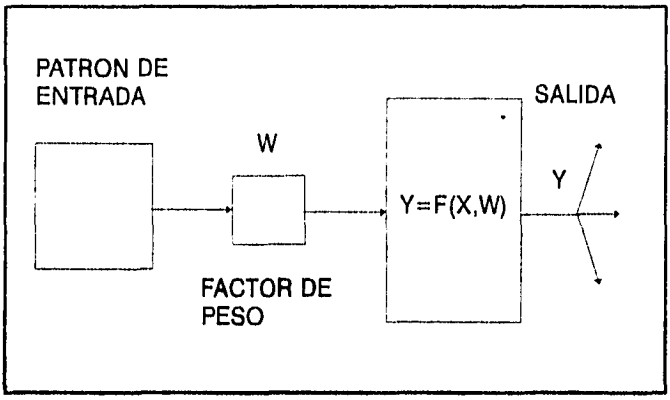


Fig. 1.2. Elementos de Procesamiento con una sola conexión de peso.

Además, los elementos de procesamiento pueden encontrarse en dos situaciones diferentes: la primera de ellas, se presenta cuando un elemento de procesamiento se encuentra en el nivel de entrada y por consiguiente recibirá un sólo patrón de entrada, como se observa en la fig. 1.2. La segunda ocurre cuando cada elemento de procesamiento que se encuentre en un nivel que no sea el de entrada, tiene varias conexiones de peso provenientes de otros elementos de procesamiento de diferente nivel, como se observa en la fig. 1.3.

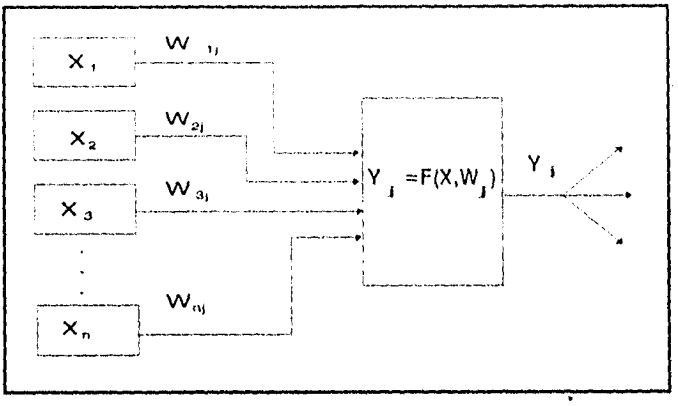


Fig. 1.3. Elementos de Procesamiento con varias conexiones de peso.

Como consecuencia de las situaciones anteriores se tienen que los elementos de procesamiento tienen dos cualidades importantes:

1.- Los elementos de procesamiento necesitan únicamente de información ubicada en las entradas de estos, para posteriormente almacenarlas en el interior de dicho elemento y finalmente producir un valor de salida. Una ventaja muy importante de esta característica es que ya no se requiere de una información extra para un mejor desempeño.

2.- Durante su funcionamiento, los elementos de procesamiento utilizan los valores de entrada para procesarlos y producir un sólo valor de salida, transmitiéndose mediante las conexiones hacia un elemento de procesamiento receptor o bien puede ser utilizado como valor de salida de la red.

Como se hizo anteriormente con las conexiones, el valor que se le asigna a cada elemento de procesamiento así como a la etiqueta que lo representará, serán denotados con una misma variable, por ejemplo, para representar el *j-ésimo* elemento de procesamiento que se localiza en el nivel F_y se utiliza la expresión Y_j; y por consiguiente el valor que tenga este elemento será también de la forma y_j.

El valor de salida de un elemento de procesamiento está en función de las entradas y los pesos que lleguen a él, como se puede observar claramente en la fig. 1.3., en donde *yj* representa la función de las salidas producidas en un nivel precedente, denotada por la ecuación $F_x=X=(X_1,X_2,...,X_n)$ y de los pesos localizados desde el nivel *Fx* hasta el elemento y_j; es decir, $W=(W_1,W_2,...,W_{nj})$; de acuerdo a lo anterior, la ecuación que representa al elemento *yj* es:

$$Y_j = F(X,W_j) \quad \text{..... (5)}$$

1.3.5. Vectores De Entrada Y Salida

Las redes neuronales artificiales están compuestas por gran número de unidades interconectadas de manera que estas a su vez forman vectores que dependiendo de su ubicación o nivel en que se encuentren dentro de la red neuronal, se les denominan como vectores de entrada, vectores ocultos o vectores de salida.

Dependiendo del número de niveles que conformen a la red neuronal, tendrá esta una cantidad proporcional de niveles ocultos y a su vez, un número determinado de vectores de salida, así como los vectores de entrada.

Además de la descripción de cada componente de una red neuronal artificial se dará a conocer su terminología, para ello consideraremos la figura 1.1.

En la fig. 1.1. se observa la estructura de una red neuronal artificial, la cual consta de dos niveles (nivel de entrada y nivel de salida). Dentro de cada nivel se encuentran los elementos de procesamiento que en conjunto forman vectores de entrada y salida, cuya función es la de manejar la información proveniente del nivel de entrada y enviar una respuesta hacia la salida.

La nomenclatura para describir a los vectores de entrada y salida se denotará con las primeras letras del alfabeto (por ejemplo A,B,C,..etc) acompañadas de un subíndice, el cual servirá para indicar la ubicación del vector dentro la red neuronal. De esta manera, el vectores de entrada se representarán por la siguiente ecuación:

$$A_k = (a_{k1},a_{k2},a_{k3},...,a_{kn}) \text{ donde } k = 1,2,3,...,n \quad \text{..... (1)}$$

y el vector de salida por la ecuación:

$$B_k = (b_{k1},b_{k2},b_{k3},...,b_{kn}) \text{ donde } k = 1,2,3,...,n \quad \text{..... (2)}$$

Los elementos de procesamiento que se encuentren en un nivel cualquiera, se representan con la misma variable que le fue asignada a dicho nivel.

1.3.6. Vectores de Elementos de Procesamiento

Por su parte, al conjunto de elementos de procesamiento de un nivel se le conoce como Vector de Elementos de Procesamiento (PE's) al que se le interpreta por cualquier letra intermedia del alfabeto (por ejemplo F,G,H,...etc), aunque en muchos de los casos, el utilizar 3 niveles de elementos de procesamiento, son los suficientes para formar una red neuronal artificial aceptable. De esta manera, los elementos de procesamiento que se encuentran localizados en el nivel de entrada se representarán mediante la siguiente ecuación:

$$F_x = (x_1, x_2, \ldots, x_n) \quad \ldots\ldots (3)$$

De donde cada elemento x_i recibe información de su correspondiente entrada que se denotará como a_{ki}. El siguiente nivel de elementos de procesamiento será indicado como Fy, y así sucesivamente. Volviendo a la fig. 1.1., se observa que el segundo nivel de la red es considerado como el nivel de salida, que se representa con la siguiente ecuación:

$$F_y = (y_1, y_2, \ldots, y_n) \quad \ldots\ldots (4)$$

De donde cada elemento y_j, esta correlacionado con el j-ésimo elemento de B_k.

1.3.7. Matriz De Pesos

Los pesos conectados entre los niveles de la red neuronal, sirven para modificar los valores que circulan en las conexiones, son almacenados en un arreglo matricial, denominado como Matriz de Peso, la cual puede estar representada por las ultimas letras del alfabeto (T,U,V, etc).

Para el caso de la fig. 1.1. que se muestra una red neuronal de dos niveles, tenemos que su respectiva matriz de pesos es como la que se ilustra en la figura 1.4., que sirve para representar las correspondientes conexiones entre los elementos de procesamiento del nivel de entrada señalado como Fx y los del nivel Fy (nivel de salida).

De acuerdo a la matriz de pesos de la figura 1.4. el valor de 1 significa la conexión de los pesos de los elementos de procesamiento de x_1 con el de y_1.

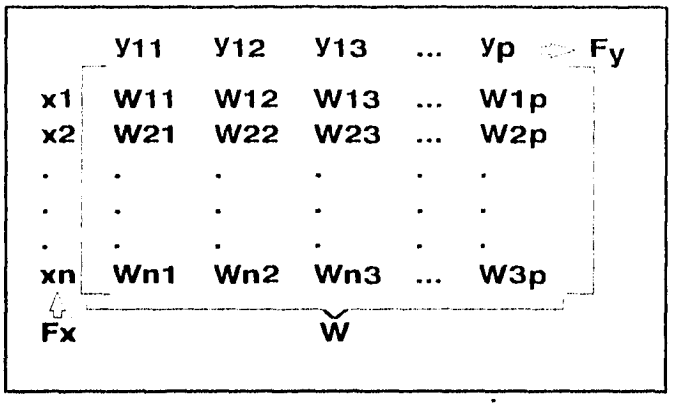


Fig. 1.4. Matriz de peso de una red neuronal de dos niveles

1.4. FUNCIONAMIENTO BASICO DE UNA RED NEURONAL ARTIFICIAL

Como se observa en la fig. 1.5. el proceso de sinápsis se presenta mediante un peso que puede ser modificado, el cual es asociado con cada una de las conexiones.

La mayoría de los modelos de redes neuronales artificiales presentan dos desventajas importantes y básicas: por un lado, no tienen la capacidad de proporcionar la geometría más acertada del funcionamiento de las dendritas y los axones, y por el otro, el de asemejar o interpretar la salida eléctrica, como la que se produce en una neurona biológica, como un simple valor numérico que presenta la magnitud de disparo.

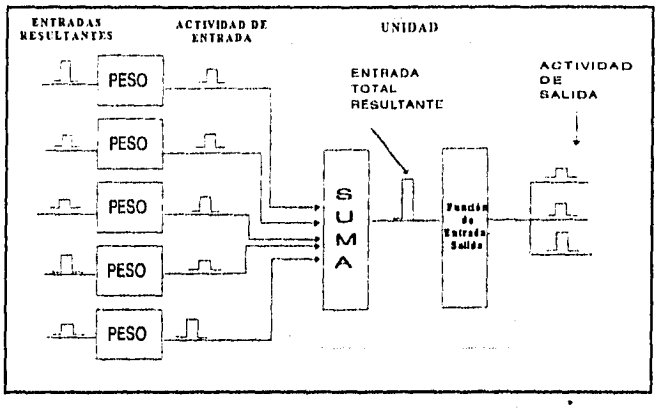


Fig. 1.5. Proceso de sipnásis en una red neuronal artificial.

De la fig. 1.5. se puede observar que cada unidad transforma una actividad proveniente del exterior en un vector de entrada, siendo este a su vez transmitido a otras unidades; este proceso se lleva a cabo durante dos fases: primero, el vector de entrada es multiplicado por un factor de peso colocado sobre su conector, dando origen a un valor denominado *Entrada Resultante,* posteriormente se suma simultáneamente con las otras entradas, formándose así, un valor denominado *entrada total resultante;* segundo, una unidad hace uso de una función de entrada salida con el fin de transformar el valor de entrada total resultante a un valor de salida, llamado *actividad de salida.*

La función entrada-salida y los pesos que se manejan en una red neuronal en perfecta combinación, determina el desempeño que logre la red. Por su parte, la función entrada-salida puede encontrarse en cualquiera de las siguientes tres categorías: *Lineal, Umbral o Sigmoidal (en función de sigma).*

En caso de que la función se encontrase en la categoría de *unidades lineales,* tendríamos que la actividad de salida de la red neuronal será proporcional al total de las entradas. Para el segundo caso, el de las unidades umbrales, la salida únicamente puede tomar uno de dos niveles, el que será determinado dependiendo si el valor de la entrada total resultante sea mayor o menor que alguno de los valores preestablecidos. Por último para el caso de las unidades de tipo sigmoidal, el valor de salida, constantemente se encuentra en estado de variación y esto trae como consecuencia ciertos cambios en la entrada.

De las tres categorías anteriores, la que se considera más próxima al comportamiento de las neuronas biológicas, es la de las unidades sigmoidales, aunque las tres son consideradas únicamente como aproximaciones al comportamiento real.

La clase de red neuronal artificial que con mayor frecuencia se diseña y se construye es la que se estructura de tres grupos o niveles, que son interconectados de la siguiente manera:

El nivel que contiene a las unidades de entrada se conecta al nivel de las unidades ocultas, siendo este a su vez conectado con el nivel formado con las unidades de salida. Por su parte el funcionamiento de las unidades de entrada consiste en suministrar la información del exterior de realimentación más reciente en tanto, la función de las unidades ocultas se va componiendo de acuerdo a las actividades que realizan las unidades de entrada en combinación con los pesos que han sido colocados sobre las conexiones que se encuentran entre las unidades de entrada y las unidades ocultas.

De una forma muy similar, el comportamiento de las unidades de salida se deriva tanto de las actividades ocultas como de los pesos colocados sobre las conexiones entre ambas unidades. Este tipo de red además de ser uno de los más sencillos, es uno de los más interesantes debido a que las unidades ocultas tienen la capacidad de construir sus propias interpretaciones acerca de las actividades de entrada, de esta manera, mientras los pesos sufran ciertas modificaciones, darán otras interpretaciones diferentes sobre las actividades de entrada.

1.5. METODOS PARA DISEÑO DE ELEMENTOS DE PROCESAMIENTO

A continuación se mencionan los métodos más utilizados para el diseño de elementos de procesamiento.

FUNCION	ECUACION	COMENTARIOS
POR COMBINACION LINEAL	$Y_j = f\left[\sum_{i=0}^{n} x_i w_{ij}\right] = f(xw_j)$ donde $w_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$	Este método es de los más usados en computación y también es conocido como *producto punto*.

Tabla 1.1. Función por Combinación lineal.

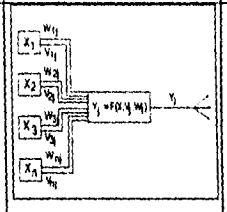
FUNCION	FIGURA	DESCRIPCION	ECUACION
CON CONEXIONES DOBLES (media-varianza).		Esta función es utilizada cuando existen dos conexiones a un mismo elemento. Las cuales representan al valor de la clase y la varianza en dicha clase.	$Y_j = g\left[\sum_{i=0}^{n}\left(\frac{W_{ij} - X_j}{V_{ij}}\right)^2\right]$ donde $g(x) = \exp\left(\frac{-x^2}{2}\right)$

Tabla 1.2. Función con conexiones dobles.

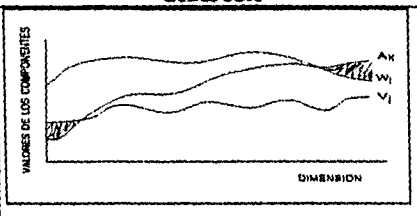
FUNCION	DESCRIPCION	GRAFICA
CON CONEXIONES MIN/MAX	Son usadas con menor frecuencia. Aunque utiliza conexiones dobles, el objetivo de estas funciones, es la de asignar como límite inferior de la clase a uno de los vectores pertenecientes al conjunto V_j, y un límite máximo, a uno de los vectores W_j. Y los patrones que se encuentren dentro de estos valores producirán un valor conocido como *valor de Activación Min/Max.*	 **Gráfica de los puntos mínimos y máximos**

Tabla 1.3. Función con conexiones mínimas / máximas

1.6. FUNCIONES P.E.

Existe una gran variedad de funciones P.E., cuyo número posiblemente sea infinito, pero básicamente son cinco las funciones más utilizadas en las redes neuronales artificiales, las cuales son:

1) Función P.E. lineal
2) Función P.E. Escalón
3) Función P.E. Rampa
4) Función P.E. Sigmoidal
5) Función P.E. Gaussiana

De las funciones anteriores, a excepción de la primera, todas presentan un comportamiento no lineal durante el desempeño de la red neuronal, dependiendo del grado de limitación que se les de a los valores de salida, por medio de un rango preestablecido. A continuación, se dará una breve descripción de estas funciones:

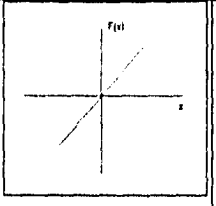
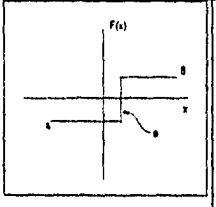
FUNCION	REPRESENTACION GRAFICA	ECUACION	COMENTARIOS
LINEAL		$f(x) = \alpha x$ donde $x \in \Re$ = valor escalar positivo	Se obtiene a la salida una señal modulada linealmente, producida por la función P.E. Lineal
ESCALON		$f(x) = \begin{cases} \beta \text{ si } x \geq \theta \\ \beta \text{ si } x < \theta \end{cases}$	Estas funciones producen salidas propias de un sistema binario, es decir, en respuesta a las señales de entrada, envían un 1 si el valor x es positivo. En términos binarios queda de la siguiente forma: $f(x) = \begin{cases} 1 \text{ si } x \geq 0 \\ 0 \text{ otro valor} \end{cases}$

Tabla 1.4.

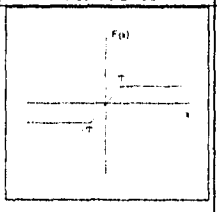
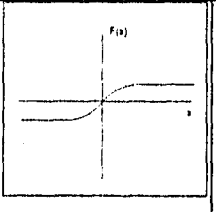
FUNCION	REPRESENTACION GRAFICA	ECUACION	COMENTARIOS
RAMPA		$f(x)=\begin{cases}\tau & \text{si } x \geq \tau \\ x & \text{si } \lvert x\rvert < \tau \\ -\tau & \text{si } x \leq -\tau\end{cases}$	Esta función es una consecuencia de combinar la función P.E. Lineal y la función P.E. Escalón. Esta función tiene un límite inferior y un límite superior
SIGMOIDAL		$f(x)=\frac{1}{1+e^{-ax}}$ Función logística En tanto, la función sigmoidal logística tiene otras dos alternativas, la primera, es la de Tangente Hiperbólica representada por: $f(x)=\tan h(x)$ y la relación aumentada de los cuadrados dada por la expresión: $f(x)=\begin{cases}\frac{x^2}{1+x^2} & \text{si } x>0 \\ 0 & \text{otro valo}\end{cases}$	Llamada así por tener forma de *S*, es semejante a la función RAMPA, a excepción de que la segunda tiene puntos de discontinuidad y la primera es una función continua, además es ilimitada, monótona, no decreciente, y proporciona una respuesta no lineal dentro de un rango preestablecido. Dentro de las funciones sigmoidales existe una que es aplicada con más frecuencia, la cual es conocida como *Función Logística.*

Tabla 1.5.

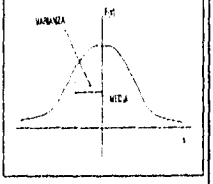
FUNCION	REPRESENTACION GRAFICA	ECUACION	COMENTARIOS
GAUSSIANA		$f(x) = \exp\left(\frac{-x^2}{v}\right)$ de donde x representa la media, y v la varianza preestablecida.	Esta función se puede considerar como radial debido a que tiene una simetría en los puntos cercanos al origen. Para el correcto desempeño de la función P.E. Gaussiana se requiere de la utilización de la varianza *V* que debe ser mayor que 0, la cual sirve para formar la función Gaussiana. En algunos casos, en las redes neuronales artificiales la función Gaussiana se utiliza en combinación con un conjunto de conexiones dobles, y en otros casos, la varianza esta preestablecida.

Tabla 1.6.

En resumen, una regla que hay que seguir cuidadosamente para tener el éxito en cualquier aplicación que se le destine a la red neuronal artificial con la que se vaya a trabajar, es la de tener una correcta visión de las características en las que se basará nuestra red, para tener la selección más adecuada de los elementos que corresponderán a la misma.

1.7. REDES NEURONALES AUTOASOCIATIVAS Y HETEROASOCIATIVAS

Una situación muy interesante, es que no todas las redes neuronales artificiales operan con algunos tipos comunes de vectores, es decir, algunos sistemas neuronales requieren para su operación de patrones simples, a estos sistemas se les conoce con el nombre de *Red Neuronal Artificial Autoasociativa;* pero también existen los sistemas denominados *Redes Neuronales Artificiales Heteroasociativos,* debido a que utilizan para su funcionamiento patrones pares.

Para las redes del tipo *autoasociativa* tenemos a un conjunto de patrones que se encuentran previamente almacenados en la red, los cuales son comparados con el patrón de entrada y si alguno es similar, entonces ambos patrones se asocian, en la fig. 1.6. se muestra este tipo de red, en la cual, un cuadrado incompleto, es asociado con el cuadro que se encuentra en el interior de la red.

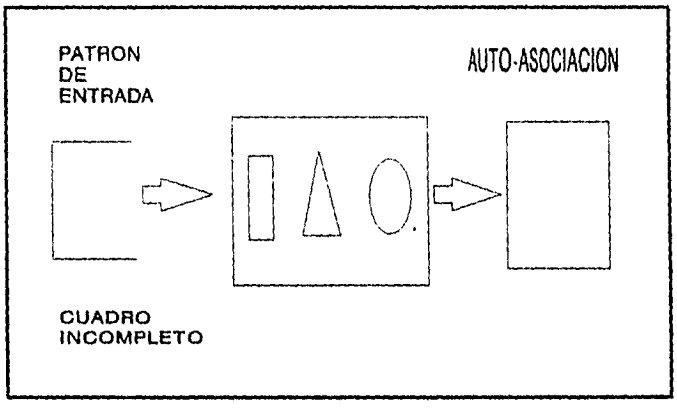


Fig. 1.6. Red neuronal artificial autoasociativa.

En las redes heteroasociativas, los patrones que se encuentran en su interior, están agrupados en pares y de igual manera que en la red autoasociativa, se compara la entrada con los patrones almacenados, asociándole el par (romboide) que le fue asignado al patrón que se le asemeja (cuadrado). Esto se observa en la siguiente fig. 1.7.

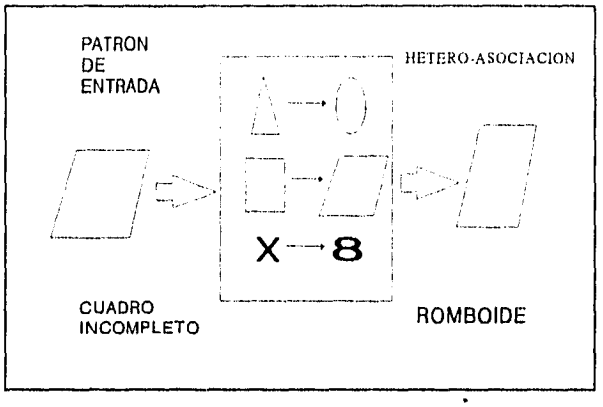


Fig. 1.7. Red neuronal artificial Heteroasociativa.

Debido al tipo de procesamiento que realice la red neuronal, existen básicamente dos tipos de redes que son: las que únicamente tienen la capacidad de procesar datos en código binario; este tipo de máquinas se pueden ver tanto en el *Modelo de la Red Neuronal Artificial de Hopfield*[9] perfeccionado por Amari en 1972, como en el modelo llamado *El Cerebro como una simple caja de Anderson*[10].

Y las que tienen la característica de procesar datos con valores reales, como es el caso del *Método de Realimentación hacia atrás de Williams*[11] *y del vector de cuantización para el aprendizaje de Kohonen*[12], cuyo propósito es el de producir un código obtenido de un patrón de entrada de *n* dimensiones, el cual es transmitido a través de un canal y utilizado para reconstruir la entrada original con un mínimo de distorsión.

[9] J.J. Hopfield. *"Neural Network and Physical Systems with Emergent Collective Computational Abilities"*. Proc. Natl. Acad. Sci., Vol. 79. Apr., 1982.
[10] Anderson, J.J. Silverstein, S. Ritz, and R. Jones. *"Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model"*. Physch. Rev., Vol. 84, 1977.
[11] Jacek M. Zurada. *"Introduction to Artificial Neural System"*. West Publishing Company. USA, 1992.
[12] Kohonen, T. *"Learning Vector Quantization for Pattern Recognition"*. Helsinki University of Tecnology, 1986.

1.8. TOPOLOGIAS

Las redes neuronales artificiales se componen de niveles que a su vez contienen elementos de procesamiento interconectados mediante pesos de conexión. Una topología es la forma en que son conectados dentro de la red estos elementos en combinación con los patrones de entrada.

1.8.1 Niveles

Contienen a los elementos de procesamiento (PE's), los cuales tienen dos propiedades importantes:

1) Las conexiones que proporcionan la información a los elementos de procesamiento de un mismo nivel, provienen de un mismo origen, como se puede apreciar en la fig. 1.8., en la que los elementos de procesamiento del nivel marcado como Fx, reciben la información proveniente de los patrones de entrada, y de la misma forma los elementos de procesamiento del nivel Fy reciben información de los elementos de procesamiento del nivel Fx.

2) Los elementos de procesamiento pertenecientes a un mismo nivel manejan la misma información debido a que utilizan tanto la misma función PE como el tipo de conexiones como se muestra en la fig. 1.8.

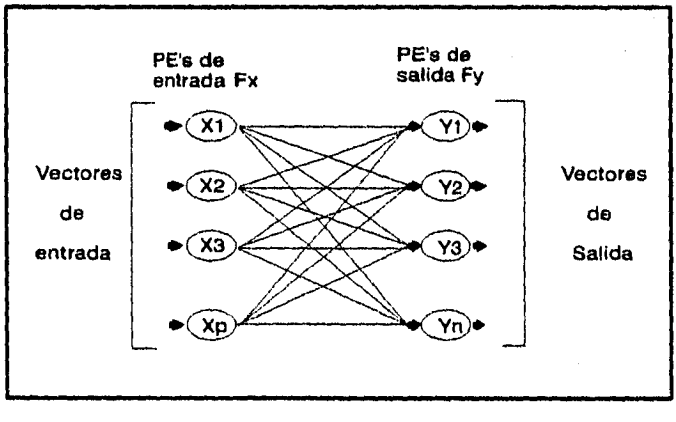


Fig. 1.8. Red Neuronal de 2 niveles.

Existe dos tipos de interconexiones que emplean las redes neuronales artificiales que son: *Intrainveles.*-(del latín intra *"dentro"*) las cuales conectan los elementos de procesamiento de un mismo nivel; y las conexiones *interniveles* (del latín inter *"entre"*) conectan elementos de procesamiento de diferentes niveles.

1.8.2. Redes de Alimentación hacia adelante y hacia atrás

Las redes neuronales artificiales que tiene alimentación hacia adelante son las que poseen conexiones para trasladar la información en una sola dirección, es decir, de entrada a salida, y que no poseen lazos de retroalimentación[13] (fig. 1.9.(a)). Por el contrario si durante la trayectoria de la información en una red neuronal existe alguna retroalimentación conectada a uno de los niveles anteriorés o a un elemento de procesamiento en particular, entonces se trata de una red de alimentación hacia atrás (fig. 1.9.(b)).

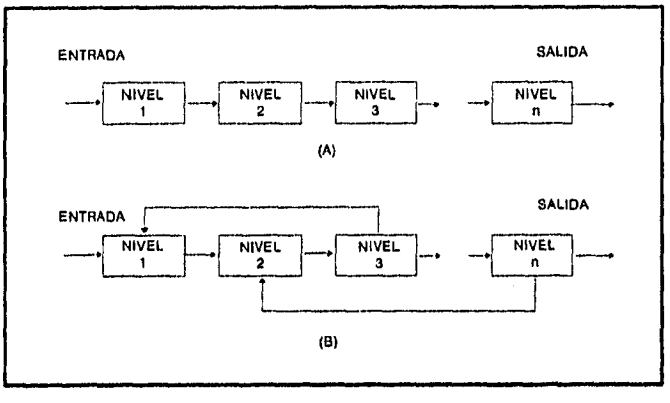


Fig. 1.9. **(a) Red de alimentación hacia adelante**
(b) Red de alimentación hacia atrás

[13] Jacek M. Zurada. *"Introduction to Artificial Neural System"*. West Publishing Company. USA, 1992.

1.8.3. Topologías Básicas

Tres de las topologías básicas de las redes neuronales artificiales, son el *Instar, el Outstar*[14] y la *Adaline.*

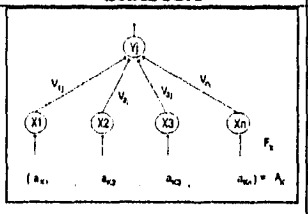
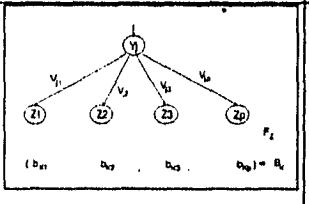
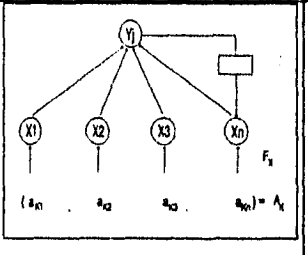
TIPO	ECUACION	GRAFICA	OBSERVACION
Instar	$v = \frac{da_i}{\sum_{i=1}^{n} da_i}$ donde $i = 1,2,3,\ldots,n$		Se le considera como red de patrón codificado[15].
Outstar	$Z_i = Y_i W_{ij}$ donde i = 1,2,3,...,n		Este tipo de red puede ser utilizado como complemento para la red Instar ya que el valor de salida Y_j producido por esta red se puede utilizar como valor de entrada Y_j para la red outstar.
Adaline[16] (Adaptive Linear Neuron "Neurona lineal Adaptiva")			Tienen muchas características topológicas semejantes con el Instar , a excepción de que en la Adaline los pesos representados por V_j tienen la propiedad de ser modificables.

Tabla 1.7.

[14] Grossberg, S. *"Studies of Mind and Brain"*. Boston: Reidel, 1982.

[15] Un patrón codificado es el que se obtiene utilizando los valores de los patrones representados por la expresión $A_k=(a_{k1},a_{k2},...a_{kn})$ y normalizar cada uno de estos valores para finalmente interpretarlos y utilizarlos como los pesos de conexión, representados por $W_j=(w_{1j},w_{2j},...wnj)$.

[16] Widrow, B. and M. Hoff. *"Adaptive Switching Circuits"*. in WESCON Convention Records, Vol. 4, 1960.

1.8.4. Red de un Nivel

El cual está formado por un conjunto de intraconexiones de sus elementos de procesamiento, como se observa en la fig. 1.10.

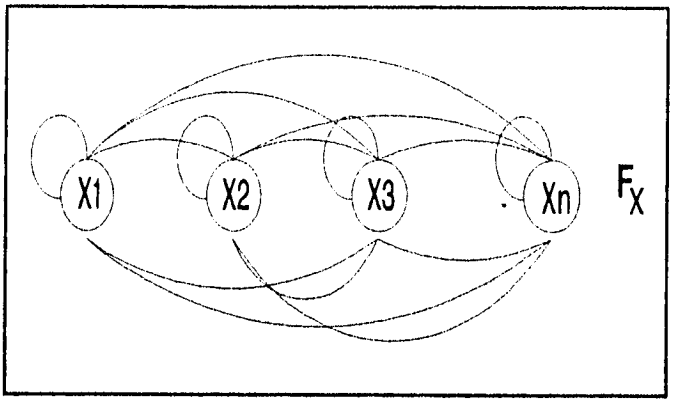


Fig. 1.10. Red neuronal de un sólo nivel.

De acuerdo a la fig. 1.10. tenemos que este tipo de redes consisten de *"n"* elementos de procesamiento en un nivel *Fx*, en donde además de que las conexiones de cada elemento de procesamiento se hacen con los elementos restantes, pueden reconectarse así mismos, dando origen a una matriz de conexiones con *n* entradas, por los que tiene la capacidad de aceptar un patrón de entrada de n^2 dimensiones.

Entre los tipos de procesamiento de patrones que llevan a cabo las redes neuronales de un sólo nivel se encuentran las siguientes:

1.8.4.1. Complementación de Patrones.

Las redes neuronales artificiales inicialmente se les presenta a la entrada una parte del patrón a analizar, asumiendo que en base a esta porción, la red sea capaz de complementar al patrón con las partes restantes y así obtener a la salida el patrón completo, como se muestra en la fig. 1.11, en donde a la red se le aplica a la entrada una parte o la mitad de la imagen, obteniendo a la salida la imagen completa, siendo que previamente la red durante su proceso busca las partes faltantes y los complementa a la imagen.

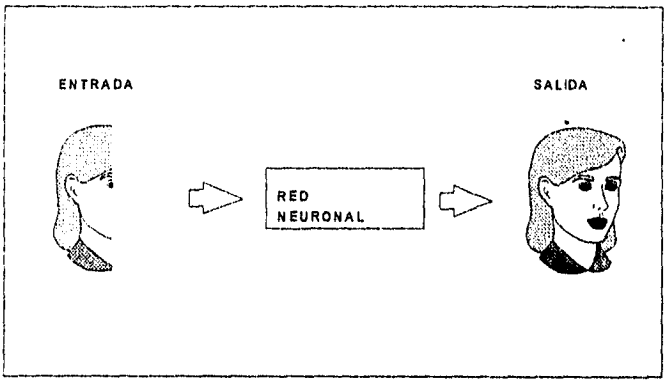


Fig. 1.11. Proceso de complementación de patrones.

1.8.4.2. Eliminación de Ruido.

Es similar al anterior, solo que aquí se desea una respuesta libre de todo ruido, la diferencia entre estos dos tipos de procesos esta básicamente en el modo de operación, fig. 1.12.

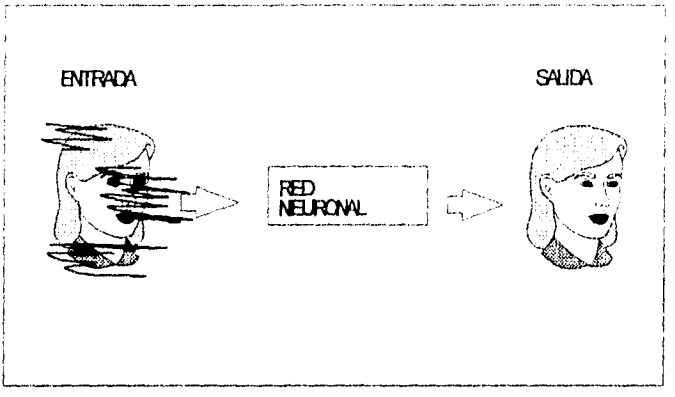


Fig. 1.12. Proceso de Eliminación de ruido.

1.8.4.3. Técnica de Optimización[17]

Consiste en una técnica para resolver problemas representados mediante una ecuación matemática, la cual debe ser maximizada o minimizada, según sea el caso. En este tipo de comportamiento, al problema se le interpreta como una función de energía , la cual describe el comportamiento del sistema neuronal, por lo cual, si la función de energía desciende a un valor mínimo, la red buscara siempre mantenerla en un valor estable para poder encontrar la solución. Para esto las entradas a la red neuronal representan el estado inicial del problema y los valores finales que produzcan los elementos de procesamiento serán los parámetros de la solución, fig. 1.13.

[17] Hopfield J. and Tank. *"Neural Computation of Decision in Optimization Problems"*. Biol. Cybernet., Vol. 52. 1985.

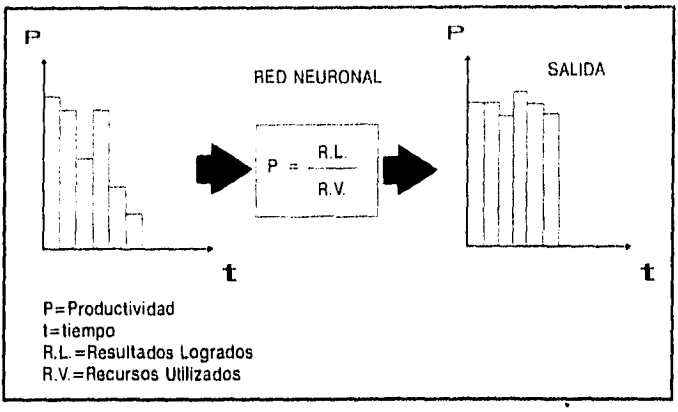


Fig. 1.13. Proceso de optimización neuronal.

1.8.4.4. Mejoramiento de Contrastes.

Se caracteriza por utilizar conexiones denominadas "encendido-centro/apagado-alrededor". Las conexiones encendido-centro, conocidos también como autoconexiones positivas representados por $(w_{ij}=(0)$ *para* $i=1,2,...n)$, son las que permiten que el valor de activación de un patrón vaya incrementándose por si mismo. Por su parte, las conexiones apagado-alrededor llamadas también conexiones contiguas negativas, expresadas por $(w_{ij}=(0)$ *para* $i=j)$, compiten con las conexiones encendido-centro.

Si los elementos de procesamiento del nivel *Fx* son conectados con algunos de los elementos de procesamiento contiguos, se obtendrá una competencia local que dará como resultado valores de activación de diferentes tamaños, como se observa en la fig. 1.14.

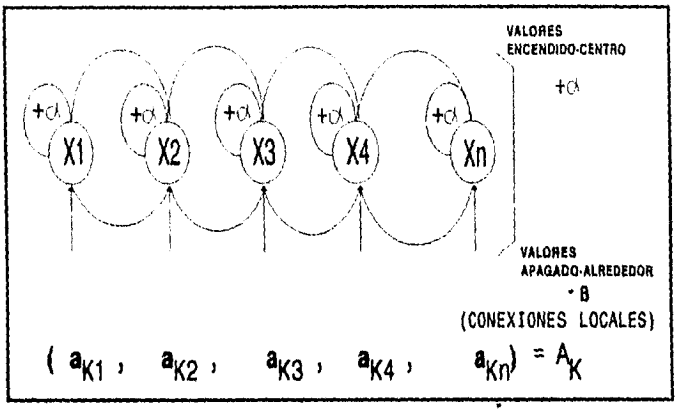


Fig. 1.14. Proceso de mejoramiento de contrastes con conexiones locales.

Si las conexiones apagado-alrededor se interconectan alrededor del nivel *Fx*, como se ilustra en la fig. 1.15., habrá una competencia global y solo habrá un ganador.

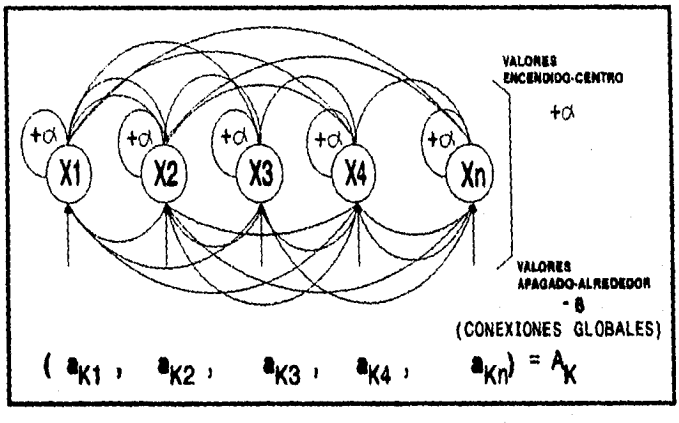


Figura 1.15. Proceso de mejoramiento de contrastes con conexiones globales.

1.8.5. Red de Dos Niveles

Se conforman de la siguiente manera : un nivel Fx formado por *n* elementos de procesamiento, y un nivel *Fy* de *"p"* elementos de procesamientos interconectados en su totalidad, fig. 1.16., dando origen así a una matriz de peso w de dimensión *n x p*, en la cual las entradas *wij* representan los pesos de las conexiones del *i-ésimo* elemento del nivel *Fx(x)* hasta el *j-ésimo* elemento de procesamiento del nivel *Fy(yj)*.

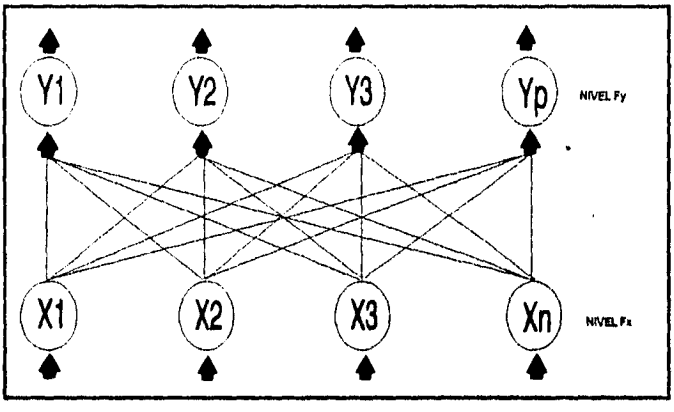


Fig. 1.16. Red de 2 niveles.

A continuación se ilustran tres tipos de redes neuronales de dos niveles:

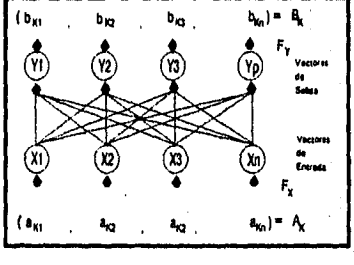
TIPO	GRAFICA	OBSERVACIONES
Igualación de patrones con alimentación hacia adelante.		Este tipo de red neuronal asigna el patrón de entrada A_k a su correspondiente patrón de salida B_k para k=1,2,...,n como se muestra en la figura. Acepta el patrón de entrada A_k produciendo así un patrón de salida Fy=($y_1,y_2,..y_p$).

Tabla 1.8.

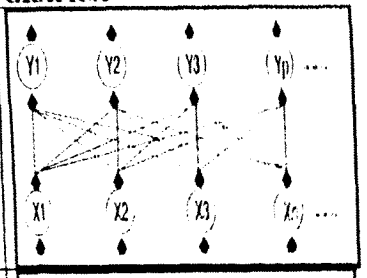
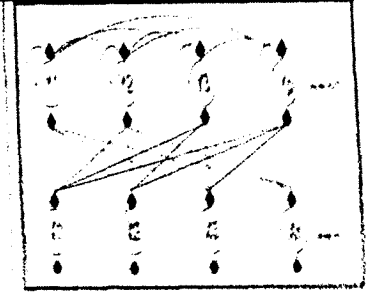
TIPO	GRAFICA	OBSERVACIONES
Igualación de patrones con alimentación hacia atrás.		En la fig se observa que se aceptan entradas de cualquiera de los dos niveles de la red, es decir, del nivel F_x o del nivel F_y, para finalmente producir una salida que se enviará a alguno de los dos niveles, según sea el caso.
Clasificacion de patrones con alimentación hacia adelante		En este tipo de redes neuronales artificiales se crean tres tipos de clases para lo cual se emplea un patrón de entrada X_p y cualquiera de las 3 clases, para lo cual se observa en la figura 3 a una persona se asignara a esta clase cuyo [illegible]

Tabla 1.1

Varias de las redes neuronales se pueden clasificar dentro de una o más de las siguientes categorías: [illegible]

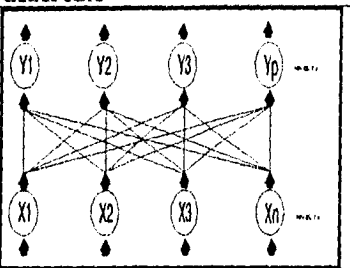
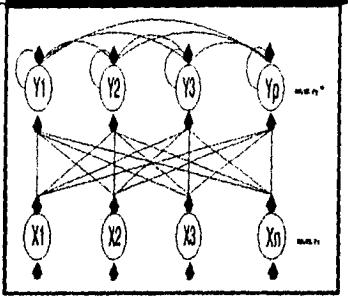
TIPO	GRAFICA	OBSERVACIONES
Igualación de patrones con alimentación hacia atrás.		En la fig. se observa que se aceptan entradas de cualquiera de los dos niveles de la red, es decir, del nivel F_x o del nivel F_y, para finalmente producir una salida que se enviará a alguno de los dos niveles, según sea el caso.
Clasificación de patrones con alimentación hacia adelante.		En este tipo de redes neuronales artificiales se crean una serie de clases para lo cual se asigna un patrón de entrada A_k a cualquiera de las p clases, como se observa en la figura. En este proceso se interpreta a cada clase como a un elemento de procesamiento del nivel Fy independiente a las demás, esto trae como consecuencia la reducción de la tarea para clasificar a los patrones y seleccionar el elemento de procesamiento del nivel Fy que mejor responda al patrón de entrada.

Tabla 1.9.

Muchos de los sistemas cuya función es la de clasificar patrones utilizan la competencia entre las conexiones encendido-centro y apagado-alrededor para llevar a cabo esta tarea.

1.8.6. Red Multinivel

Las redes neuronales artificiales que en su estructura tienen más de dos niveles, son consideradas como *redes multiniveles* como se observa en la fig. 1.17. Tenemos un modelo de red multinivel en la cual existe un nivel con elementos de procesamiento de entrada marcado como *Fx*, también tiene *L* niveles ocultos de elementos de procesamiento representados por $F_y(y_1,y_2,...y_l)$ y finalmente un nivel de salida *Fz*. A los niveles *Fy* se les llama ocultos debido a que intervienen en las conexiones entre los niveles de entrada y salida, es decir, los elementos de procesamiento del nivel de entrada F_x son conectados a los elementos del nivel oculto F_{y1}, éste a su vez es conectado al nivel oculto F_{y2} y así sucesivamente, hasta que el último nivel oculto F_{yL} es conectado con el nivel de salida F_z.

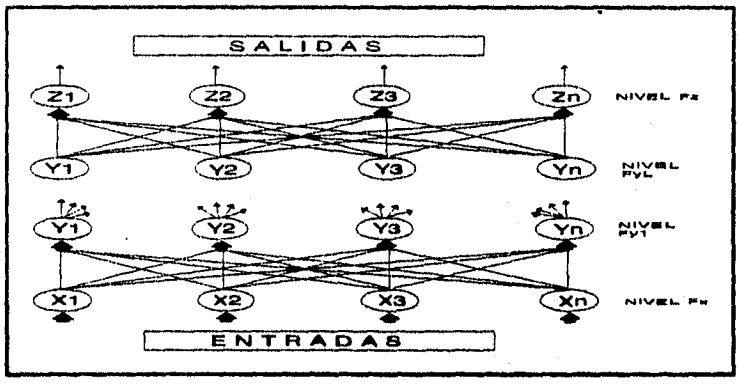


Fig. 1.17. Red de tipo multinivel.

Aunque las conexiones que se presentan en la fig. 1.17, se realizan entre niveles contiguos, se puede dar el caso en que las conexiones de un nivel pasen por encima de alguno o algunos niveles como puede ser cuando el nivel de entrada es conectado directamente con el nivel de salida; pero también puede suceder que algunas conexiones se realicen entre elementos de procesamientos de un mismo nivel.

El mecanismo que permite a las redes neuronales multiniveles realizar rastreos completos consiste en particionar en regiones el área que se le asigna a la entrada para posteriormente ir realizando rastreos de patrones de cada región particionada sucesivamente, para así producir eventualmente respuestas de salida. Debido a este mecanismo ahora las redes neuronales tienen la capacidad de resolver problemas en donde intervengan momentos de decisión muy complejos.

Se ha comprobado que el empleo de tan sólo tres niveles son los suficiente para resolver cualquier problema donde se necesite de rastreos de patrones no lineales, es decir, para cualquier grado de dificultad que se presente, tan sólo basta un nivel oculto. Algunas de las técnicas más comunes utilizadas en las redes neuronales multinivel, donde intervienen los pesos de conexión modificables para realizar rastreos de patrones lineales o no lineales, son: el Algoritmo de Backpropagación[18] y el modelo Neocognitrón[19].

1.8.7. Redes Enlazadas Aleatoriamente

La conexión de los pesos se realiza de forma aleatoria, dentro de un rango especifico. Cuando un peso de conexión tiene valor de cero equivale a decir que no hay presencia de conexión; debido a este tipo de conexiones aleatorias de valores binarios, las redes se simplifican en cuanto a número de conexiones. Este tipo de redes son frecuentemente utilizadas de las siguientes tres formas:

1. Pesos iniciales. En este caso los valores iniciales de conexión que son presentados a la entrada de la red son previamente elegidos al azar siempre y cuando se encuentren dentro de un rango preestablecido.

2. Procesamiento de Patrones. En los dos primeros niveles de una red neuronal multinivel se colocan un conjunto de conexiones de valor binario aleatorio para simular la operación de un patrón preprocesador; este patrón es utilizado la aumentar la dimensión del área que ha sido utilizada para rastrear patrones, este incremento se utiliza para mejorar dicho rastreo.

3."Conocimiento" como causa de la aleatoriedad. Las primeras investigaciones realizadas sobre las redes neuronales artificiales se dedicaron en gran parte al estudio de análisis de sistemas de valores binarios conectados aleatoriamente.

[18] Rosemblatt, F. *"Principies of Neurodynamics"*. Washington, D.C.: Spartan Books, 1962.
[19] Fukushima, K. *"Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition"*. Neural Networks, Vol. 1, 1988.

CAPITULO II

RECONOCIMIENTO DE PATRONES

2.1. PROPIEDADES DE LAS REDES NEURONALES

En esta sección se describirán brevemente las propiedades de las redes neuronales, las cuales son: bases matemáticas, paralelismo inherente, almacenamiento de conocimiento, tolerancia de falla, adaptabilidad y prácticas de reconocimiento de patrones (Ver Fig. 2.1.). También consideraremos algunas de las limitaciones de las redes neuronales.

- Bases matemáticas
- Paralelismo Inherente
- Memoria asociativa distribuida
- Tolerancia de falla
- Adaptabilidad
- Reconocimiento de patrones
- Intuición y reconocimiento de patrones estáticos

Fig. 2.1. Las redes neuronales son notadas por...

2.1.1. Bases Matemáticas

Las Redes neuronales son unas de las pocas tecnologías relacionadas con la Inteligencia Artificial que tienen un riguroso fundamento matemático. Esto provee una medida de seguridad para la comunidad científica en general y permite a los matemáticos algo de deporte. En realidad, esto era en cierto modo el espacio matemático y los antepininos estáticos de esta tecnología que atrajo la atención de la Academia Nacional de Ciencias cuando Hopfield presentó su trabajo sobre Redes Neuronales en 1892. En contraste, los sistemas expertos se basan excesivamente en la heurística, o modo empírico, los cuales son mucho menos formales.

Algunas de las matemáticas son enteramente sofisticadas, usando ecuaciones diferenciales, álgebra lineal, y matrices covariantes. Los Pesos, sumas y funciones de transferencia, y algoritmos de aprendizaje todo cuenta excesivamente sobre las matemáticas. Estas ecuaciones afectan entradas, memoria, determinación de niveles de energía, convergencia y estabilidad. Aunque las matemáticas son íntegramente relacionadas al diseño, funcionamiento y afinación de las redes neuronales, no es necesario ser un matemático para entender los principios básicos de operación.

2.1.2. Paralelismo Inherente

No están las redes neuronales estructuradas en forma paralela únicamente, pero la secuencia de procesamiento es en paralelo y simultáneo. El esquema total del sistema, así como los elementos individuales combinan teoría y practica durante el desempeño en paralelo.

Los elementos de procesamiento en una capa están operando todos en común acuerdo. La Computación es distribuida sobre más de un elemento de procesamiento y es hecha simultáneamente. Aunque las computadoras digitales tienden a simular este paralelismo, el hardware de las redes neuronales realmente desempeñará las operaciones en paralelo. Muchas decisiones rápidas serán posibles y serán disponibles para ser realizadas en tiempo real.

2.1.3. Almacenamiento de Conocimiento

El conocimiento dentro de una red neuronal no es almacenado en locaciones especificas de memoria, como es en la computación convencional. El conocimiento es distribuido por todo el sistema, esta es la respuesta dinámica a las entradas y la arquitectura de la red. Porque el conocimiento es distribuido, el sistema usa muchas conexiones para recuperar soluciones a problemas particulares.

- No es almacenado en locaciones específicas de memoria.
- Es relacionado a la estructura de red.
- Consiste de el estado global de la red en algunas condiciones de equilibrio.

Fig. 2.2. Conocimiento en un red neural

No sólo es la memoria en una red neuronal distribuida, es también asociativa. "Una memoria asociativa es un mapeo de dato a dato, una abstracción matemática de la estructura asociativa familiar del aprendizaje humano y animal". Un sistema de red neuronal observa por la mas cercana pareja, tanto como nuestro cerebro localiza el contenido de memoria para emparejar una entrada estímulo en lugar de un esquema de direccionamiento. Por ejemplo, si usted susurra un poco de una tonada de una melodía, y es una que nosotros hemos escuchado antes, nosotros podemos estar seguros para "nombrar esa melodía". En otra ilustración más personal: Un maestro de escuela secundaria puede llegar a ser muy bueno en la lectura de escritura virtual ilegible de algunos de sus estudiantes. A pesar de las variaciones, distorsiones y omisiones, el humano es bueno para recuperar el patrón correcto. Así de esta forma son las redes neuronales.

Compare este acercamiento con la computadora digital convencional usando RAM y ROM. Cada factor es almacenado en una locación única. Un factor es cargado para proveer las direcciones, aunque no haya particular relación entre el factor y la dirección. Las redes neuronales no trabajan de esta forma (Ver Fig. 2.2). Las computadoras que cargan datos en una manera asociativa, en vez de direcciones, están ahora comenzado a ser de un modo factiblemente comercial.

La memoria asociativa tiene dos beneficios: la habilidad para almacenar un gran número de patrones complicados (tales como lenguaje, escenas visuales, movimientos de robot, comportamiento espaciotemporal, comportamiento social), y la habilidad para clasificar

nuevos patrones para almacenarlos rápidamente. Dentro de las computadoras convencionales, para comparación, sobresalen las computadoras seriales de alta velocidad, ellas hacen pobremente en un tiempo real el reconocimiento de patrones.

La categoría especial de redes neuronales de reconocimiento de patrones que combina el rellamado automático asociativo, es algunas veces llamado memoria de contenido direccionable (CAM). Aquí, la memoria es realmente una representación de la información contenida. Preparando el marco y palabras dato que pueden ser emparejadas exactamente acorde a los contenidos de la palabra, así respuestas múltiples son posibles para cada tecla de búsqueda. La más general "memoria asociativa" puede accesar o modificar celdas basadas en sus contenidos, pero no cuenta con un exacto emparejamiento con una tecla dato.

El conocimiento es también relacionado a la estructura de la red -como las señales de salida de un nodo es conectado a la señal de entrada de muchos otros nodos- y al peso relativo de cada nodo a un nodo. El conocimiento consiste del estado total de la red en la misma condición de equilibrio después que ha respondido a los patrones de entrada. En su presentación de 1982 a la Academia Nacional de Ciencias, Hopfield comenzó en el fin opuesto, describiendo la función del sistema como de la estabilidad desarrollada. Elementos de procesamiento responden a entradas, y las modificación de red continúan hasta un estado estable. La red no computa respuestas como lo hacen las computadoras convencionales, algunas veces es más que una materia de memorización de las respuestas por las entradas dadas.

2.1.4. Tolerancia Defecto

Las redes neuronales son extremadamente faltas de tolerancia y degradadas fácilmente. Ellas pueden aprender y tomar decisiones basadas en datos incompletos. Porque el conocimiento es distribuido a través del sistema en lugar de permanecer residente en locaciones simples de memoria., un porcentaje de los nodos pueden ser inoperativos sin significantes cambios en el comportamiento global del sistema. Resistencia al fallo en el hardware aun siendo este muy grande en una red neuronal, lo que en una computadora convencional usualmente es catastrófico.

2.1.5. Adaptabilidad

La adaptabilidad es la habilidad de una red neuronal para autoajustarse. Esta es una de dos características importantes para redes neuronales. La cual esta compuesta de cuatro aspectos:

- Aprendizaje
- Auto-organización
- Generalización
- Entrenamiento
  - Supervisado
  - No supervisado

Fig. 2.3. La Adaptabilidad incluye ...

2.1.5.1. Aprendizaje

El aprendizaje es el nivel de un simple PE. El aprendizaje ocurre cuando las conexiones de peso son ajustadas. Algoritmos específicos para cada problema de la red necesitan solucionarse para poder pasar a un problema mayor y hacer así la red especifica, aunque su uso podría ser estrictamente limitado. Como un ejemplo, supóngase que usted estuvo usando una red para ajustar el ángulo del brazo de un robot en respuesta a la retroalimentación desde las imágenes de cámara. Si la precisión de un grado fuese requerida, un algoritmo necesitaría permitir 360 variaciones diferentes en un plano dado. Esto podría resultar un problema de considerables proporciones, especialmente en espacios de tres dimensiones.

Una red neuronal, sin embargo, generara su propio algoritmo para ajustar las conexiones de peso entre los elementos de procesamiento.

Ejemplo: Robot Forklift. Martín Merieta ha entrenado un brazo de robot para reconocer y recoger paletas siempre que ellas están en ángulos extravagantes. El robot industrial tiene un tenedor como su fin-efecto. Usa una variedad de sensores infrarrojos para determinar la locación y posición de objetos que están para ser alzados y movidos. Una red neuronal guía el movimiento en seis dimensiones para determinar los puntos x,y,z en el espacio, el grado de inclinación, rodamiento y postura para poder ajustar e insertar el tenedor dentro de una paleta. Estas funciones de control son aprendidas en lugar de ser programadas. Un maestro humano muestra al robot como hacer esto. Durante el entrenamiento, el robot hace juegos con la información desde sus sensores próximos a la trayectoria dada, comandada por el operador humano vía joistick o teclado. En seguida la red hace los ajustes internos para imitar los patrones, esto puede ejecutarse sobre si mismo.

Los resultados son alentadores. En algunos casos, con muy pocos puntos enseñados, la red aprende a recoger la paleta desde cualquier posición arbitraria de comienzo. Un punto enseñado es simplemente, un punto sobre la ruta, no la trayectoria entera. La habilidad de la red para generalizar la trayectoria desde un pequeño número de puntos es una de sus más imprevistas características. Para permitir una similar flexibilidad usando algoritmos resulta una abrumadora tarea.

2.1.5.2. Auto-organización

La auto-organización es la modificación de muchos elementos de procesamiento al mismo tiempo. Sesiones de entrenamiento ejercitan las reglas para aprender como las modificaciones toman lugar por todo el sistema entero de red. Esto es como si las redes neuronales estuvieran desarrollando sus propias heurísticas como que ellas ven a través de las iteraciones. La red converge, o se posa en una respuesta estable.

Ejemplo: Robots de Nakano. Kaoru Nakano, un profesor asociado a la Universidad de Tokio, ha usado redes neuronales en la construcción de un número de diferentes robots auto-organizadores. Sus robots han aprendido a hacer una variedad de cosas: autoreparse, caminar, hablar, jugar pelota. El comportamiento auto-organizador llega a ser aparente cuando las acciones al azar intentadas son sucesivas y aun son totalmente diferentes que al poder humano que tiene programado. Tome el acto del lanzador de pelotas. El robot, con un número de servos controla su brazo flexible, es recompensado o castigado sobre las bases de cuan lejos fue lanzada la bola.

2.1.5.3. Generalización

La generalización es la habilidad de la red para responder a entradas que no han sido vistas antes. La entrada puede ser parcial o incompleta, como serán discutida después en el párrafo sobre intuición, predicción, y emparejamiento estático de patrones. La generalización toma la habilidad para aprender y autoajustarse a pasos mas lejos. El sistema puede hipotetizar una respuesta.

Otros dos robots auto-organizadores de Nakano aprendieron a formar palabras. Observando el mundo alrededor de ellos, ellos unieron nombres a conceptos. Cuando dos robots reconocedores tuvieron la misma experiencia, ellos intercambiaron palabras. Las palabras internas de cada uno podían ser modificadas para archivarlas a una pareja más cercana. De este modo, ellos inventaron y comunicaron su propio lenguaje a otros robots.

La habilidad para aprender funciones de control complicadas, y la habilidad para responder a cambiantes o unespectados ambientes, expande la utilidad de los robots. Esta tecnología puede ser usada para habilidades submarinas, en maquinaria automatizada, sobre cadena de montaje, y para tareas arriesgadas, tales como exposición a materiales radiactivos.

2.1.5.4. Entrenamiento

El entrenamiento es la manera en que una red neuronal aprende. El entrenamiento puede ser supervisado (Modelos de Hopfield y Perceptrones son ejemplos del uso de entrenamiento supervisado) o no supervisado (redes Kohonen). El primero provee a la red con ejemplos de la respuesta deseada, el posterior no proporciona salidas de ejemplo, pero la red puede crear grupos de características deseadas (tales como robots creando palabras). Los robots de Nakano no habían obtenido avance de entrenamiento, ellos aprendieron sobre sus propias experiencias, pero estuvieron reforzándose en relación a sus acciones. Reforzando, graduando o escalando la red sobre su funcionamiento, en algún lado entre ellos

El entrenamiento puede incluir tales actividades como proveer orientación a bases de datos, colocando objetos en frente de sensores o presentando otros ejemplos. Un ejemplo de entrenamiento supervisado sería enseñar a una red a reconocer imágenes de números. Usted podría presentar entradas de imágenes de números con las salidas esperadas. Esto es lo que dice la red, "Cuando tu ves este patrón (un 7 por ejemplo) dame un 7. Todo esto es hecho para archivar una particular auto-organización del sistema.

Después del entrenamiento, el sistema esta listo para usarse. Dependiendo de la tarea ha realizar, la red puede tener sus pesos entrenados "congelados"; así de esta modo son imposibilitadas sus leyes de aprendizaje.

Esto podría ser apropiado para una aplicación que realiza la tarea de texto a lenguaje (voz). No sería apropiado, sin embargo, para un sistema que debe continuamente ajustándose a condiciones de cambio, tales como una red para hacer procesos de control de tiempo real.

2.1.6. Reconocimiento de Patrones

Diferentes tipos de redes están bajo consideración por varios tipos de búsqueda y objetivos de aplicación. Diferencias en modelos son en su mayor parte debido a el número de elementos de procesamiento, como están conectados y cuales son las leyes del aprendizaje. Estas diferencias estructurales crean diferentes paradigmas de aprendizaje para las cuales las neuronas realmente son clasificadas.

La estructura de una red neuronal le permite ser apta en tipos particulares de aprendizaje. Muchas redes son buenas en el reconocimiento de patrones. Otras pudieran ser llamadas constructoras de clasificación.

Algunos sistemas extraen características, tales como esquinas de sólidos, y son detectores de regularidad. Todavía otras redes son asociadores automáticos. Absolutamente una variedad de tareas pueden ser emprendidas, pero la mayor parte de ellos se relacionan de algún modo con una habilidad para distinguir patrones.

En general, las redes neuronales son mejor distinguidas para el reconocimiento de patrones. Las redes tienen la habilidad para escoger un conjunto de características previamente aprendidas, o ellas pueden seleccionar y generar sus propios caracteres distintivos de patrones. Esto no es limitado a patrones estáticos cualquiera. El potencial para procesamientos complejos y patrones dinámicos que varían en tiempo y espacio proveen esperanzas para nuevas soluciones en tiempo real.

Ejemplo: Nestor Writer. Un ejemplo comercial de coincidencia de patrón es Nestor Writer, un reconocedor de manuscrito (Primer producto es Nestor, el cual corre sobre una IBM PC AT y es parcialmente basado en un reconocedor de patrón de red neural). Las entradas son en forma de manuscrito sobre una almohadilla digitalizadora. Después de comenzar el entrenamiento sobre un conjunto típico de ejemplos de manuscritos, Nestor Writer puede interpretar el manuscrito que no ha visto previamente, y esto puede ser completado a pesar de cambios en escala, mayúsculas, en posición, distorsión e idiosincrasias en estilo.

La flexibilidad del sistema se extiende a aceptar otros símbolos como Kanji, caracteres japoneses. La dificultad computada a la entrada es eliminada. No solamente pueden las redes detectar patrones , ellas pueden hacer esto tanto en tiempo (como en señales auditadas) como en espacio (usando información visual).

2.1.7. Patrones Incompletos

Suponga que parte de los patrones de entrada son extraviados o contienen información engañosa o con ruido (Ver Fig. 2.4). Que sucede entonces?.

Justamente como usted puede reconocer gente cuando observa solamente una parte de su cara, las redes neuronales son buenas en la observación de patrones parciales y adivinar de quien es. A esto se le llama **intuición**. Esta capacidad es combinada con una habilidad para la reconstrucción estática de patrones, donde el patrón computado es el más cercano a el patrón parcial. En otro sentido, esto significa que el patrón computado será el que mejor se adapte a la información. Si usted toma un punto sobre el patrón parcial, y encuentra el correspondiente punto sobre el patrón computado, la suma de los cuadrados eliminan problemas íntegramente en suma de errores positivos y negativos de las distancias entre todos estos correspondientes conjuntos de puntos tan pequeño como sea posible. Las características de intuición, predicción y reconstrucción estática de patrones permiten a las redes neuronales tratar con situaciones en las cuales la entrada puede ser cubierta, incompleta, ambigua o puede siempre tener algo de información corrompida.

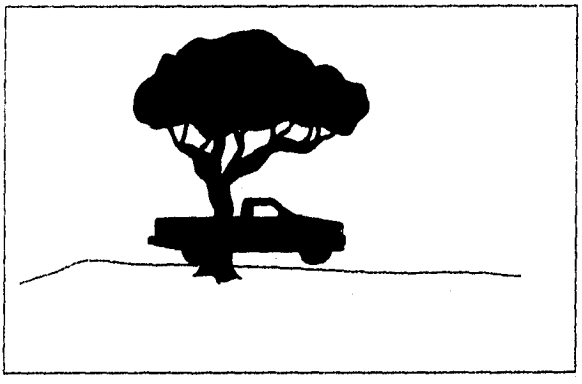


Fig. 2.4. Que parte del patrón, fue perdida en el objeto?.

Porque la mayoría de la información en este mundo es inexacta, esta característica llega a ser altamente significante. Esto también se vincula a la adaptabilidad, la habilidad de una red para aprender, auto-organizarse, generalizar y proveer la llave a esta característica.

2.1.8. Tareas Apropiadas

Algunas veces tareas apropiadas para redes neuronales son realizadas de igual forma por humanos. Reconocimiento de caras es una tarea relativamente fácil de realizar, hasta por un niño pequeño, pero esto ha sido uno de los enormes problemas tradicionales de la Inteligencia Artificial. El reconocimiento de discurso continuo y síntesis son ejemplos adicionales de tareas de Redes Neuronales que son emprendidos como sucesos razonables.

Hay abundancia de buenos ejemplos contables, no obstante, algunas tareas en las cuales las redes neuronales sobresalen han sido dificultosas para que los humanos las realicen bien. El Control de Calidad de uniones soldadas es corrientemente hecho estallando una imagen digitalizada de las uniones y tienen inspección humana. Ellas hacen un juicio cualitativo en si la junta soldada es aceptable o inaceptable. El Control de Calidad es una prioridad para la industria manufacturera, pero esto no es fácilmente logrado.

Las redes neuronales pueden tomar la imagen digitalizada como entrada y conseguir información cuantitativa así como un evaluación cualitativa. Métodos de estática están en proceso con mejoras en las redes neuronales. Inicialmente, muchas tareas para redes neuronales tuvieron grandes conjuntos de ejemplos disponibles o podría generar ejemplos satisfactoriamente apropiados. En algunas aplicaciones es difícil formalizar reglas, pero mucha de la información y ejemplos son disponibles.

Esto es verdadero para algunos problemas de clasificación. Ejemplos como AVCO Financial Services en Irvine, California. Utilizan más de 10 mil ejemplos de préstamos pasados para entrenar su red neuronal, las cuales evaluaron solicitudes de préstamo. El sistema predictor de tiempo basado en redes neuronales se basa en un gran volumen de información pasada para pronosticar condiciones futuras. Note, sin embargo, que direcciones cambiantes en los ejemplos de entrada serán recogidos por la red, así una ventana de movimiento de información de datos históricos permitiría a la red continuar con el aprendizaje y adaptamiento.

El trabajo actual en las redes neuronales esta examinando formas para reducir el tamaño (medida) del conjunto de entrenamiento requerido. Texas Instruments, por ejemplo, ha encontrado buenos sucesos con aplicaciones de técnicas de comprensión de datos a los conjuntos entrenadores. He aquí una posibilidad: si la entrada es una matriz de 100 por 100 pixeles, sume las líneas que cruzan y simplifique la entrada a una matriz de 1 por 100.

Algunas tareas muy a menudo involucran mapeo de objetos en un conjunto con objetos en otro conjunto. Muchas ediciones comerciales significantes son realmente problemas de mapeo. Por ejemplo, para lograr texto a discurso (como en NETtalk), una red tiene que poder asociar las entradas de cartas, espacios y puntuaciones con las salidas de sonidos, pausas y entonaciones. Las imágenes visuales pueden necesitar ser digitalizadas, codificadas, rotadas, transformadas, y pareadas con una apropiada categoría por una

correspondiente imagen de salida. Las técnicas de compresión de datos pueden necesitar proyectar entradas a algunos pequeños conjuntos de datos (para transmisión), el cual puede después ser proyectado detrás del conjunto original. Las redes neuronales pueden estar observándose en una estática basada en tecnología de mapeo. "Después de que la gente entienda que las redes neuronales son altamente eficiente por su funcionamiento en paralelo, una muy rápida tecnología de computación que trabaja estadísticamente en un ensayo y base de error para solucionar problemas que usted no sabría como solucionar de cualquier otra forma, entonces ellas empezaran una revolución en la industria de la computación", dice Robert Smithson, Director de programas de redes neuronales de Lockheed. Algunos sistemas de red actuales requieren que todos los datos de entrada necesarios sean disponibles en cualquier tiempo.

En contraste con un sistema común que no tendría posibilidad para interaccionar, un sistema experto, el cual solicita al usuario entradas con la consulta procedida. De igual forma, con redes neuronales, las entradas tienen que ser numéricas y es generalmente binaria o analógicamente valuada (analógica). Este requerimiento puede necesitar preprocesamiento de datos de entrada. Otros sistemas, tales como el predictor de ambiente (clima) mencionado anteriormente, puede continuamente integrar nueva información, como esta llegue a ser disponible. Sistemas adaptables tienen que tener esta capacidad.

2.1.9. Tipos de Problemas Direccionados

Los tipos de problemas dirigidos por esta tecnología abarcan en cierta medida otra característica. Aunque, un poco de las principales áreas de investigación han sido mencionadas, nosotros hemos preferido enfatizar aquí algunas de las actuales direcciones y aplicaciones potenciales. La Fig 2.5 muestra algunas de las mayores áreas de investigación y de aplicación.

Proyectos de redes neuronales están trabajando sobre los problemas del procesamiento de lenguaje natural, como reconocimiento de discurso continuo y síntesis y transformación de texto a voz. Productos disponibles que hacen reconocimiento de caracteres complejos y análisis de manuscrito, incluyendo análisis de caracteres complejos. Los problemas de reconocimiento de patrones adicionales comienzan a ser trabajados sobre la inclusión de problemas de reconocimiento de imagen de varios tipos. Esta áreas son de particular interés por el aparato militar para aplicaciones tales como determinación de "amigo o enemigo".

Si una imagen parcial direcciona a la versión completa, el termino autoasociativo es usado para describir la red. Por otra parte, si el objeto es identificado desde la entrada que es distorsionada o desorientada, el término heteroasociativo se aplica. Aunque una

red autoasociativa le daría una versión mejorada de la entrada, la red heteroasociativa usualmente le da la información faltante únicamente.

- Reconocimiento y síntesis de discurso.
  Procesamiento de lenguaje natural -texto a discurso.
- Reconocimiento de carácter
  Manuscrito, caracteres complejos
- Reconocimiento de imagen - Autoasociativo/Heteroasociativo
- Compresión de Imagen (información)
- Problemas completos de NP (explosión combinatorial)

Fig. 2.5. Areas de investigación y aplicaciones

Las redes autoasociativas identifican un objeto con ver únicamente parte de ella, como identificar un objeto oscuro en una imagen o recuperar una bibliografía dando únicamente algunas palabras claves. Otro ejemplo sería involucrar algo a una asociación con algunas características: si un animal trepa arboles, entonces quizá sea un gato.

Un ejemplos de una red heteroasociativa (ver Fig. 2.6) incluiría la vista a un objetivo en diferentes ángulos pero todavía identificando eso como el mismo objeto, o reconociendo alguno "en disfraz" o verificación de firmas.

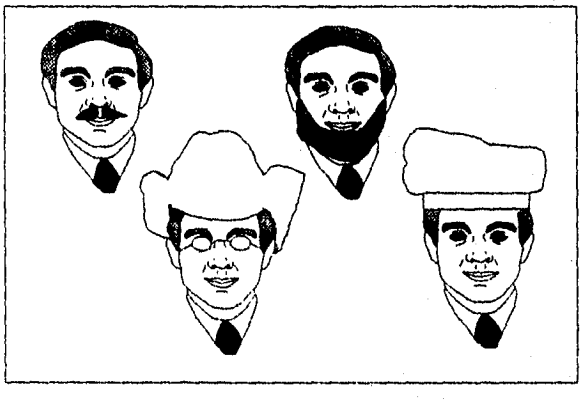


Fig. 2.6.

Los bancos tienen tantos cheques que pasan por unas bases diarias que son imposibles para humanos para examinar cada uno para su corrección. Banatec (en Dallas) es una compañía haciendo uso de redes neuronales para leer cheques.

Las redes neuronales han sido usadas para robótica y aplicaciones de control desde hace gran tiempo. Ellas ofrecen algunas ventajas sobre el enfoque convencional.

Las transformaciones pueden ser aprendidas de ejemplos en lugar de tener que ser derivadas y programadas explícitamente. Controladores de tiempo real, de aprendizaje y adaptativo basados en redes neuronales llegan a ser posibles porque de tales capacidades de la red, como una línea de aprendizaje con intervención mínima, habilita retardadores que serán usados durante la realimentación, y una habilidad para aprender modelos internos del mundo externo.

Con las redes, los sistemas roboticos, pueden ser mas flexibles respondiendo a variaciones dentro de su propio sistema, y se mueve en entornos menos precisos definidos.

Michael Kuperstein, presidente de Neurogen, Inc, ha prototipeado esta flexibilidad en desarrollar un robot neural llamado "Infant" (Redes Neuronales Adaptivas Interactuantes Formando Topografías) "Infant" usa 2 cámaras posicionadas como ojos y tiene un brazo articulado con una mano trasquiladora . Como un bebe puede explorar el mundo y aprender a través de sus movimientos, así Infant explora su mundo y expande sus capacidades. Este camina sobre un terreno desigual, apresa nuevos objetos y aprende a coordinar movimientos para asociarlos con estas experiencias.

La compresión de imagen (dato) es otra tarea emprendida por algunas redes neuronales. Un experimento demostrado, utiliza la reducción de escala de grises de una imagen de 8 bits, dentro de 1 bit por pixel. Tales técnicas muestran también por compresión compleja del dato sensor durante el procesamiento convencional.

Algunas tareas llamadas problemas incompletos y que se pensaban ser intratables, hoy en día están siendo tratables. Los sistemas con neuronas están haciendo el trabajo de rutinar la red de trafico telefónico, localización de chips sobre una tarjeta por buenos marcos proyectados y otros similares recursos de localización de problemas. No solamente rutinado las soluciones se pretende minimizar retardos, algunas veces ellos deben también tener el potencial para responder rápidamente a cambios radicales de condiciones de medio ambiente. Esto a menudo se presenta en sistemas de communicaciones militares. La red neuronal puede no encontrar la mejor solución (la cual puede tomar una enorme cantidad de tiempo siempre sobre una supercomputadora), pero encuentra una respuesta lo suficientemente buena.

2.1.10. Limitaciones y Concernientes

Con las buenas nuevas hay malas también. Hay un número de problemas que las redes neuronales no manejan bien (ver Fig. 2.7.).

Las redes no son buenas, si respuestas precisas son requeridas. Usted no querría usar una red neuronal artificial, por ejemplo, para colocar sus finanzas. Nosotros mencionamos antes que las redes son generalmente buenas en tareas que la gente hace. Extendiendo la analogía, nosotros podríamos decir que hay mucha gente la cual no maneja las finanzas particularmente bien.

Las redes neuronales no pueden contar; ellas observan la selva, no los arboles. Contar todo, es duro para el procesamiento en paralelo. El aprendizaje es duro, tu tiendes a construir sobre lo que tu conoces. No hay estándares para algoritmos de aprendizaje o para entrenamiento de redes.

- El aprendizaje es duro -usted tiende a construir sobre lo que conoce.
- No puede justificar respuestas
- No son buenas si respuestas precisas son requeridas
- No puede contar (observa bosques, no arboles)
- No puede (aun) hacer cosas que las computadoras convencionales pueden hacer.
- Diseñando una red es todavía un proceso misterioso.

Fig. 2.7. Limitaciones

El entrenamiento es dificultoso y algunas veces tedioso. El número de pruebas requeridas pueden ser significantemente grande. Diseñando una red es todavía un proceso misterioso.

Cuando usted construye una, usted intenta una regla de aprendizaje aquí, una diferente función de transferencia hay entonces. Usted ha evaluado y visto como trabaja. Diseñando la arquitectura es una cuidadosa prueba y esfuerzo error.

Un problema dificultoso que las redes neuronales deben superar, es en el dominio del tiempo. Para solucionar reconocimiento de lenguaje, el lenguaje natural se sobreentiende, y diversifica aspectos de la percepción visual, esto sería necesario para diseñar redes neuronales que se adaptan continuamente en el tiempo.

Las investigaciones presentes han envuelto en su mayor parte representaciones espaciales en algunas propiedades temporales muy interesantes, pero estudios adicionales son necesarios (y están en progreso) con observaciones en el tiempo.

En general, una red neuronal no puede justificar sus respuestas. No hay facilidad para (hacer parejas) el "como" o "porque" encontrado en un sistema experto. No hay sentido para parar el sistema y decir "que estas tu haciendo ahora?" esto es como si la red estuviera diciendo: "confía en mi, confía en mi".

En respuesta a este problema, Estefan Gallant de la Universidad de Northeastern ha surgido con una fácil explicación. Ha patentado 2 paquetes de programas constructores de sistemas expertos conectores de ejemplos de entrenamiento.

La Corporación de Neurocomputación Hercht-Nielsen ha permicionado el método Gallant, y otras corporaciones, tales como NeuralWave y Nestor, están de la misma forma trabajando sobre justificación de sistemas. La habilidad de una red neuronal para producir conjunto de reglas que podrían potencialmente proveer explicación para modelos de redes crédito -récord e identificación de problemas en sistemas de diagnóstico.

Escalamiento es otra edición. Problemas de juego pueden converger en una razonable cantidad de tiempo; problemas reales pueden aumentar la tarea mas allá de límites razonables. Un pequeño prototipo puede trabajar solamente bien en el laboratorio, pero esto puede ser completamente otra historia para construir un modelo comercialmente viable.

Cuando las redes generalizan, o adivinan, ellas pueden ser erróneas. Errores pueden ser duros para deshacer u olvidar. Los errores son extendidos sobre las correcciones en la misma forma que están las respuestas correctas, así esto es duro para conocer cuales PE's son culpables.(Algoritmos de Backpropagación direccionan estas ediciones, pero tiene limitaciones tales como requerimientos extensivos de entrenamiento fuera de línea).

Las redes neuronales no pueden todavía hacer cosas que las computadoras convencionales con menos esfuerzo. "Cuando ellas hacen algo que no pueden ser hechas de cualquier otra forma, esto será revolucionario", dice Martin Minsky, Investigador de AI de MIT (Attempts para hacer las redes neuronales mas como computadoras tradicionales (i.c., hace lógica, almacena y carga grandes controladores de información con alta exactitud) no están empleando las virtudes exactas de redes neuronales. Al mismo tiempo, explorando los limites de redes neuronales con respecto a su poder computacional pueden manifestar ellas mas potencial en esta área que es corrientemente realizada.

Inescapada, hay las limitaciones de hardware. Aunque esto puede ser usado para estudiar un proceso paralelo por simulación esto sobre una maquina secuencial, la simulación no es la meta. Maquinas paralelas son necesitadas para procesos paralelo para archivar velocidad y costo efectivo en la implementación en masa. Pero la barrera del hardware esta comenzado a desmoronarse.

2.2. CONCEPTOS BASICOS DE RECONOCIMIENTO DE PATRONES

El reconocimiento es relacionado con base a atributos de inicios humanos, también como otros organismos vivientes. Un patrón es la descripción de un objeto. Nosotros estamos desempeñando actos de reconocimiento a cada instante de nuestro caminar de vida. Reconocemos objetos alrededor nuestro, y nos movemos y actuamos en relación a ellos. Podemos observar a un amigo en una multitud y reconocerlo; podemos reconocer la voz de un individuo conocido; podemos leer manuscrito y analizar huellas digitales; distinguir risas de gestos de enojo. Un ser humano es un sistema de información muy sofisticado, en cierto modo porque el posee una capacidad de reconocimiento superior de patrones.

De acuerdo a la naturaleza de los patrones a ser reconocidos, dividimos nuestros actos de reconocimiento dentro de dos tipos mayores: el reconocimiento de ítems concretos y el reconocimiento de ítems abstractos. Podemos reconocer caracteres, fotografías, música, y objetos alrededor nuestro. Esto se refiere a un reconocimiento sensorial, el cual incluye reconocimiento de patrones visual y aural. Este proceso de reconocimiento envuelve la identificación de patrones espacial y temporal. Por otro lado reconocemos un viejo argumento, o una solución a un problema, con nuestros ojos y oídos cerrados. Este proceso involucra el reconocimiento de ítems abstractos y pueden ser termino de reconocimiento conceptual, en contraste al reconocimiento de patrones visual o aural. Ejemplos de patrones espaciales son caracteres, huellas digitales, mapas de medio ambiente, objetos físicos, y fotografías. Los patrones temporales incluye lenguaje, formas de onda, electrocardiogramas, firmas y series de tiempo.

El reconocimiento de patrones concretos por el ser humano puede ser considerado como un problema psicofisiológicos el cual involucra una relación entre una persona y un estímulo físico. Cuando una persona percibe un patrón, el hace una inferencia inductiva y asocia esta percepción con algunos conceptos generales o pistas los cuales han derivado de sus experiencias pasadas. El reconocimiento humano es en realidad una pregunta de estimación de los relativos sobrantes que los datos de entrada pueden ser asociados con uno de un conjunto de poblaciones estáticas conocidas el cual depende de nuestra experiencia pasada y la cual forma las pistas y la información a priori del reconocimiento. Así, el problema de reconocimiento de patrones puede ser relacionado como discriminación de datos de entrada, no entre patrones individuales pero entre poblaciones, la búsqueda por características o atributos invariantes entre miembros de una población.

El estudio de problemas de reconocimiento de patrones puede ser lógicamente dividido dentro de dos categorías mayores:

1. El estudio de la capacidad de reconocimiento de patrones de seres humanos y otros organismos de vida.
2. El desarrollo de teorías y técnicas para el diseño de materiales capaces de desempeñar una tarea de reconocimiento dada para aplicaciones específicas.

El objetivo de la primera área se relaciona con disciplinas como psicología, fisiología y biología. La segunda área se relaciona primeramente con Ingeniería, Computación, y Ciencias de la Información.

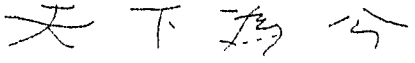
En este parrafo nos referiremos a la computación, Ciencias de la Información y aspectos de Ingeniería del diseño de sistemas automáticos de reconocimiento de patrones. En un lenguaje simple, el reconocimiento de patrones puede ser definido como la categorización de datos de entrada dentro de clases identificables de extracción de características significantes o atributos de los datos de un medio o detalles irrelevantes. La predicción de medio ambiente puede ser tratado como un problema de reconocimiento de patrones, la entrada reservada de datos son en la forma de mapas de medio ambiente. El sistema interpreta estos mapas para extraer las características significantes y hacer un pronóstico basado en estas características. Diagnósticos médicos pueden ser considerados como problemas de reconocimiento de patrones. Los síntomas sirven como los datos de entrada al sistema de reconocimiento, el cual identifica la enfermedad por análisis de los datos de entrada. Un sistema de reconocimiento de carácter es un sistema de reconocimiento de patrones el cual recibe señales ópticas como los datos de entrada e identifica el nombre del carácter. En un sistema de reconocimiento de lenguaje, el nombre de la palabra hablada es identificada sobre las bases de forma de onda acústicas recibida. La tabla 2.1 describe alguna tareas de clasificación, en conjunto con la correspondiente información de entrada y las respuestas de salida.

Clasificación de tareas	Datos de Entrada	Respuesta de Salida
Reconocimiento de Carácter	Señales ópticas o golpes	Nombre del carácter
Reconocimiento de Lenguaje	Formas de Onda Acústicas	Nombre de la palabra
Reconocimiento Parlante	Voz	Nombre del Parlante
Predicción de Medio Ambiente	Mapas de Medio Ambiente	Predicción de Medio Ambiente
Diágnosis Medico	Síntomas	Enfermedad
Predicción de Mercado de Valores	Noticias Financieras y cartas	Predicción de Altas y Bajas de Mercado

Tabla 2.1

El objetivo del reconocimiento de patrones involucra un número de disciplinas científicas, uniéndolas a ellas en la investigación de una solución para el problema común de miembros de reconocimiento de una clase dada en un conjunto que contiene elementos fuzzy de muchas clases de patrones . Una clase de patrones es una categoría determinada por algunos atributos comunes dados. Un patrón es la descripción de cualquier miembro de una categoría representando una clase de patrón cuando un conjunto de patrones caen dentro de clases disconjuntas son disponible, esto es deseado para categorizar estos patrones dentro de sus respectivas clases a través del uso de algunos materiales automáticos. La lectura y procesamiento de cheques cancelados ejemplifica un problema de reconocimiento de patrones. Tales tareas pueden ser fácilmente desempeñados por trabajadores humanos, sin embargo, una máquina puede archivar a muy grande velocidad. Por otro lado algunas tareas de reconocimiento son de tal naturaleza que ellas pueden ser difícilmente desarrolladas por un ser humano solo. Un ejemplo de tal problema de reconocimiento es la detección del sonido de un submarino en medio de otras señales marinas y divulgar a través del análisis de sonidos subacuáticos.

Una obvia pero simple-razonada solución al reconocimiento de patrones es desarrollar, un número de exámenes simples sobre los patrones individuales de entrada en orden a extraer las características de cada clase de patrones. Tales exámenes serían suficientes para distinguir entre patrones de entrada permisibles que pertenezcan a clases diferentes. Considere, por ejemplo los siguientes cuatro caracteres chinos:



Estos caracteres simples pueden ser reconocidos realizando exámenes sobre la existencia de un golpe vertical, un golpe horizontal, un punto simple, un marco inferior abierto, un tope abierto, y un punto secuencial, y conteo del número y secuencia de golpes. Como un segundo ejemplo, considere las siguientes cinco letras en inglés:

C O I N S

Estas letras pueden ser clasificadas haciendo exámenes sobre la existencia de tales características como un lago, un simple bay un doble bay, una línea vertical, y una línea corta. Un diagrama de bloques funcional ilustra el concepto de reconocimiento de patrones descrito arriba y es mostrado en la Fig. 2.8.

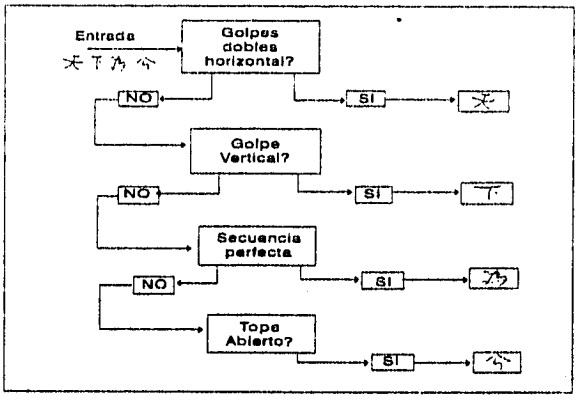


Fig. 2.8. Esquema simple de pregunta-respuesta para la clasificación de Caracteres.

Sin embargo no hay teoría general para determinar cual de todos los posibles exámenes en el mundo real será aplicado a los patrones de entrada. Pocos o pobres exámenes escogidos no caracterizarán los patrones de entrada lo suficiente para permitir categorizaciones dentro de sus respectivas clases de patrones. Muchos exámenes, por otro lado, necesariamente incrementarán la complejidad de los cálculos involucrados en el subsecuente análisis. Ni una regla general hay disponible la cual deba proveer pistas concernientes de como encontrar cualquier conjunto de exámenes, este acercamiento cuentan así mismo excesivamente sobre la experiencia pasada y la intuición de ingeniería del diseñador, y consecuentemente no hace indicación para soluciones satisfactorias a muchos problemas de reconocimiento de patrones que resultan en práctica. Más aproximados poderosos pueden ser concebidos para el estudio cuidadoso de los problemas involucrados en el proceso de reconocimiento de patrones.
Una relación jerárquica existe entre patrones y clases de patrones. En la Fig. 2.9. los caracteres alfanuméricos y los caracteres chinos son patrones, y caracteres en una clase de

patrones. El alfabeto y números son patrones si el carácter alfanumérico es considerado como la clase patrón. Impresos y manuscritos A's, por ejemplo son patrones correspondientes a la letra inglés A, la cual es un patrón clase. En muchos sistemas de información necesitamos una máquina para reconocer varios fons o letras impresas y números, y diferentes estilos de letras manuscritas y numeraciones. En este caso hay 62 clases de patrones representando 26 letras mayúsculas, 26 letras minúsculas, y 10 números. Los diferentes fons y estilos de una letra en particular o número forma los patrones dentro de esta clase.

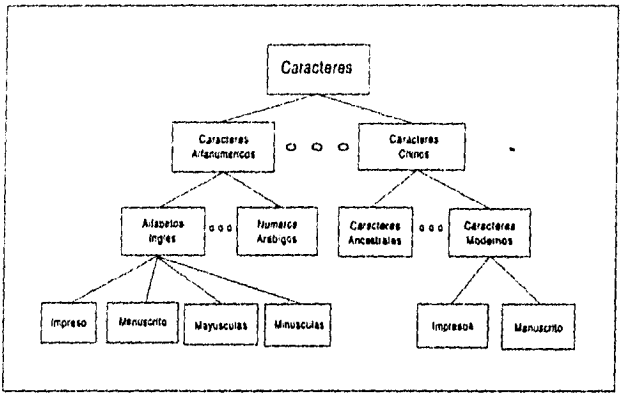


Fig. 2.9. Relación Jerárquica entre patrones y clases de patrones

Considere el problema de reconocimiento de carácter. Una letra específica o número, no es materia de como es impreso o escrito, retiene algunos atributos comunes los cuales son usados como los significados para la identificación. La letra o número es identificado y clasificado acorde a los atributos observados. De este modo, las funciones básicas de un sistema de reconocimiento de patrones son para detectar y extraer características comunes de los patrones describiendo los objetos que pertenecen a la misma clase de patrón, y para reconocer este patrón en cualquier nuevo ambiente y clasificarlo como un miembro de una de las clases de patrones en consideración.

2.3. DISEÑO, CONCEPTOS Y METODOLOGÍAS

Los conceptos de diseño para el reconocimiento automático de patrones son motivados por las formas en las cuales las clases de patrones son caracterizados y definidos. Nuestra experiencia sugiere algunas posibilidades básicas. Cuando una clase de patrón es caracterizado por una lista de sus miembros, el diseño de un sistema de reconocimiento puede ser basado sobre los conceptos de miembros-lista. Cuando una clase de patrón es caracterizada por propiedades comunes compartidas por todos sus miembros, el diseño puede ser basado en los conceptos de propiedades comunes. Cuando una clase patrón exhibe propiedades grupo en el espacio patrón, el diseño puede ser basado sobre conceptos grupos. Estos tres conceptos de diseño básicos son discutidos en los siguientes párrafos.

2.3.1. Concepto de Miembro-Lista

Caracterizado de una clase patrón por una lista de sus miembros sugiere el reconocimiento automático de patrones por emparejamiento de plantilla. El conjunto de patrones pertenecen a la misma clase de patrón que es almacenado en el sistema de reconocimiento de patrón. Cuando un patrón desconocido es mostrado al sistema, este es comparado con el patrón almacenado uno a uno. El sistema de reconocimiento de patrón clasifica este patrón de entrada como miembro de una clase de patrón si este hace pareja con uno de los patrones almacenados pertenecientes a esta clase patrón. Por consiguiente, si letras o diferentes fonts son almacenados en el sistema de reconocimiento de patrones, tales letras pueden ser reconocidas por la calidad de miembro-lista acercamiento tan grande como ellas no son distorsionadas por ruido oportuno, mal entintado, papel poroso. Claramente, este es un simple método pensado. Sin embargo, este concepto puede direccionar para el diseño de esquemas menos caros de reconocimiento el cual sirve de propósito en ciertas aplicaciones. El acercamiento miembro-lista trabajará satisfactoriamente sobre la condición de más cercanos ejemplos de patrones perfectos.

2.3.2. Concepto de Propiedad Común

La caractización de una clase patrón por propiedades comunes compartidas por todos sus miembros sugiere el reconocimiento automático de patrones vía la detección y el procesamiento de características similares. La asunción básica en este método es que los patrones pertenecerán a la misma clase que posee ciertas propiedades comunes o atributos los cuales reflejan similitudes entre estos patrones. Las propiedades comunes, por ejemplo, pueden ser almacenados en el reconocimiento de patrones. Cuando un patrón desconocido es observado por el sistema, sus características son extraídas y algunas veces codificadas y entonces son comparadas con las características almacenadas. El esquema de reconocimiento clasificará el nuevo patrón así perteneciente a la clase de patrón con características similares. De este modo, el problema principal en este acercamiento es determinar propiedades comunes de un conjunto finito de patrones ejemplos conocidos para pertenecer a la clase patrón a ser reconocido.

El requerimiento de almacenaje por las características de una clase patrón es mucho menos severo que por todos los patrones de la clase. Desde características de una clase patrón son invariantes, comparación de características permite variación en patrones individuales. Por otra parte, variaciones significantes de patrones no pueden ser tolerados en plantillas de emparejamientos. Si todas las características de una clase pueden ser determinadas de patrones ejemplo, el procedimiento de reconocimiento se reduce simplemente a emparejamiento de características. Sin embargo, esto es extremadamente difícil, si no imposible, para encontrar el conjunto completo de características discriminantes para una clase patrón, como fue previamente mencionado. La utilización de este concepto, por consiguiente, a menudo necesita el desarrollo de técnicas de selección de características las cuales son óptimas en algunos sentidos.

2.3.3. Concepto de Agrupamiento

Cuando los patrones de alguna clase son vectores cuyos componentes son números reales, una clase patrón puede ser caracterizada por sus propiedades grupo en el espacio patrón. El diseño de un sistema de reconocimiento de patrones basado en este concepto general es guiado por el relativo arreglo geométrico de varios grupos patrones. Si las clases son carecterizadas por grupos los cuales están muy aparte los esquemas de reconocimiento simples tal como clasificadores de distancia mínima pueden ser completamente usados. Cuando los grupos sobreponen, sin embargo, llegan ha ser necesario utilizar mas técnicas sofisticadas para la partición del espacio patrón. Sobreponiendo grupos son el resultado de una deficiencia en la información observada y la presencia de medida de ruido.
De aquí el grado de sobreposición puede a menudo ser minimizado incrementando el número y la calidad de las medidas de funcionamiento sobre los patrones de una clase. Los conceptos

de diseño básico para el reconocimiento básico de patrones descrito arriba pueden ser implementado por tres principales categorías de metodología: heurística, matemática, y lingüística o sintáctica. Esto no es común para encontrar una combinación de estos métodos en un sistema de reconocimiento de patrones.

1.- Métodos heuristicos

El acercamiento heurístico es basado en la intuición humana y la experiencia, haciendo uso de los miembros-lista y conceptos de propiedades comunes. Un sistema diseñado usando este principio generalmente consiste de un conjunto de procedimientos desarrollados para tareas especializadas de reconocimiento. Aunque el acercamiento heurístico es una importante rama del diseño de sistemas de reconocimiento de patrones, poco puede ser dicho acerca de los principios generalizados en esta área desde que cada problema requiere la aplicación de reglas especificas diseñadas para el diseño. En consiguiente, que la estructura y funcionamiento de un sistema heurístico dependerá para un gran grado sobre las habilidades y experiencia de diseñadores de sistemas.

2.- Métodos matemáticos

El acercamiento matemático es basado en reglas de clasificación las cuales son formuladas y derivadas en una estructura matemática, haciendo uso de propiedades comunes y conceptos de agrupamiento. Esto es en contraste con el acercamiento heurístico, en el cual las decisiones son basadas en reglas .El acercamiento matemático puede ser subdividido en dos categorías: determinístico y estático.

El acercamiento determinístico es basado en una estructura matemática la cual no emplea explícitamente las propiedades estáticas de las clases patrones bajo consideración. El acercamiento estático es basado en reglas de clasificación matemática las cuales son formuladas y derivadas en una estructura estática. El diseño de un clasificador de un patrón estático es generalmente basado sobre la regla de clasificación de Bayes y sus variaciones. Esta regla produce óptimos clasificados cuando la función de densidad de probabilidad de cada población de patrones y la probabilidad de ocurrencia de cada clase patrón son conocidos.

3.- Método lingüístico (sintáctico)

Caracterización de patrones por elementos primitivos (subpatrones) y sus relaciones sugieren reconocimiento automático de patrones por la lingüística o acercamiento sintáctico, haciendo uso de los conceptos de propiedad común. Un patrón puede ser descrito por una estructura jerárquica de subpatrones análogos jerárquicos a la estructura sintáctica de lenguajes. Esto permite la aplicación de una teoría de lenguaje normal para el problema de reconocimiento de patrones. Un patrón gramático es considerado consistente de conjuntos finitos de elementos llamados variables, primitivos, y producciones. Las reglas de producción determinan el tipo de gramática. Entre las mas gramaticalmente estudiados están los gramaticadores regulares, gramaticadores de contexto libre, y gramaticadores de contexto sensitivo. La esencia de este acercamiento descansa en la selección de patrones primitivos, la unión de éstos y su relación dentro de gramaticadores patrón, y análisis y reconocimiento en términos de estos gramaticadores. Este acercamiento, es particularmente usado en el trato con patrones los cuales no pueden ser convenientemente descritos por medidas numéricas o son tan complejas que las características normales no pueden ser identificadas y los propiedades globales deben ser usadas.

En la década de los 80's se presentó un considerable interés y rápidos avances en la investigación y desarrollo en reconocimiento automático de patrones y máquinas de aprendizaje. Ejemplos de sistemas automáticos de reconocimiento existen en abundancia. Afortunados intentos han sido hechos para diseñar o programar máquinas para leer impresiones o caracteres manuscritos, para visualización de electrocardiogramas y electroencefalogramas, para reconocer palabras habladas, para identificar huellas digitales y para interpretar fotografías.

Otras aplicaciones incluyen reconocimiento de caracteres manuscritos y palabras, diagnosis médico genetal, clasificación de ondas sísmicas, detección de destinos, predicción de medio ambiente e identificación de fallas y defectos en materiales mecánicos y procesos de manufactura.

2.4. RECONOCIMIENTO OPTICO

En muchas implementaciones para redes neuronales ópticas, los niveles de actividad de operación de elementos de procesamiento están representados por los niveles de opacación de pixeles, arreglos de LED's u otros emisores discretos. Esto tiene algunas ventajas, ya que los destellos de luz cruzan caminos sin interferir con otros, porque la alta densidad de señales de interferencia eléctrica o implementaciones ópticas en el cual el movimiento de fotones o puntos luminosos, es sustituido por electrones[1].

La densidad de interconexiones ópticas pueden ser mucho más grandes que el más avanzado silicón y procesadores de galio o arsénico. Los acercamientos ópticos también generan menos calor y pueden eventualmente operar a una velocidad arriba de 1,000 veces más rápido que los acercamientos electrónicos. Ofrecen grandes ancho de banda y fácil integración con sensores fuente-fin[2].

Dana Anderson (Universidad de Colorado) construyó una memoria asociativa óptica. Si el sistema muestra parte del patrón, la memoria recupera el patrón entero. Si el sistema no presenta cualquier patrón, este desplegará un patrón al azar que Anderson llama "ensueño". Científicos del Laboratorio Bell, entre otros, realizaron investigaciones considerables sobre neurocomputadoras ópticas y la practicalidad de técnicas ópticas de interruptor.

Administradores de bases de datos masivas pueden ser una área que podría ser dominada por tecnología óptica. Pero esto tomará tiempo. Todas las computadoras ópticas no son aún posible. Porque las interconexiones ópticas con interruptores ópticos llegaran a ser un lujo. Esto es aun duro para completar con el corrientemente mas maduro y la tecnología improvista de computadores electrónicos[3].

---

[1] J.Schwartz(Presidente de Schwartz Asociados, Vista Montaña, CA) *"Neural Networks: Capabilities and Applications,"* IEEE Videoconferencias, Septiembre 27, 1989.

[2] DARPA, *"Neural Networks Study,"* (Fairfax, VA:AFCEA International Press, 1988), p. XXVIII.

[3] David A. Brown, *"Neural Research May Benfit AI,"* IEEE Software, vol. 5, no. 6 (Noviembre 1988):88.

2.4.1.Optoelectrónicos.

Muchos modelos de redes neuronales que han mostrado buenas promesas, han a la fecha sido demasiado complejo para construir en hardware. Consecuentemente, ellos han sido corridos sobre simulaciones de software, a menudo sobre supercomputadores. Un planteamiento optoelectrónico, sin embargo, permitiría el diseño de modelos complejos de ICs de redes neuronales. Redes neuronales implementadas en Hardware de orden de magnitudes mas complejas que fueron previamente posibles pueden ser construidas con la tecnología de hoy.

El esquema optoelectrónico puede soportar una densidad de 250,000 interconexiones por pulgada cuadrada. Como mencionamos arriba, los fotones no interfieren con cualquier otro como hacen los electrones. En adición, el esquema permite la flexibilidad de reconfiguración de hardware dentro de modelos enteros diferentes matemáticos, o actualizar el corriente funcionamiento de modelo, con modificación de hardware.

La construcción de redes neuronales fotoconductivas tienen 5 capas que parecen un "sandwich" (ver Fig. 2.10.). Una capa fuente de luz planar, ilumina una capa fotoconductiva. Una capa entre estos 2 sirve como un disfraz. Esta capa media un modulador de luz espacial que actúa como una válvula de luz para modular la iluminación. Este almacena los pesos de interconexión entre arreglos de emisores discretos y detectores.

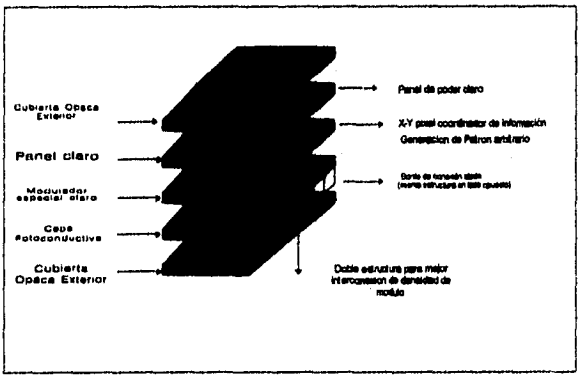


Fig. 2.10. Capas en un red neuronal optoeléctrica (reimpreso con permiso de "Electronic Design", Vol 37, No. 12, Junio 8, 1989. c 1989, Renton Publishing)

En el tope y fondo del sandwich están las capas opacas que permiten a la estructura ser repetida y archivar una elevada densidad de interconectores por módulo. Cualquier patrón duo-dimensional arbitrario puede ser creado, y las interconexiones tridimensionales serían posibles si las sombras de grises son permitidos. Diferentes "mascaras" podrían encaminar a el patrón de interconexiones y cumplir diferentes tareas sin tener que cambiar el hardware. Tales esquemas como este ofrecen la posibilidad para velocidad con flexibilidad y relativa facilidad de uso.

2.4.2. Implementaciones Holográficas

Uno de los más prometedores materiales para implementación de redes neuronales pueden ser los hologramas. Los hologramas pueden ser "programados" para permitir una variedad de mapeo. Un lente convencional mapea cada rayo de luz a un particular punto sobre una imagen plana. Un holograma planar de 1 pulgada cuadrada debe conectar 10,000 fuentes de luz a 10,000 destinaciones arbitrarias. Para archivar la densidad real se requerirá un holograma volumen, hecho de cristal fotorefractivo. Aunque los hologramas todavía parecen algo misterioso para la mayoría de la población, ellos están llegando a ser más comunes.

La revista "Geographics National" pone un holograma globo sobre su cubierta de Diciembre de 1988, y usted puede tener un holograma sobre su tarjeta de crédito. Un holograma es un tipo de fotografía tridimensional producida por fotografía con débiles lentes. La imagen puede ser vista desde una variedad de ángulos. Algo fantasmagórico, aparecerá para ser suspendido en el espacio. Cualquier pieza del holograma contiene la misma información como la totalidad del conjunto y puede reconstruirse la imagen entera. Un holograma común es hecho proyectando luz dentro de un plato fotográfico desde 2 fuentes del mismo objeto y desde un destello de referencia, la luz reflejada por un espejo desde el objeto sobre el plato. El plato entonces contiene patrones que no hacen semejanza a el objeto original y que parecen estar en el vacío. Una luz fuente coherente (diferencia de fase constante), tal como un destello láser, puede ser usado para reconstruir la imagen. Lo que usted observa es una semejanza tridimensional proyectada dentro del espacio a una distancia del plato.

En una implementación holográfica de red neuronal, los hologramas pueden servir como almacenadores de memoria. Luz desde un láser es usado para iluminar 2 objetos que están asociados mutuamente. Los 2 destellos de luz cruzan a través de una sustancia holográfica, tal como un cristal fotorefractivo y producen un patrón interferencia que es grabado sobre el medio.

Un cristal fotorefractivo doblará un destello de luz cuando este golpeé el cristal; los lásers pueden ser usados para cambiar la cantidad de curvatura. Este proceso forma enrejados en el cristal, un grabado del patrón interferencia. Después de que el grabado ha sido hecho, usted puede apagar la luz de cualquier objeto y reconstruir esa luz desde el otro objeto. Hay muchas variaciones pero los principios de operación básicos son similares. Todos ellos almacenan imágenes referencia en cualquier parte de un fino u holograma volumen y recobrarlos en un ciclo retroalimentador iluminado coherentemente. Una imagen de entrada, la cual puede ser ruidosa o incompleta, es aplicada al sistema y es simultáneamente correlacionado ópticamente con todas las imágenes de referencia almacenadas. Estas correlaciones son el umbral y son retroalimentadas a la entrada, donde la correlación es mayormente reforzada (y posiblemente corregida y completada).

La imagen mejorada va alrededor del ciclo repetidamente, acercando la imagen almacenada más cerca sobre cada paso, hasta que el sistema se estabiliza sobre la imagen deseada. Para rellamar una asociación de información distorsionada, ès necesario usar algunos materiales ópticos no lineales tales como lentes y espejos. Suponga un número de parejas de objetos que son asociados y registrados sobre el holograma. Por ejemplo, un par asociativo podría ser un perro/hueso y otro podría ser gato/ratón. Entonces suponga su entrada una imagen que fué una combinación de 2 objetos asociados, una combinación de hueso y ratón. Se confunde? Si. Posiblemente la imagen de salida será un hueso o tal vez un ratón? Si una decisión es requerida, los circuitos ópticos pueden hacer el proceso de decisión-realización. Un conjunto de espejos, por ejemplo, pueden tomar el destello de salida y propagar el destello de luz circularmente de regreso a la entrada donde este se intercepta, así mismo en el cristal. Con este ciclo iterativo llevado a cabo, la energía es reforzada y con posible ruido ayuda a decidir la competición entre posibilidades almacenadas. La memoria rellamará una imagen completa de un objeto solo, no partes de imagenes.

Hay una diferencia importante entre hologramas sobre platos fotográficos y hologramas sobre sustancias fotorefractivas. El primero puede leer pero no puede ser reescrito. Media fotorefractiva, sin embargo, puede ser usado para leer y escribir hologramas simultáneamente. La rejilla registrada envolverá sobre tiempo, produciendo una memoria dinámica. Las escalas de tiempo por cambio son diferentes para sustancias diferentes, pero los objetos en la memoria holográfica podrá ser alterada, reforzada o borrada, sobre el tiempo, esto depende sobre cuan cercanas la entrada se parece a objetos ya almacenados o si este es enteramente una nueva entrada.

Lo dinámico de tales sistemas son una área de considerable investigación. La adición de retroalimentación óptica no lineal para memoria holográfica, por ejemplo ha resultado en una nueva clase de memoria asociativa holográfica no lineal (NHAM). Este sistema ejecuta asociaciones entre patrones de entrada y patrones almacenados como en la holográfica clásica. A diferencia de la holografía clásica, son usadas no-linealidades externas para tomar decisiones y seleccionar entre alternativas competitivas, realizar correcciones de error e incrementan capacidad de almacenamiento.

En adición para trabajar sobre mejores mecanismos no lineales, las investigaciones concernientes incluyen mejor calidad de imagen, grande y permanente capacidad de almacenamiento y programabilidad. Para archivar prácticamente, los sistemas holográficos necesitarán ser integrados con redes neuronales ópticas de propósito general tan bien como el interfacear a la computadora huésped electrónica convencional.

2.4.3. Retinas de Silicón

Carver Mead otro diseñador de chips, cree en la importancia de la estructura biológica e intenta archivar el mismo tipo de representaciones en silicón como las observadas en la biológica. El cree que la estructura biológica debe sus arquitecturas a la estructura de información atada a ella. Las estructuras de los oídos, por ejemplo, son diseñados para procesar la estructura de onda particular de sonidos. Mead y sus colaboradores usan los términos electrónicos para describir sus acercamientos en la creación de materiales biológicos sintéticos.

La retina de silicón desarrollada por Mead y sus colegas es un fotodetector fabricado sobre un chip análogo VLSI de 6 x 8 mm. Un chip solo contiene 100,000 transistores y tiene mas de 2,000 celdas fotosensitivas arregladas en un arreglo cuadrado. Cada celda es 100 por 125 micrones y tiene un pequeño circuito construido enteramente de CMOS. Los alambres simulan los conos encontrados en una retina humana. La información es preprocesada, como lo es hecho por el ojo y el oído. El procesamiento de imagen convencional intenta descubrir límites de objetos y sus movimientos por puntos correlacionados desde imágenes discretas. Estos procedimientos tradicionales registran intensidades absolutas y entonces intentan reconstruir la información posterior.

Por que este no es como los sistemas biológicos "ven", Mead rechaza estos procedimientos. El llama el acercamiento "demasiado costoso" para ser natural. En lugar de ello, su sistema modela el ojo en respuesta a el índice de cambio en las intensidades de luz. La retina de silicón de Mead puede registrar el movimiento continuo de un objeto, precisar y uniformizar sin usar una computadora, para hacer cualquier procesamiento de imagen. Si los sensores convencionales fueron usados para efectuar la misma tarea, ellos requerirán el poder de una supercomputadora CRAY, y hasta entonces, el movimiento registrado no sería tan uniforme como su chip lo pudo hacer. En adición a la retina (lo cual Mead afirma que es medio cercanamente de lo que una retina humana puede hacer), Mead ha también desarrollado un caracol (parte de un oído) de silicón. El eventualmente espera construir un sistema nervioso entero de silicón y crear redes neuronales artificiales que nunca paran la adaptabilidad. Su libro "Analog VLSI and Neural Systems" posiblemente llegará a ser un trabajo fundamental en este campo.

2.4.4. Reconocimiento de Caracteres

Un ejemplo práctico de clasificaciones de patrones automático es encontrado en materiales de reconocimiento de carácter óptico tal como las máquinas que leen los caracteres código sobre cheques de banco ordinarios. El conjunto de carácter estilizado encontrado sobre los más cheques u.s. en día es la Familia Asociación de Banqueros Americana E-13B conjunto de caracteres fuente. Como se muestra en la Fig. 2.11., este conjunto consiste de 14 caracteres los cuales han sido a propósito diseñado sobre una zona cuadriculada de 9x7 en orden para facilitar su lectura.

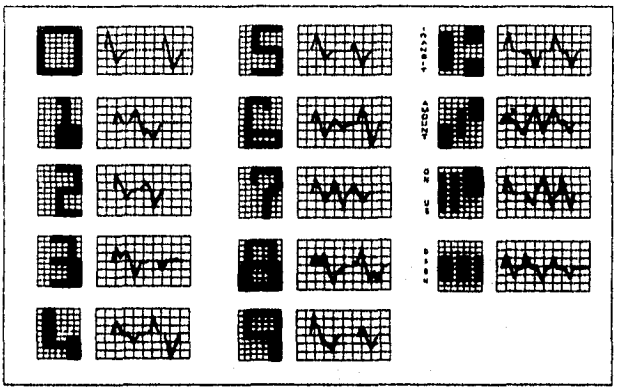


Fig. 2.11. E-13B Asociación de Banqueros Americana, conjunto de fonts caracteres formas de onda correspondientes.

Los caracteres son usualmente impresos en tinta la cual contiene fondos finos de material magnético. Si el carácter es comenzado a leer por un material magnético, la tinta es magnetizada antes de la operación de lectura en orden para acentuar la presencia de los caracteres y de este modo facilitar el proceso de lectura.

Los caracteres son típicamente escaneados en una dirección horizontal tal como una particular cabeza de lectura la cual es estrecha pero mas alta que el carácter. Como la cabeza se mueve a través del carácter, esto produce una señal eléctrica la cual es condicionada para ser proporcional a el índice de incremento del área carácter bajo la cabeza. Considere, por ejemplo, la forma de onda asociada con el número cero en la Fig. 2.11. Como la cabeza de lectura se mueve de izquierda a derecha, el área vista por la cabeza comienza a incrementarse, de tal modo produciendo una derivación positiva.

Como la cabeza comienza a dejar la "pierna" izquierda del cero, el área bajo la cabeza comienza a decrementar, produciendo una derivación negativa. Cuando la cabeza esta en la zona media del carácter el área permanece constante, produciendo un cero derivado, este patrón se repite así mismo cuando la cabeza introduce la pierna derecha del carácter,

como se muestra en la figura. El diseño de los caracteres es vista para ser tal que la forma de onda de cada carácter es distinto de todos los otros.

Esto es notado que las cimas y ceros de cada forma de onda ocurre aproximadamente sobre las líneas verticales de los fondos cuadriculares sobre el cual estas formas de onda son desplegadas en la figura.

La E-13B caracteres ha sido diseñada así para mostrar las formas de onda únicamente en los puntos que produzcan la suficiente información para su apropiada clasificación.

El lector de carácter tiene únicamente estos puntos almacenados para cada uno de los 14 caracteres. Cuando un carácter es para clasificación, el sistema empareja sus formas de onda en contra de las formas de onda prealmacenadas y clasifica el carácter acorde a la pareja mas cercana. Este esquema hace uso de los miembros-lista o conceptos de propiedad común. Lecturas mas estilizadas de carácter fuente sobre el mercado de hoy operan sobre este principio.

Maquinas capaces de reconocer una variedad de fonts han sido también comercialmente implementados. El sistema de entrada 80 desarrollado por Equipo de Reconocimiento Incorporado, por ejemplo, lee mecanografía, impresiones, e información manuscrita directamente de documentos fuente de índices arriba de 3600 caracteres por segundo. El vocabulario de este sistema es modular y puede ser diseñado a los requerimientos de aplicaciones específicas.

Un sistema de particular font puede leer cualquiera de una vasta selección de tipos de fonts, mientras un sistema multifont puede leer simultáneamente una variedad de diferentes tipos de fonts escogidos por el usuario de una lista de fonts disponibles. Arriba de 360 distintos caracteres pueden ser reconocidos por una particular máquina. Un sistema puede también ser estructurado así que esto leerá números a mano impresos y seleccionar letras manuimpresas y símbolos en conjunción con la lectura de información máquina-impresa.

2.5. El PROCESAMIENTO DE DISCURSO DIGITAL, COMUNICACION POR VOZ HOMBRE-MAQUINA

La investigación en el reconocimiento mecánico de discursos es situada como la base fundamental para los avances en el reconocimiento de patrones e inteligencia artificial. Explica la naturaleza de algunos avances y proveen una introducción al estado del reconocimiento de discurso automático (Automatic Speech Recognition (ASR)), se hace mayor la comunicación hombre-maquina debido a las ventajas inherentes de la comunicación por voz y nuestra necesidad de comunicación con máquinas. Las ventajas inherentes de la voz surge de su universalidad, conveniencia, y rapidez. Hasta con las mejores ayudas de comunicación de alta tecnología, la voz o las palabras permanecen sin rival como el modo más rápido y más conveniente para la comunicación interactiva entre seres humanos. En experimentos en los cuales se involucra la voz y otras modalidades de comunicación (p.e., mecanografiar), la información está cambiada al menos dos veces tan rápidamente con discurso como sin discurso[4]. Aún más, las manos y ojos son libres y el orador tiene movilidad, lo que otros modos de comunicación no ofrecen.

Progresos hacia sistemas de reconocimiento de discurso automáticos económicamente viables han sido lentos pero en los pasados 25 años constantes, con un sobresaliente despunte en la tasa de progreso durante los últimos cinco años. Los sistemas de reconocimiento de palabra aislados para vocabularios pequeños con altavoces conocidos se habían puesto comercialmente exitosos durante los últimos cinco años ampliamente al costo disminuido de hardware adecuado.

Pero mientras máquinas que reconocen formas limitadas de discurso continuo están comenzando a aparecer como curiosidades en laboratorios, probablemente pasarán algunos años antes que las máquinas reconozcan discursos de conversación normal. La razón es que el mejorar los sistemas de reconocimiento existentes para discursos continuos requiere más que razonables avances en tecnología de hardware. Sin embargo es solamente una cuestión de tiempo. Mientras tanto estas investigaciones están estimulando avances conceptuales en reconocimiento de patrones e inteligencia artificial, que a su vez están dibujando formas de trabajo en psicología cognitiva, lingüística, psicobiología, y otros campos más retirados del análisis acústicos de discursos.

Al considerar los aspectos variados del reconocimiento de discurso, si por tripular o maquinar, comenzaremos con inteligencia artificial y proceso de la información contextual. Entonces movernos a recientes avances en reconocimiento de discurso automático (ASR), problemas teóricos, y finalmente a soluciones potenciales.

[4] A. Chapanis, *"Interactive Human Communication,"* Scientific American, Vol. 232 (3), pp. 36-42, 1975.

Hay recientes publicaciones que complementan los materiales a cubrir aquí: Reddy[5] da a un excelente esbozo del campo de reconocimiento de discurso completo, cubriendo un reconocimiento de palabras aisladas, reconocimiento de discurso continuo, y comprensión de discurso por computadora. Martin[6] da un buen detalle descriptivo de sistemas de reconocimiento de palabras aisladas. Schroeder[7] describe el procesamiento de la señal humana en " Modelos de Audición". Makhoul[8] presenta un excelente papel sumarización codificación predecible lineal (LPC), una de las mayores innovaciones en la investigación de discurso en recientes años.

Cual es la posible y razonable espera de reconocimiento de discurso, y confiabilidad en el trabajo? Las respuestas dependen del tipo de sistema de discurso. El reconocimiento de palabra aislada es relativamente fácil para vocabularios pequeños (10 a 50 palabras) y para un conjunto pequeño de altavoces conocidos.

En el otro extremo observamos los sistemas de reconocimiento de discurso continuo, vocabularios irrestrictos, altavoces desconocidos, y muchos tópicos de conversación no son concebibles con tecnología y conocimiento de hoy. Intermediarias en dificultad son sistemas de reconocimiento/comprensión de discursos continuos limitados en vocabulario, tamaño y número de altavoces que requieren cuidadosamente de la pronunciación del discurso en un tópico particular. Tales sistemas están ciertamente en los límites exteriores de la tecnología de hoy.

Las aplicaciones que se sirve hoy por equipo de reconocimiento de discurso automático (ASR) comercialmente disponible incluye el siguiente:

1) Entrada de datos por inspectores de control de calidad (Owens-Illinois Corp. la utiliza por medio de T. V. para inspeccionar las placas de revestimiento; La Ford Motor Company para inspección de línea de ensamble de carros; Continental puede por inspección jalar anillos y poner tapas; y Semiconductor Manufacturer la usa con microscopios, para inspeccionar componentes microscópicos.

2) Control de equipo de materiales-manipulación (United Airlines para manipulación de equipaje; Kresge para control de sistema de encauzado de paquete)

3) programación de la computadora de propósito especial (un fabricante utiliza reconocimiento de discurso automático para programar máquinas herramienta), y

[5] D.R. Reddy, *"Speech Recognition by Machine: A Review,"* to be published in the Proc. IEEE, April 1976.

[6] T.B. Martin, *"Applications of Limited Vocabulary Recognition Systems,"* in Speech Recognition: Invited Papers of the IEEE Symposium (D.R. Reddy, ed.), Academic Press, New York, 1976.

[7] M.R. Schroeder, *"Models of Hearing,"* Proc. IEEE, Vol. 63 (9), pp. 1322-1352, September 1976.

[8] J. Makhoul, *"Spectral Analysis of Speech by Linear Prediction,"* IEEE Trans. Audio Electroacoustics, Vol. Au-21 (3), pp. 140-148, 1973.

4) editando de información financiera (EMI Ltd., Inglaterra la utiliza para reunir datos numéricos de una variedad de fuentes para preparar estadísticas financieras mensualmente).

2.5.1. El esbozo del Acto de reconocedor de discurso

Todos los reconocedores de discurso, biológicos o mecánicos, tienen transductores para convertir ondas de sonido en representaciones internas. Todos los reconocedores de discurso tienen modelos internos almacenados en memoria de los patrones acústicos producidos por actos de discurso que concuerdan con representaciones internas de discurso desconocido. Algunos modelos pueden estar como patrones (p. e., palabra patrón); otros modelos como conjuntos de reglas (p. e., reglas gramaticales de Inglés). Otros modelos están implícitos en las funciones de transferencia del transductor (micrófonos, amplificadores, etc.) y compresores de información. Veremos como estos modelos están relacionados por medio de la Fig. 2.12.

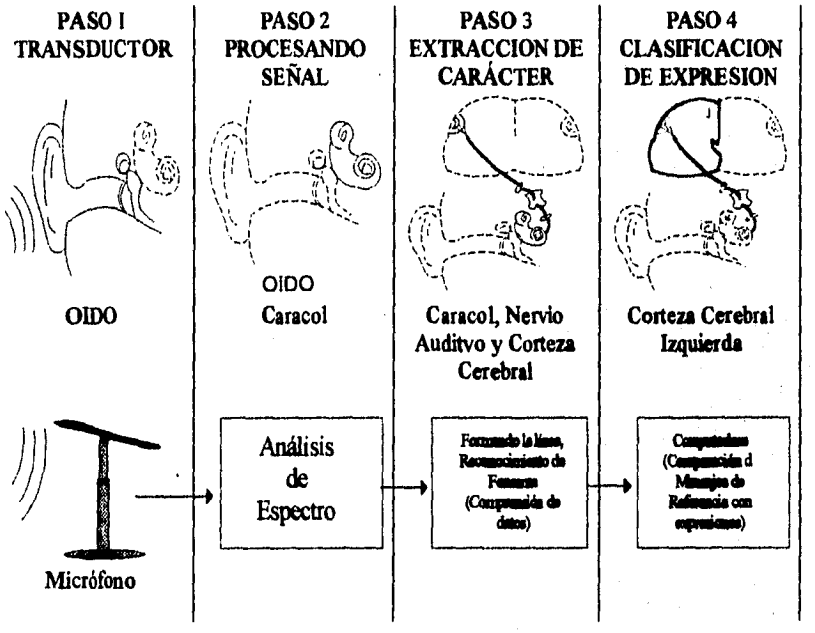


Fig. 2.12. Reconocimiento de palabras usando en forma paralela el reconocimiento humano y de máquina.

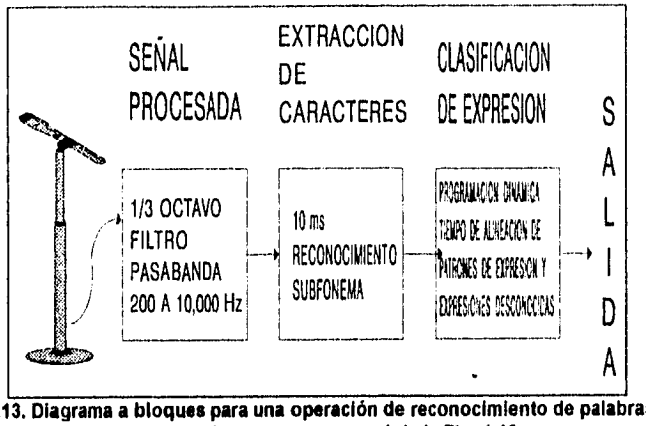


Fig. 2.13. Diagrama a bloques para una operación de reconocimiento de palabras, un ejemplo de la estructura general de la Fig. 1.13.

2.5.2. El procesamiento digital de palabras

Al considerar la aplicación de técnicas de procesamiento de señales digitales a problemas de comunicación de palabras, es de gran ayuda para dividirse en tres tópicos principales: la representación de palabras en forma digital, la instrumentación de técnicas de procesamiento sofisticadas, y las clases de aplicaciones que se basan en el procesamiento digital.

La representación de palabras en forma digital es, por supuesto, de interés fundamental. A este respecto estamos guiados por el teorema de muestreo[9] que plantea que un límite de banda puede ser representado por muestras tomadas periódicamente a un tiempo siempre que las muestras estén tomadas en una tasa suficientemente alta. Así, el proceso de muestreo es el fundamento de toda la teoría y aplicación del procesamiento digital de palabras. Hay muchas posibilidades para la representación de palabras. Como se muestra en Fig. 2.14., estas representaciones pueden estar clasificadas en dos grupos amplios, es decir en representaciones en forma de onda (waveform) y representaciones paramétricas. Las representaciones de forma de onda, como el nombre implica, está referido a conservar la "forma de la mercancía" de la señal analógica de la palabra a través de un muestreo y proceso de cuantificación. Las representaciones paramétricas, por otra parte, están referidas a la representación de la señal como la salida de un modelo para producción de palabras.

[9]H. Nyquist, *"Certain Topics in Telegraph Transmission Theory,"* Trans. AIEE, Vol. 47, pp. 617-644, February 1928.

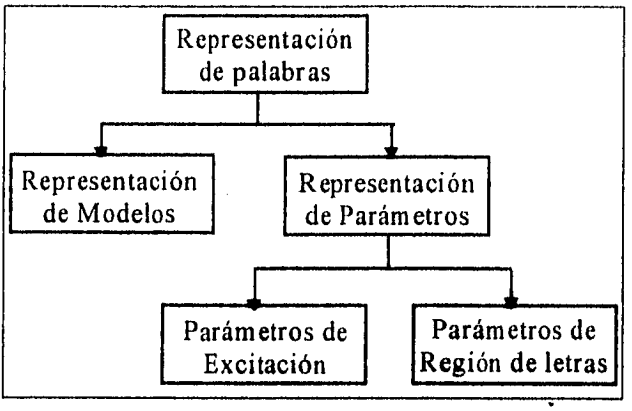


Fig. 2.14. Representación de palabras.

El primer paso al obtener una representación paramétrica es frecuentemente una representación de la forma de onda digital, que obtiene los parámetros del modelo para producción de palabras. Los parámetros de este modelo están clasificados convenientemente como parámetros de excitación (relacionados con la fuente de discurso sonoro) o parámetros de respuesta de tacto vocales (relacionado con el discurso individual sonoro).

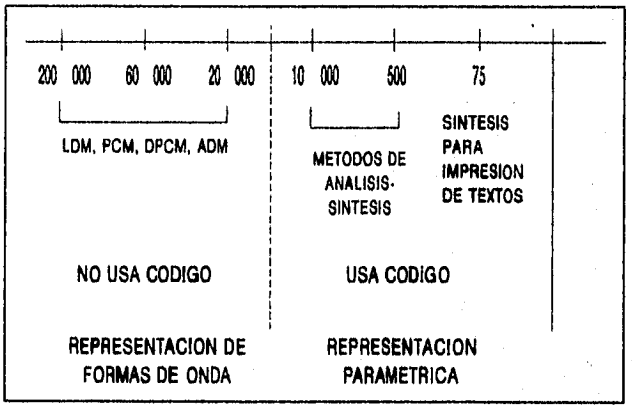


Fig. 2.15. Rango de regla de bits para varios tipos de representaciones de palabras.

La Fig. 2.15. muestra una comparación de un número de diferentes representaciones de señal de discurso de acuerdo con la tasa de información requerida. La línea punteada, es una tasa de información de cerca de 15,000 pedacitos por segundo, separa la tasa de información alta de la representación de la forma de onda a la izquierda de la más baja tasa de representación parámetricas a la derecha. Esta figura muestra variaciones en la tasa de información desde 75 pedacitos por segundo (aproximadamente la información de mensaje básica del texto) tasas de información hacia arriba de 200,000 pedacitos por segundo para el una simple representación de forma de onda. Esta representación acerca de unas 3000 o 1 variación en tasas de información en dependencia de la representación de la señal. Por supuesto la tasa de información no es la única consideración al elegir una representación de palabras. Otras consideraciones son el costo, flexibilidad de la representación, calidad de las palabras, etc.

2.5.3. Aplicaciones típicas de la comunicación por voz

La aplicación definitiva es quizás la consideración más importante en la opción de una representación de la señal y los métodos de procesamiento digital subsiguientemente aplicados. La Fig. 2.16. muestra sólo algunas de las muchas áreas de aplicación en comunicaciones de palabras. A continuación se hará una breve descripción de cada área.

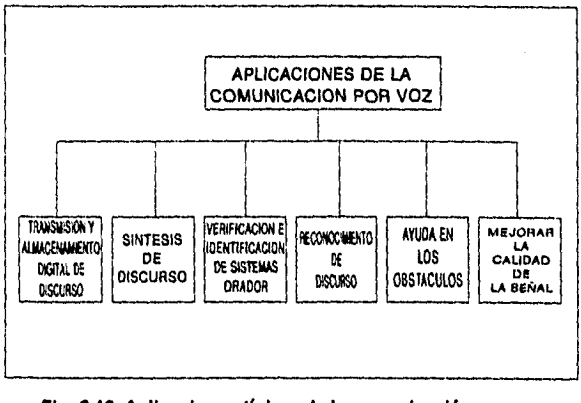


Fig. 2.16. Aplicaciones típicas de la comunicación por voz

2.5.3.1. La transmisión y almacenamiento digital de discurso[10].

Una de las aplicaciones más temprana y más importantes del procesamiento de palabras fue el *vocoder* o "codificador de voz", inventado por Homer Dudley en los 1930[11]. El propósito del *vocoder* fue reducir el ancho de banda requerido para transmitir la señal de discurso. La necesidad de conservar el ancho de banda permanece, en muchas situaciones, a pesar del aumento del ancho de banda suministrado por el satélite, microondas, y sistemas de comunicaciones ópticos.

Aún más, una necesidad ha surgido para sistemas que digitalizan palabras en como bajo un poco tasa como sea posible, consistente con costo de terminal bajo para aplicaciones futuras en la toda planta telefónica digital. También, la posibilidad de extremadamente sofisticada {encryption } de la señal de discurso es motivación suficiente para el uso de transmisión digital en muchas aplicaciones.

2.5.3.2. Los sistemas de síntesis se discurso

Gran parte del interés en sistemas de síntesis de discurso está simulado por la necesidad de almacenamiento digital económico de discurso para sistemas de respuesta de voz de la computadora[12]. Los sistemas de respuesta de voz de la computadora son básicamente un todo digital, servicio de información automático que puede estar consultado por una persona de un tablero o terminal de llave, y que responde con la información deseada por voz. Desde un teléfono de Toque-tono ordinario puede ser el teclado para un sistema, las capacidades de tal servicio de información automático puede estar hecho universalmente disponible sobre las facilidades telefónicas cambiadas sin la necesidad de algún equipo especializado adicional[13]. Los sistemas de síntesis de discurso también juegan un papel fundamental al aprender acerca del proceso de producción de discursos humanos[14].

---

[10]J. L. Flanagan, *"Computers That Talk and Listen: Man-Machine Communication by Voice,"* Proc. IEEE, Vol. 64, No. 4, pp. 416-432, April 1976.

[11]H. Dudley, *"Remaking Speech,"* J. Acoust. Soc. Am., Vol. 11, pp. 169-177, 1939.

[12]L. R. Rabiner and R. W. Schafer, *"Digital Techniques for Computer Voice Response: Implementations and Applications,"* Proc. IEEE, Vol 64, pp. 416-433, April 1976.

[13]J. L. Flanagan, *"Computers That Talk and Listen: Man-Machine Communication by Voice,"* Proc. IEEE, Vol. 64, No. 4, pp. 416-432, April 1976.

[14]C. H. Coker, *"A Model of Articulatory Dynamics and Control,"* Proc. IEEE, Vol. 64, No. 4, pp. 452-460, April 1976.

2.5.3.3. La verificación e identificación de sistemas orador[15]

Las técnicas de verificación y identificación de orador involucra la autenticidad o identificación de un orador de un conjunto grande de oradores posibles. Un sistema de verificación de orador tiene que decidir si un orador es la persona que reclama para ser. Un sistema tal es potencialmente aplicable para situaciones de estricto control de acceso para información o áreas restringidas tales como transacciones de crédito automatizadas. Un sistema de identificación de orador tiene que decidir que orador entre un conjunto de altavoces produjo una serie de palabras dadas. Tales sistemas tienen aplicaciones forenses potenciales.

2.5.3.4. Los sistemas de reconocimiento de discurso[16]

Reconocimiento de discurso es, en su forma más general, una conversión de una onda acústica en su equivalente escrito de la información del mensaje. La naturaleza del problema de reconocimiento de discurso es pesadamente dependiente de las restricciones situadas en orador, situación y mensaje de hablar contexto. Las aplicaciones potenciales de sistemas de reconocimiento de discurso son muchos y variados, (una voz operada comunicación de máquina de escribir y voz con computadora). También, un sistema de reconocimiento de discurso combinado con un sistema sintetizador de discurso abarca los sistemas de comunicación de velocidad de transmisión de bits bajos definitivos.

2.5.3.5. Ayuda en los obstáculos

Esta aplicación invalidada preocupa procesamiento de una señal de discurso para hacer la información disponible en una forma que es mejor pareada a una persona invalidada que está normalmente disponible. El retroceso de tasa por ejemplo variable de cintas preregistradas proveen un oportunidad para un " lector " de ciego proceder a cualquier espacio deseado dentro de material de discurso. También una variedad de técnicas de procesamiento de la señal han estado aplicado diseñar ayudas sensoriales y los displays visuales de información de discurso ayuda como en personas sordas enseñantes para hablar[17].

---

[15]B.S. Atal, *"Automatic Recognition of Speakers from Their Voices,"* Proc. IEEE, Vol. 64, No. 4, pp. 460-475, April 1976.

[16]D. R. Reddy, *"Speech Recognition by Machine: A Review,"* Proc. IEEE, Vol. 64, No. 4, pp. 501-531, April 1976.

[17]H. Levitt, *"Speech Processing Aids for the Deaf: An Overview,"* IEEE Trans. on Audio and Electroacoustics, Vol. AU-21, pp. 269-273, June 1973.

2.5.3.6. La mejora de calidad de la señal

En muchas situaciones, la señal de discurso está degradada en modos que limita su efectividad para la comunicación. En tales técnicas de procesamiento de señales digitales de estos casos pueden estar aplicados para mejorar la calidad del discurso. Los ejemplos incluyen tales aplicaciones como la eliminación de reverberación o ecos de discurso, o la remoción de ruido de discurso, o la restauración de discurso registrado en una mezcla de ruido.

2.5.4. Ejemplos de aplicaciones de la comunicación por voz hombre-maquina

En esta sección se expondrán algunas aplicaciones, diversos modos de modelar y los parámetros asociados que se derivan de la utilización de un sistema integrado cuyo propósito es transmitir o extraer información automáticamente de una señal de palabras. Se darán ejemplos representativo de sistemas de procesamiento digital de palabras y técnicas de procesamiento utilizados en tales sistemas. Unicamente se darán ejemplos relacionados con comunicación por voz hombre-maquina.

Sistemas para transmisión digital de voz no están incluidos, aún cuando esta es una de las áreas más grandes de aplicación. Hay varias razones para restringir nuestra atención a la comunicación hombre-maquina. Ante todo, esta área es extremadamente grande en el uso de técnicas de procesamiento digital de discurso, y por lo tanto es ilustrativo de casi todos los métodos de procesamiento, así como extremadamente importante para la creación de nuevas área de aplicaciones, y que muestra un potencial tremendo para su uso en el futuro[18].

Generalmente hay tres áreas mayores hablando de modos de comunicación dentro del área general de comunicación por voz hombre-maquina. Estas áreas incluyen:

1. Expresen sistemas de respuesta
2. Los sistemas de reconocimiento de orador
3. Los sistemas de reconocimiento de discurso.

Expresen sistemas de respuesta son designados para responder a una solicitud para información utilizando mensajes hablados. Así expresan respuestas sistemas de comunicación por voz en una dirección solamente. Por otra parte, las áreas 2 y 3 de la lista anterior son sistemas en que la comunicación está dada por voz humana para manejar o manipular. Para sistemas de reconocimiento de orador la tarea del sistema es verificar una identidad de altavoces (i, e., una decisión si/no en cuanto a si el orador es el

[18] J. L. Flanagan, *"Computers that Talk and Listen: Man-Machine Communication by Voice," Proc. IEEE*, Vol. 64, No. 4, pp. 405-415, April 1976.

que reclama ser), o para identificar el orador de algún conjunto conocido. Así el área de reconocimiento de orador está dividido en las subáreas: de verificación de orador e identificación de orador.

La última área, reconocimiento de discurso, puede estar subdividida en un gran número de subáreas dependiendo de los factores como el tamaño de vocabulario, población de orador, hablar condicionado, etc. La tarea básica de un sistema de reconocimiento de discurso es el reconocer una de otra la expresión exacta (fonética u ortografía de palabras en un texto), o en caso contrario responder con la expresión correcta (responder de la manera correcta como debe de ir la palabra). Los conceptos de entendimiento son de suma importancia en el reconocimiento de expresiones en discursos continuos de vocabulario, mientras que el concepto de reconocimiento exacto es de suma importancia para vocabulario limitado, población de orador pequeña, aislamiento de sistemas de palabra. A continuación se detallan sistemas representativos de cada una de las áreas de comunicación por voz de hombre-maquina. También se detallaran los procesos generales de información del sistema, para comprender la operación del sistema completo.

2.5.4.1. Sistemas de respuesta de voz

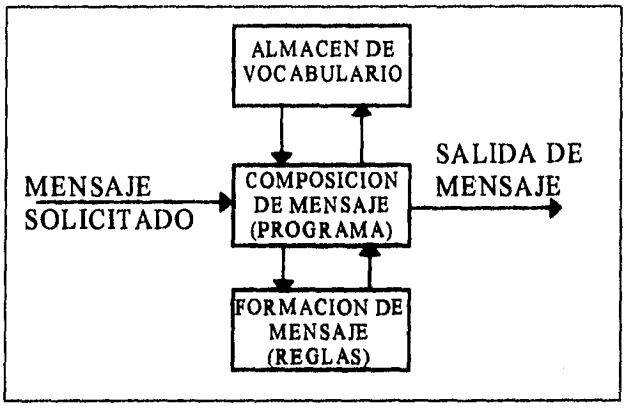


Fig. 2.17. Diagrama de Bloques de un sistema de respuesta por voz

La Fig. 2.17 muestra un diagrama de bloques de un sistema de respuesta por voz de una computadora.

Los elementos de un sistema de respuesta de voz incluye:

1. La provisión para almacenamiento de un vocabulario para el sistema de respuesta de voz.
2. Las reglas para mensajes formados de elementos del vocabulario.
3. Un programa para mensajes de respuesta de voz que componen.

La entrada al sistema de respuesta de voz está en forma de una solicitud de mensaje. El cual puede estar iniciado por otro sistema de proceso de la información o directamente por una información localizadora humana del sistema de respuesta de voz.

Los mensajes de salida están en forma de discurso pronunciado en respuesta al mensaje solicitado. Un ejemplo simple sería un sistema de un directorio telefónico automatizado en que números de teléfono impropiamente marcados estarían detectado, el tipo de problema determinado (e. g., el teléfono ha sido desconectado, o un nuevo número estado asignado, etc.) y una solicitud para un mensaje apropiado estaría enviado a un sistema de respuesta de voz. Para tal sistema las entradas de vocahulario son frases generalmente completas, así como un limitado número de palabras aisladas (e. g., los dígitos con diversas inflexiones habladas).

Como un segundo ejemplo, considere un sistema de recuperación de la información como un sistema de cita de precio de existencias donde un usuario puede teclear en un código a través de un teléfono de sonido de tono, para el precio de unas existencias deseadas. El sistema decodificaría la señal de sonido de tono, determina el precio actual de las existencias, y entonces emite una solicitud al sistema de respuesta de voz el cual crea el mensaje hablado apropiado. Para este caso el vocabulario consistiría en una variedad ancha de palahras y frases.

Hay dos enfoques principales a la instrumentación de un sistema de respuesta de voz. Un enfoque es para intentar construir una máquina con poderes de discurso comparable a un humano. Tales sistemas (frecuentemente referida a discursos sintetizados por reglas del sistemas). En este caso el almacenamiento de vocabulario es esencialmente un diccionario pronunciable y las reglas de formación de mensaje tienen que generar las señales de control requeridas, (e. g., reclamo, intensidad, y parámetros de respuesta de tract curso de vocales) para controlar un discurso sintetizado basado en el modelo de producción de discurso. Tales sistemas son de interés cuando un vocabulario excesivamente grande es requerido. La implementación de tales sistemas es un desafío, sin embargo el mayor problema es el descubrimiento de las reglas para controlar el sintetizador.

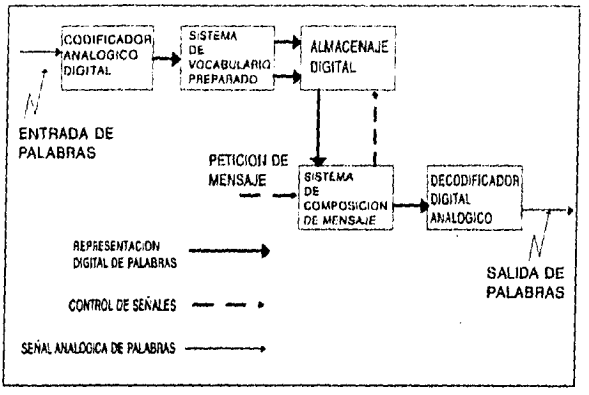


Fig. 2.18. Diagrama a bloques de un sistema digital de respuesta por voz

El segundo tipo de sistema de respuesta de voz de la computadora es el sistema de vocabulario limitado en que el mensaje de salida está creado concatenando elementos de discurso naturales aislados en el almacenamiento de vocabulario. La Fig. 2.18 muestra un diagrama de bloques de un sistema de respuesta de voz digital en que el vocabulario consiste de palabras y frases aisladas que están representadas en forma digital y almacenada en memoria digital. Los mensajes están creados recuperando las palabras y frases requeridas de almacenamiento y reproduciéndolas en la secuencia adecuada. Hay tres consideraciones mayores en la designación de sistemas de respuesta de voz de este tipo. Primero, unos medios de representar y almacenar los elementos de vocabulario básicos tienen que estar seleccionados y un sistema diseñado para permitir el acceso fácil a cada elemento del vocabulario. Segundo, unos medios de grabación de discursos editables seleccionando los elementos de vocabulario deseados que tienen que estar suministrados, junto con unos medios de grabación de los elementos de vocabulario hacia el medio de almacenamiento. El tercer requisito es un sistema para seleccionar y reproducir elementos de vocabulario en secuencia prescrita (ejemplo. el sistema de composición de mensaje).

Desde el objetivo de un sistema de respuesta de voz es para producir discursos expresados que son útiles para comunicación con humanos, la inteligibilidad es la importancia principal. Sin embargo, factores subjetivos como calidad y naturaleza del discurso expresado tiene un gran efecto en la utilidad y aceptabilidad de un sistema de respuesta de voz. Así, es importante que los tres componentes del sistema de respuesta por voz son designados de modo que hay potencial máximo para la producción de un discurso natural sonante altamente inteligible.

2.5.4.1.1. Consideraciones generales en el diseño de un sistemas de respuesta de voz

Los desarrollos en las técnicas digitales para la representación como procesamiento de palabras y en el área de hardware digital ha echo posibles instrumentar un sistema de respuesta de voz utilizando técnicas digitales. Como retratada en Fig. 2.18., un sistema digital requiere primero un codificador analógico-digital; un sistema para obtener una representación digital de una señal de discurso(señal analógica). Así mismo un decodificador digital-analógico apropiado es requerido para la conversión de una forma de representación digital a una señal analógica. Una vez que el vocabulario está representado en forma digital, puede estar almacenado en una memoria digital. El sistema de composición de mensaje es entonces requerido para accesar las entradas de vocabulario en la secuencia correcta y la concatenación de términos en una representación digital del mensaje deseado. Esta representación digital está alimentado a su vez al decodificador digital-analógico.

Dentro de está estructura general, hay tremenda flexibilidad en diseño de sistemas. El factor clave en el diseño de un sistema de respuesta de voz digital es la opción de la forma de representación digital para el discurso expresado que presenta el vocabulario. La opción de método codificador digital tiene un gran impacto sobre la cantidad y tipo de memoria digital que está requerida, y sobre el método de composición de mensaje. En consideración la opción de método codificador digital para aplicaciones de respuesta de voz es de gran ayuda para considerar tres factores:

1. La tasa de información (velocidad de transmisión de bits) requerida para calidad de discurso aceptable.
2. La complejidad de los planes de codificación y decodificación.
3. La flexibilidad de la representación; el potencial para modificación de los elementos del vocabulario.

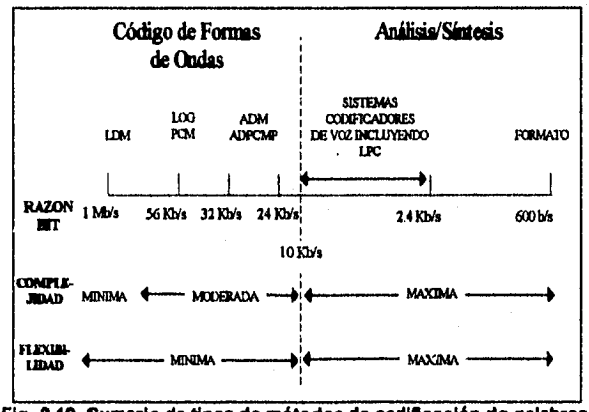


Fig. 2.19. Sumario de tipos de métodos de codificación de palabras.

La Fig. 2.19. muestra una comparación de un número de las representaciones digitales, las cuales están compuestas por tres factores. La representaciones de la codificación de la forma de onda requiere claramente mayor tasa de bit y así requeriría de más almacenamiento digital para un vocabulario de respuesta de voz. Las representaciones de la forma de onda son los más simples en función de los algoritmos de codificación/decodificación. Por otra parte, los sistemas de análisis/síntesis, los cuales literalmente "apartan la señal de discurso, " tiene el mayor potencial para modificar los elementos de vocabulario en modos útiles. Las primeras dos consideraciones, velocidad de transmisión de bits y complejidad de implementación de la representación digital, impacta principalmente en la economía del diseño de toda respuesta de voz digital. Para ilustrar estos intereses permítanos considerar un vocabulario típico, apropiado para muchas aplicaciones, que podrían involucrar cerca de 100 palabras cuya duración promedio podría ser menor que 1 segundo. Así, como estimado relativamente conservador de requisitos de almacenamiento, la Tabla 2.2. muestra el almacenamiento digital requirido por 100 segundos de material de discurso para un rango amplio de velocidades de transmisión de bits. Para el caso de diario de codificación PCM, el requisito de memoria no es irrazonable. La principal interrogante es el costo del almacenamiento digital relativo al costo de codificar y decodificar hardware para sistemas codificadores más complicados. En la última columna de la Tabla 2.2. el valor de 0.1¢ por bit[2] está aplicado a la memoria requerida para unos 100 segundo de vocabulario para obtener un estimado de costo de memoria. Comprendiendo que la memoria pueda estar compartida entre una multiplicidad de salida de canales, puede estar visto que el costo de decodificadores y codificadores digitales tienen que estar mantenidos bajos para que no haga dominante el costo de consideración. Es un claro ejemplo por que la forma de sintetizar del tipó requerida para salida de discurso de alta calidad de producto costaría significativamente más que el costo de la memoria para un vocabulario pequeño.

Método de Codificación	Razón de Bit	Almacenamiento Digital Requerido para 100 segundos (bits)	Costo Aproximado a 0.1¢/bit
PCM	40 Kb/s	4,000,000	$4000.00
ADPCM	24 Kb/s	2,400,000	2400.00
LPC	2.4 Kb/s	240,000	240.00
Formants	0.6 Kb/s	60,000	60.00

Tabla 2.2. Requerimientos de Memoria para almacenamiento por Representación Digital de palabras.

La preparación de vocabulario y edición es básicamente distinto a lo concierne en el diseño de sistemas de respuesta de voz automático. En está área, técnicas digitales tienen el potencial de alta eficiencia y gran flexibilidad en los elementos de vocabulario preparables y salida de voz de alta calidad. Típicamente, las palabras y frases se maquillan

una respuesta de voz de vocabulario están hablado por un orador entrenado y una grabación de audio de alta calidad está hecha. Una palabra o frase está registrada y entonces esta es convertida a una representación codificada analógico/digital. La representación digital (cual puede ser una u otra forma de onda o representación de análisis-síntesis) está entonces almacenada lejos temporalmente en forma digital en la computadora. Un plan automático es requerido para encontrar el comienzo y fin de la expresión de modo que las regiones de silencio que las rodean pueden estar eliminadas. El proceso de grabación puede estar repetido fácilmente hasta una duración satisfactoria una inflexión está obtenido.

Como la etapa final al preparar un vocabulario de respuesta de voz, un plan automático puede comparar las intensidades de todas palabras en el vocabulario y ajustar adecuadamente todos los niveles a algún nivel uniforme o acorde a niveles prescritos determinados por el uso pretendido de cada entrada de vocabulario. Este podría involucrar simplemente un cálculo de amplitud de la señal de pico en una medida más sofisticada de intensidad como energía a corto plazo que puede estar empleado.

Una vez que la palabra o frase está adecuadamente registrada, está almacenada en su área permanente de la memoria de vocabulario. Esto involucra establecer un sistema simple de archivos de discurso y un directorio de direcciones de memoria que están utilizados por el sistema de composición de mensaje para localizar el comienzo y fin de cada entrada de vocabulario.

Los medios dados para preparación y almacenamiento de un vocabulario de palabras y frases, el sistema de respuesta de voz está completado ofreciendo uhos medios para discurso que se componen de expresiones de elementos de vocabulario. Aquí de nuevo, la forma de representación digital tiene un impacto mayor. Si una representación de forma de onda está utilizada, todas ellas necesitan una simple concatenación de las formas de onda de los elementos del vocabulario.

Esto puede guiar para algún discurso sonoro no natural de expresiones que consisten de un vocabulario principalmente de palabras aisladas, pero este enfoque tiene la virtud que el sistema de composición de mensaje pueda ser extremadamente simple. Sin duda un sistema tal puede fácilmente ser instrumentado utilizando un microprocesador. Por otra parte, representaciones basadas en alguna forma de análisis-síntesis de oferta acrecentar flexibilidad para alterar las propiedades de los elementos de vocabulario de modo que el compuesto de la expresión retiene parte de las propiedades de expresión natural (tiempo, inflexión, etc.).

Esta ventaja es potencialmente más importante que las velocidades de transmisión de bits más bajo que son también posibles con representaciones de análisis-síntesis. Debido a que las entradas de vocabulario están representado en función de parámetros fundamentales de producción de discurso, es posible para alterar el reclamo y duración de una palabra para hace adapta un contexto de mensaje particular.

Interesando más aún es la posibilidad de alterar los parámetros de discurso en fronteras de palabra para producir un discurso sintetizado de expresiones sonoras que más discurso natural como conectado. Para hacer este en hasta el simple situaciones requiere reglas para determinar reclamo y tiempo apropiado y algoritmos para alterar los parámetros de discurso para lograr duración de palabra cambia una convergencia de palabras fronteras. Debido a la tasa de información baja de la representación paramétrica, un microprocesador sería adecuado instrumentar un bastante sofisticado mensaje sistema de composición basado en un análisis-síntesis.

2.5.4.2. Un sistema de respuesta de voz digital de múltiple-salida

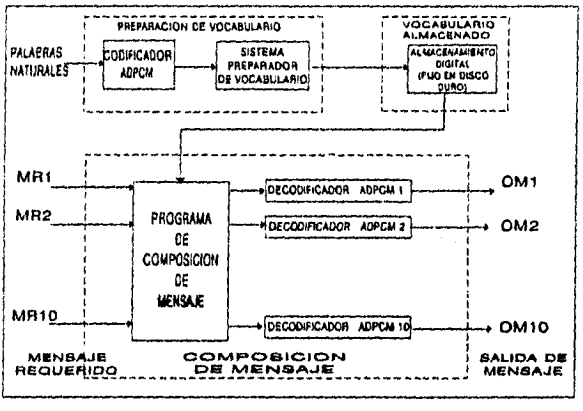


Fig. 2.20. Diagrama a bloques de un sistema digital multilínea de respuesta por voz

La Fig. 2.20 muestra un diagrama de bloques de un sistema multilínea de respuesta de voz digital que ha estado instrumentado en los Laboratorios Bell utilizando una computadora de propósito general pequeña[19,20]. En este sistema, los elementos de vocabulario se representaron utilizando codificación de ADPCM a 24 Kb/s. El codificador y decodificadores de ADPCM estuvieron instrumentados en hardware. Los

[19]L. H. Rosenthal, L. R. Rabiner, R. W. Schafer, P. Cummiskey, and J.L. Flanagan, *"A multiline Computer Voice Response System Utilizing ADPCM Coded Speech", IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-22, No. 5, pp. 339-352, October 1974.

[20]L.R. Rabiner and R. W. Schafer, *"Digital Techniques for ComputerVoice Response: Implementations and Applications", Proc. IEEE*, Vol. 64, No. 4, pp. 416-433, April 1976.

puntos de comienzo y fin de cada palabra estuvo automáticamente localizado por un algoritmo que es basado en el cálculo de la "energía" corta-tiempo de las palabras en-clave de ADPCM[21].

El vocabulario fue almacenado en un disco duro para una rápida integración del programa de composición de mensaje. Esta parte del sistema involucra operaciones y transferencias de datos principalmente lógicas. Un punto importante, sin embargo, es que con la mayoría de las computadoras y sistemas de memoria, existe la posibilidad de utilizar un simple vocabulario de memoria para servir las necesidades de canalización de muchos mensajes. La computadora recibe el mensaje requerido a través de líneas telefónicas o de interfaces digitales directas de otras computadoras. El programa de composición de mensaje localiza los elementos de vocabulario requeridos y recuperan a ellos de memoria. Las representaciones digitales de los mensajes deseados están almacenados en buffers en la memoria de acceso al azar de la computadora. Estos buffers son accesados por los decodificadores de ADPCM continuos a la memoria directa (DMA) cánales. El sistema instrumentado en los Laboratorios Bell puede ser capaz de respuesta de voz simultánea en 10 canales[22][23]. Ha estado utilizado en una variedad de aplicaciones.

La flexibilidad del sistema de respuesta de voz, ha facilitado su uso en una variedad muy amplia de aplicaciones experimentales dentro de Laboratorios Bell. Los siguientes sistemas se han estado instrumentando y hasta la fecha estudiado:

1. Un sistema para instrucciones vocales producibles para equipo de comunicaciones alambrador.
2. Un sistema de ayuda de directorio.
3. Un sistema de cita de precio de existencias.
4. Un sistema de información de prueba de conjunto de datos.
5. Un sistema de información de vuelo.
6. Un sistema de verificación de orador.

[21]L. H. Rosenthal, L. R. Rabiner, R. W. Schafer, P. Cummiskey, and J.L. Flanagan, "A multiline Computer Voice Response System Utilizing ADPCM Coded Speech," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-22, No. 5, pp. 339-352, October 1974.

[22]L. H. Rosenthal, L. R. Rabiner, R. W. Schafer, P. Cummiskey, and J.L. Flanagan, "A multiline Computer Voice Response System Utilizing ADPCM Coded Speech," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-22, No. 5, pp. 339-352, October 1974.

[23]L.R. Rabiner and R. W. Schafer, *"Digital Techniques for ComputerVoice Response: Implementations and Applications." Proc. IEEE*, Vol. 64, No. 4, pp. 416-433, April 1976.

2.5.4.3. Sistemas de reconocimiento de orador

En reconocimiento de orador, las técnicas de procesamiento digitales son frecuentemente el primer paso en la solución de problemas de reconocimiento de patrones. El cual está retratado en la Fig. 2.21.

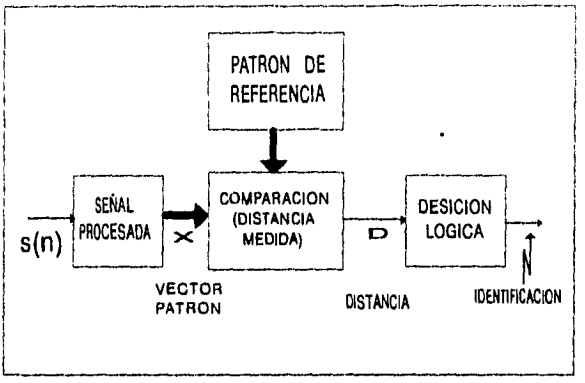


Fig. 2.21. Representación general de un problema de reconocimiento de orador

En esta figura, una representación (vector de patrón) de la señal de discurso se obtiene mediante el uso de técnicas de procesamiento de discurso digitales que preserven los caracteres distintivos de la señal de discurso que son relevantes para la identidad del orador. El consiguiente patrón es comparado con la referencia previa preparada del patrón y lógica de decisión posteriores están utilizadas para hacer una opción entre alternativas disponibles.

Hay dos subáreas distintas en el reconocimiento de orador, la verificación e identificación del orador. Para la verificación de orador una identidad está asegurada por el usuario, y la decisión requerida del sistema de verificación es estrictamente binaria; i,e., aceptar o rechazar la identidad asegurada. Para hacer esta decisión, un conjunto de caracteres distintivos diseñados para retener información esencial acerca de la identidad del orador está medida de una o más expresiones del orador, y las consiguientes mediciones están comparadas (frecuentemente utilizando alguna medida altamente no lineal de comparación) con un conjunto de patrones de referencia almacenados para el orador asegurado.

Así para la verificación de orador solamente se compara únicamente entre el conjunto (o conjuntos) de medidas, y el patrón de referencia es requerida para hacer la decisión final para aceptar o rechazar la identidad asegurada.

Generalmente una medida de distancia entre las mediciones dadas, y la distribución de referencia almacenada está computado. Basados en los costos relativos de hacer los dos tipos posibles de error (i, e, verificar un impostor, o rechazar el orador correcto) un umbral apropiado es establecer la función de distancia.
Está mostrado rápidamente que la probabilidad de hacer los dos tipos de errores descritos anteriormente fue esencialmente independiente de N, la cantidad de patrones de referencia almacenados en el sistema, desde los patrones de referencia por todos los demás altavoces se introducen en formar la distribución estable que caracterise todos los altavoces. En términos más matemáticos, si denotáramos la probabilidad de distribución para el vector de medición x para el i^{th} orador como $p_i(x)$, entonces una regla de decisión simple para verificación de orador podría estar de la forma:

$$\text{identificación de orador } i \text{ si } p_i(x) > c_i p_{av}(x) \qquad Eq.\ 1$$

$$\text{rechazo de orador } i \text{ si } p_i(x) < c_i p_{av}(x)$$

donde c_i es una constante para el i^{th} orador que determina las probabilidades de error para el i^{th} orador, y $p_{av}(x)$ es el promedio (sobre todos los altavoces en el conjunto) distribución de probabilidad para medición x. Variando la constante c_i, un simple control de la mezcla de error entre los dos tipos de errores está obtenido rápidamente.

El problema de identificación de orador difiere significativamente del problema de verificación de orador. En este caso el sistema es requerido para hacer una identificación absoluta entre los altavoces N en la población de usuario.

Así en lugar de una simple comparación entre las medidas y los estados de referencia de patrones, N comparaciones únicas son requeridas. La regla de decisión por tales sistemas está esencialmente de la forma:

$$\text{Escoja orador } i \text{ tal que } p_i(x) > p_j(x), \quad j=1,2,\ldots,N, \quad j \neq i \qquad Eq.\ 2$$

i, e., escoja el orador con la probabilidad absoluta mínima o el error. En este caso el sería posible como el usuario poblacional que es muy grande, la probabilidad de error es un termino a uno desde un número infinito de distribuciones no pueden permanecer distintas en un espacio de parámetro finito- i. e., se pone crecientemente probablemente que dos o más altavoces en el conjunto tendrán distribuciones de medición que son extremadamente cerca unos-a-otros. Bajo estos circunstancias, la confiabilidad de la identificación de orador se pone esencialmente imposible.

Puede estar visto de la discusión anterior que existe una gran cantidad de similitudes, así como diferencias, entre verificación de orador y sistemas de identificación de orador.

Cada proceso requiere de una o mas frases, crear varias mediciones en las frases de prueba, y computar una (o más) funciones de distancia entre el vector de medición y el vector de referencia almacenado. Así, en función de la señal procesada los aspectos de estos dos problemas, los métodos son bastante similares. Las diferencias mayores ocurren en la lógica de decisión. Ahora discutiremos un ejemplo de cada tipo de sistema y espectáculo algunos resultados típicos obtenidos con estos sistemas.

2.6.4.3.1. Sistema de verificación de orador

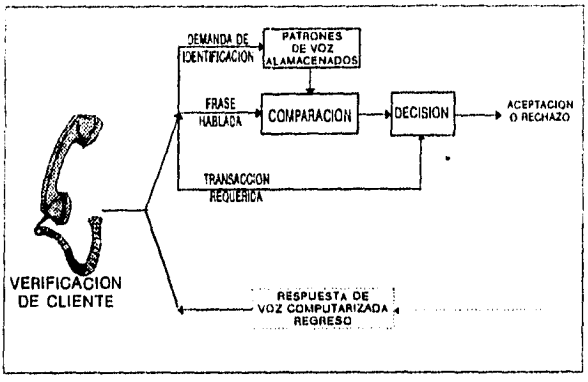


Fig. 2.22. Diagrama a bloques de un sistema de verificador de orador

La Fig. 2.22 muestra un diagrama de bloques de un sistema de verificación de orador digital en línea desarrollado por Rosemberg y otros[24][25]. La persona verificará primero su identidad asegurada, y a petición del sistema de verificación (a través de un sistema de respuesta de voz de la computadora del tipo discutido en anteriormente) le solicitará su frase de verificación, y el tipo de transacción a realizar. Por ejemplo, la transacción solicitada puede ser de acceso a información privilegiada de un banco, etc. Las expresiones habladas están procesadas para obtener patrones que son comparados con los patrones de referencia almacenados para la identidad asegurada y entonces sobre la base de la transacción solicitada determinar la constante de mezcla de error (c_i en Eq. 1) y una decisión para aceptar o rechazar al cliente.

[24]A. E. Rosenberg, *"Automatic Speaker Verification: A Review,"* *Proc. IEEE*, Vol. 64, No. 4, pp. 475-487, April 1976.

[25]A. E. Rosenberg and M. R. Sambur, *"New Techniques for Automatic Speaker Verification,"* *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-23, pp. 169-176, 1975.

2.7. Punto de vista de la tecnología

Elementos claves en la comprensión del sensoreo remoto moderno, incluye las metas generales de los usuarios de sensoreo remoto, la naturaleza del proceso sensorial, y los pasos usados en la transformación de datos sensoriales remotamente a de información útil.

(1) Metas de los usuarios. Tipicamente, los usuarios están interesados en la extracción de la información sensata remotamente, la cual asistirá en la decisión -haciendo relato a las áreas geográficas estudiadas. Mayores submetas a menudo incluyen clasificación de la área y producción de un mapa temático.

(2) El proceso sensorial. Multiespectrales sistemas de sensoreo abordo de aeronaves y naves espaciales, son a menudo usados para medir y grabar la información. Ejemplos son cámaras fotográficas y electrónicas, escaners multiespectrales, y radars. Múltiples imágenes de las áreas inspeccionadas, pueden ser grabadas, de este modo representando la escena en diferentes regiones de los espectros electromagnéticos (ultravioletas, visibles, infrarrojos y microondas) y en diferentes tiempos (diferentes temporadas del año).

(3) Registro geométrico y rectificación. Estas imágenes preprocesan pasos asegurados de múltiples imágenes de la escena que coinciden geométricamente y representan fielmente la geometría de la área terrestre. Una vez que el registro ha sido efectuado, el ensamble de dimensiones relacionadas a cada elemento, resolución terrestre, comprende de un pixel multivariable. El numero de medidas por pixel es referido para así la dimensionalidad de la información de imagen.

(4) Transformaciones. Los componentes de cada pixel puede ser sujetado a un numero de tranformaciones antes que el proceso de extracción de información realmente comience. El propósito de cada transformación puede mejorar la representación visual de la imagen (para facilitar la interpretación manual), para mejorar sus senales-a-propiedades de ruido, o reducir la dimensionalidad de la información.

(5) Un reconocimiento de patrón. Los datos sensados remotamente son usualmente clasificados usando las técnicas de reconocimiento de imagen de patrones; esto es, cada pixel es asignado a una etiqueta la cual describe la categoría de campo cubierto o la condición inferida de sus valores componentes. Los resultados de la clasificación será usada para hacer mapas temáticos o ser registrado dentro de un sistema de información geográfico y sujetado a un nivel muy alto de operación de extracción de información.

(6) Extracción de información geográfica. Muchos de los datos de utilidad derivable de información sensata remotamente son de una naturaleza regional y a menudo

requiere inferencias para ser dibujadas envolviendo varias fuentes de datos usadas en conjunto con clasificaciones de escena. Tales altos niveles de operaciones de extracción de información son referido para colectivizar así extracción de información geográfica. Ellos incluyen mapas de operaciones algebraicas y métodos basados en el arte y ciencia de imagen comprendida.

Posiblemente los aspectos mas característicos de estas aplicaciones de procesamiento de imagen y reconocimiento de patrones son la alta dimensionalidad de la información, la relativa basta resolución de la imagen, y el gran numero de pixeles en una escena (típicamente varios millones). Los datos almacenados a menudo multivariables, consisten de 4, algunas veces 12, posiblemente cientos de variables espectros (y otros) por pixel. Aunque una "escena " puede consistir de varios millones de pixeles, un "objeto" individual en la escena puede consistir de una docena o pocos pixeles y ser únicamente un pixel o a través de su pequeñisima resolución. Sobre estas circunstancias, información extraída desde la imagen remotamente sensata puede ser difícil, costosa y consumir mucho tiempo.
Imagen sensata remotamente es analizada por ambos, técnicas manual y computarizadas.

2.8. Procesamiento de Imagen

La información de imagen producida por sistemas de escaners multiespectrales son usualmente sujetados a numerosas formas de "preprocesamiento" antes de la interpretación manual o digital de la imagen procesada. En el inicio, no hay sustanciales diferencias desde las operaciones usadas en otras aplicaciones de procesamiento de imagen, sin embargo la mezcla de operaciones empleadas es , en curso, un poco especializada.

2.8.1. Ejemplificación de imagen y reconstrucción

En el sensoreo remoto, la materia ejemplificadora resulta en varias formas. En la conversión de la imagen escencial duo-dimensional dentro de un flujo dimensional de información, el sistema sensor debe ejemplificar la escena. El índice optimo en el cual estas ejemplificaciones están funcionando dependen de el campo instantáneo de vista de el sensor y distribución de frecuencia espacial de la escena. Diseñadores de sensores deben entender la naturaleza de esta dependencia en orden a maximizar la captura de detalles de la escena sobre una mano mientras minimizan la carga sobre los canales de sistemas de comunicación en el otro.

Después de que la imagen esta "sobre el terreno", por así mencionarlo, es a menudo necesario reejemplificar la imagen en orden para conducirla dentro de coincidencias con otras imágenes teniendo una diferente estructura cuadricular (diferente escala, orientación, proyección de mapas, etc.). Hay muchos métodos disponibles para efectuar esto; las consideraciones de costos computacionales tienden a manejar la selección del método. El método ejemplificado es lo mas cercanamente reejemplificado. Sobre este método, el pixel de salida en la locación (i,j) es determinada computando la posición de entrada localizada (x,y) y entonces seleccionando el pixel en la actual localización de entrada (m,n), donde m es el la parte entera de x+0.5 y n es la parte entera de y+0.5. Unas técnicas de espectro de interpolación de dos-dimensiones son disponibles, las cuales proveen grandes fidelidades de imagen lo mas cercanamente rejemplificadas, pero estas técnicas requieren esfuerzo computacional considerable. Métodos cúbicos son quizá lo mas popular, previendo un buen compromiso entre fidelidad de imagen y costo computación.

2.8.2. Codificación de Imagen

Encodificación de información de imagen sensata remotamente es necesaria para la transmisión y archivamiento de imagen. En ambos casos en orden a minimizar la calidad de información la cual debe ser tratada. Las consideraciones dominantes son tres: minimizar la transmisión de ancho de banda requerida, realizar la máxima liberación de los degradientes efectos del ruido, y minimizar los efectos los cuales tenderían a degradar el subsecuente análisis. Estas consideraciones imponen contrastes, las cuales grandemente estrechan la selección del método de encodificación.

Los principales métodos usados en el encodificamiento de datos sensados remotamente caen dentro de tres clases:

(1) Encodificamiento directo, en el dominio espacial, por ejemplo, pulsación del código de modulación, modulación delta, y codificación de perdida de ruido.
(2) Técnicas de transformación, por ejemplo, Fourier, Walsh-Hadamard, y transformada coseno.
(3) Encodificación multiespectral, por ejemplo, principales componentes, y encodificación de grupos.

Los mas efectivos métodos de encodificamiento son a menudo híbridos de técnicas de estas tres clases. Hay determinadas y adaptivas versiones de prácticamente cada método. Dentro de limites, es posible intercambiar los grados de compresión encontra de la complejidad de el método. En general, gran complejidad implica mas computación requerida para encodificar y decodificar la información.

Optimos métodos de encodificación toman dentro de consideración la naturaleza estática de las imágenes, las propiedades de los canales de comunicación y / o almacenamiento medio, los procedimientos de análisis a ser aplicados a la información, y los requerimientos de las aplicaciones en el cual los datos son a ser usados.

2.8.3. Procesamiento geométrico

Hay numerosas fuentes de distorsión geométrica en el remoto sensoreo del conjunto de imágenes, relatadas ambas a el sistema sensor y a la naturaleza de la escena, Algunas de estas son bien caracterizadas, a través de cualquiera entendiendo la vista geométrica o preinicio calibración geométrica de el sensor.

Otras causas de distorsión de una mas transitoria y randomiza naturaleza. Un ejemplo de esta aberración cumple para impredecir irregularidades en el posicionamiento del la plataforma sensor oportuna a, por ejemplo, viento o variaciones orbitales menores. En ambos casos, primero es necesario modelar las distorsiones presentes, desarrollar una función imagen-remolque basada en los modelos, entonces aplicar esta función para invertir los efectos de distorsión de la imagen.

Cuando hay conocimiento prioritario de la naturaleza de las distorsiones, las funciones de imagen desviada requeridas pueden ser generadas directamente y aplicadas para remover las distorsiones. Esto es referido como una corrección sistemática.

Sin embargo, muchas de las correcciones geométricas del conjunto de imagen sensata remotamente no son de este tipo sistemático, y los procesos de modelamiento deben ser efectuados a través del uso de puntos de control. Los puntos de control son precisamente coordenadas localizadas en la escena las cuales puede ser identificadas en la imagen. Los puntos de control pueden ser localizados en la imagen por métodos manuales, semiautomáticos o automáticos, involucrando correlación de imagen a imagen. Toscamente hablando, el proceso de corrección involucra desvío o modificación de la geometría de la imagen, así que una red de puntos de control en la imagen son hechos para tener la misma relación espacial como los puntos de control sobre el terreno. (Actualmente, la relación diferirá por una transformación geométrica llamada una proyección de mapa, la cual considera por los factores que la imagen es una representación planar de la superficie esférica aproximada de la Tierra. En la practica, la imagen a ser corregida es usualmente ajustada relativa a otra, ya "rectificada", la imagen de la misma escena.

Arriba del quinto grado de polinomios bivariables típicamente son usados para modelar distorsiones sobre grandes escenas, como tales producidos por el sistema Landsat.

El proceso de corrección actual involucra reejemplificación, como fue descrito en el principio. Estas operaciones son suficientemente complejas par tener garantizado el desarrollo de sistemas de computadora de propósito especial para sus rutinas de ejecución.

Un importante uso de estas técnicas es archivar registros geométricos de múltiples imágenes de la misma escena, a menudo desde el mismo sistema sensor pero colectado en diferentes tiempos. Tales conjuntos de datos multitemporales y multirecursos son provistos para ser de gran valor en el monitoreo de recursos de la Tierra y otras aplicaciones de sensoreo remoto.

2.8.4. Procesamiento Radiométrico

Ajustes radiométricos (intensidad) son aplicados a los grupos de imágenes sensados remotamente, ambos mejoran la fidelidad con la cual la imagen representa la escena y mejora la interpretación de la imagen. Aquí estamos interesados principalmente con la categoría anterior. Ejemplos incluyen corrección atmosférica, compensación por la función punto-esparcimiento de el sensor, y calibración radiante (de brillo).

Aunque los efectos de la atmósfera, especialmente una atmósfera nouniforme, sobre la imagen remotamente sensata puede estrictamente rebajar su interpretatibilidad, correcciones atmosféricas no son rutinariamente desempeñadas. Modelando los efectos de la atmósfera sobre información remota sensata permanece una área de intensiva búsqueda.

Comentarios similares pertenecientes a correcciones por el efecto borroso de la función punto-esparcimiento del sistema sensor. Técnicas para el modelamiento de la función punto-esparcimiento han sido desarrolladas y han mostrado efectividad por mejoramiento de la calidad visual de la imagen, pero esta corrección es únicamente aplicada en la practica, lo mínimo por el costo computacional aplicado.

Calibraciones de brillo son usadas rutinariamente. Una aplicación es para ajustar los efectos de maduración de los sensores detectores y sistemas electrónicos así que, sobre tiempo, un valor de dato dado representa un valor de brillo de escena constante, dentro de las tolerancias de los sistemas prescritos. Otra importante aplicación resulta de el factor que muchos sensores multiespectrales orbitando usan agrupamientos de detectores de estado - sólido por cada banda espectral. Las salidas de todos los detectores en una banda espectral dada debe ser precisamente calibrada relativa a cada otra; de otro modo, la imagen contendrá rayadura horizontal, la cual es cosméticamente molesto y tiene efectos borradores sobre la subsecuentes operaciones de procesamiento de imagen.

2.8.5. Mosaicado

En el mosaicado, las operaciones de ensamblado en conjunto con imágenes de escenas contiguas para formar una imagen contigua, merece especial mención debido a su importancia para muchas aplicaciones de sensoreo remoto. Produciendo un mosaicado de alta-calidad que requiere que las imágenes componentes sean hechas ambos geométrica y radiometricamente compatibles. Muchos de los ajustes descritos anteriormente son usados. Mosaicos de imágenes han sido producidas cubriendo varios estados y hasta el entero Estados Unidos.

2.8.6. Mejoramiento de Imagen

Las transformaciones geométrica y radiometrica de imagen descrita anteriomente son usadas para mejorar la fidelidad con la cual la imagen representa la escena. Muchas otras transformaciones son empleadas que realmente degradan la fidelidad de la imagen pero mejoran los procesos, manual o digital, los cuales son empleados para interpretar la imagen. Los primeros entre estos son filtrados de operaciones usadas para uniformar la imagen, para acentuar márgenes, etc. Estos están ejecutados usando cualquiera filtrado convolucional en el dominio de la imagen o varias operaciones en el dominio de frecuencia espacial. Otros mejoramientos de imagen comúnmente empleados involucran la combinación aritmética de bandas espectrales, combinaciones lineales típicas, radios, y radios de combinaciones lineales de bandas espectrales. Estas transformaciones pueden ser usadas para suprimir variaciones incidentales en la imagen esperada, por ejemplo, diferencias en vista de ángulo, o para acentuar las iteraciones entre bandas espectrales seleccionadas. Estas tranformaciones tiendes a ser altamente una aplicación especifica.

Métodos adicionales usados para extracción de características en conjunto con reconocimiento de patrones son mencionados a continuación.

2.9. Reconocimiento de Patrón

2.9.1. Reconocimiento espectral de patrón

Los conjuntos de imágenes multiespectrales sensadas remotamente, serán sujetadas ya sea a análisis supervisados o no supervisados o, mas tipicamente, un híbrido de los dos planteamientos. El análisis supervisado es usado si un suficiente "verdadero fundamento" es disponible y fácilmente localizable en la información para permitir exacta caracterización de los fundamentos que cubren clases de interés. Por otro lado, el análisis no supervisado usando agrupamientos es empleado para la partición de el espacio característico, después de este el análisis debe identificar la clase de fundamento cubierto correspondiente a cada uno de las clases de agrupamientos (a menudo llamado clases espectrales). Ya que en lo mas mínimo de algunas de las clases en la escena puede no estar contenida en el fundamento o área verdadera, el analista a menudo seleccionara usar una combinación de análisis supervisado y no supervisado.

Ambos métodos de reconocimiento de patrones, estático y no estático, son empleados. Los métodos mas comúnmente utilizados de estos, se presentan a continuación:

2.9.1.1. Clasificador Paralelipipedo (Rebanador de Nivel Multivariable)

Este clasificador paralelipipedo no estático, es el mas a menudo utilizado para el análisis de escena cualitativo, particularmente cuando un buen iteractivo sistema de despliegue de imagen a color es disponible para el entrenamiento del clasificador. En un modo supervisado de operación, el analista localiza unas cuantas regiones en la imagen (por ejemplo, campos agrícolas) representativos de cada uno de los aparentes tipos de campos cubiertos. Para definir los paralelipipedos característicos de espacio, correspondientes a cada clase cubierta de territorio, el sistema determina los valores de información máximos y mínimos en cada uno de las bandas espectrales especificados por cada una de las regiones seleccionadas. Mas que un paralelipipedo puede ser requerido para una clases dada, dependiendo de los grados de heterogeneidad espectral de la clase. El reconocimiento procede asignando un pixel dado a cualquier clase teniendo un paralelipipedo el cual contiene este pixel.

Este clasificador tiene las desventajas de un comienzo fácil de entrenamiento y accesible para la clasificación de una gran área muy rápidamente. Sin embargo, los resultados son altamente propensos a predisponer el análisis y esto es posible probablemente, por pixeles por ser clasificados ambiguamente dentro de mas de una clase de paralelípipedo de diferentes clases puede sobreponerse. Es también posible para algunos pixeles mentir externamente todos definidos paralelipipedos y de tal modo ser imposible por clasificar.

2.9.1.2. Clasificador de distancia mínimo

El clasificador de distancia mínima es el mas simple de los clasificadores estáticos comúnmente usado en las aplicaciones de sensoreo remoto. Los modos de la clase son determinados del entrenamiento de información por ambos, métodos supervisados y no supervisados. Cada pixel es entonces asignado a la clase con el modo de clase mas cercano. Este clasificador requiere una moderada cantidad de información entrenada, que es, suficiente para computar precisos estimados de los modos de clase multivariable, y los resultados que este produce no son óptimos si las matrices de clases covariantes no son equivalentes. Sin embargo, esto es relativamente rápido comparado a otros métodos estáticos y no sufren los problemas de el clasificador paralelipipedo en permitir que los pixeles sean clasificados ambiguamente o no ser clasificados en su totalidad.

2.9.1.3. Clasificador Apariencia, Máximo Gausiano

El clasificador estático de Apariencia o probabilidad- máximo Gausiano, es en alto grado el clasificador mas comúnmente utilizado en el sensoreo remoto. Los vectores significantes y las matrices covariantes de las clases requeridas para computar las funciones de densidad de clase-condicional como parte del proceso de reconocimiento son estimadas por métodos supervisados y no supervisados. Cuando es determinado que alguna de las clases espesamente viola la fundamental suposición Gausiana multivariable, el problema es usualmente repartido con la subdivision de las clases delictivas dentro de subclases, cada una representada por su propio vector significante y matriz covariante. En la practica, las clases tienden a tener una distribución Gausiana, y, además, el clasificador es relativamente tolerante a desviaciones de normalidad.

Las principales desventajas de este clasificador son dos. Porque estáticas de segundo orden son requeridas, mas información entrenamiento son necesitadas para caracterizar las clases adecuadamente de las que son necesitadas por el clasificador de distancia mínimo. El proceso de reconocimiento es comparativamente mas lento y el tiempo de computación varia aproximadamente, así como el cuadrado de el numero de características usado (la variación es lineal para los otros dos clasificadores descritos). Esto puede ser una seria desventaja porque grandes áreas a menudo deben ser

clasificadas, pero donde el uso operacional de este clasificador es hecho, este problema es atacado a través de implementaciones de software inteligente o implementaron sobre un procesador de arreglo.

Generalizaciones de el clasificador probabilidad-máximo Gausiano son empleadas. Cuando estimados razonables de las probabilidades a priori de clase y las funciones de densidad condicional de clase. Cuando algunas clases en la escena son de gran importancia que otras, un estrategia mínima de riego puede ser adoptada junto con los pesos para la clasificación en favor de la mas importantes clases.

No puede ser sobreenfatizado que el éxito de la aplicación del reconocimiento de patrones a información sensoriada remotamente (como con cualquier otro tipo de información) depende críticamente en el uso de un procedimiento clasificador de entrenamiento efectivo. Un procedimiento de entrenamiento es considerado efectivo si este produce caracterizaciones de la clases las cuales son verdaderamente representativos de la escena (as) a ser clasificadas.

2.9.2. Agrupamientos

Agrupamientos o racimos son empleados en el análisis de información de sensoreo remoto para clasificación no supervisada, primariamente como parte de el proceso de entrenamiento. La clasificación no supervisada permite al analista utilizar cualquier estructura natural la cual puede ser presentada en los datos como un auxilio a la apropiada partición de el espacio característico dentro de regiones correspondientes a las clases.

Algoritmos multivariables de agrupamientos de la interactiva variedad "fimdamentos migratorios" , son comúnmente utilizados. Otros son relacionados a histograma de selección cumbre. Muchos de estos algoritmos intentan determinar automáticamente el numero apropiado de clusters(racimos) contenidos en el conjunto de información siendo analizada. Sin embargo, métodos no completamente satisfactorios son conocidos para cumplir esto.

2.9.3. Reconocimiento contextual de patrones

Los métodos de reconocimiento de patrones descritos hasta ahora son especificación de pixel, esto es, que ellos no hacen uso de el contexto de cada pixel en el proceso de clasificación de este pixel. Así en todo información imagen, los pixeles en la información de imagen sensoriada remotamente tienen contexto espacial; en adición, así bien, ellos pueden tener temporal y otros tipos de contexto.

2.9.3.1. Contexto espacial

Muchos diferentes planteamientos fundamentales han sido tomados en el uso de el contexto espacial en el sensoreo remoto. Este rango de simple relación de vecindad incluyendo probabilidades de co-ocurrencia y textura a mas relaciones globales involucran la estructura o sintaxis fotográfica de la imagen.

La textura ha sido la característica espacial local mas intensamente estudiada para aplicaciones de sensoreo remoto. La mas exitosa, pero también la de mas demanda computacional , han sido las relaciones de tono de gris co-ocurrentes. A menudo los valores texturales característicos computados son simplemente adicionados a los componentes espectrales de el pixel y usados conjuntamente con ellos para clasificar el pixel. La textura ha sido también usada para segmentar la imagen en regiones, después cuales otra formas de análisis generalmente siguen.

Por ejemplo, puede ser asumido que cada región en una escena segmentada consiste de pixeles de la misma clase de terreno cubierta. En este caso, clasificación simple puede ser usada para clasificar simultáneamente todos los pixeles en una región dada. Si las homogéneas regiones territorio son, en promedio, grandemente comparadas a la resolución del sensor imagen, verdaderamente resulta en un relativamente gran numero de pixeles por región, entonces la clasificación simple tiene las duales ventajas de ser ambos rápidos y mas precisos que clasificación de pixel-en un tiempo.

Una Teoría de decisión compuesta ha sido invocada para proporcionar una generalización vecindad basada de el mínimo riesgo o clasificación Bayesiana. Este planteamiento es basado en las probabilidades de co-ocurrencia de clases en ves de tonos de grises. Su uso depende de la disponibilidad de suficiente información para caracterizar las frecuencias relativas con cuales clases co-ocurren en una configuración vecindad dada. Las cantidades de computación y memoria requeridas por este tipo de clasificador son sustanciales.
Reconocimiento sintatico de patrones, los cuales usan propiedades estructurales espacial de las clases para caracterizar y reconocer sus ocurrencias, incorporan información contextual de una mas global naturaleza. Este método emplea, gramáticas que debe ser

capaces de generar los patrones en una clase dada; aprendiendo estas gramáticas de ejemplos contenidos en la información disponible, llamada inferencia gramatical, es un problema dificultoso el cual es todavía el objetivo de considerable investigación.

2.9.3.2. Contexto temporal

La disponibilidad de información de repetitiva cubrimiento de misma área terreno hace esto posible para explotar información en el dominio temporal. El planteamiento "vector apilado" a el análisis de información multitemporal es basado en la formación de un vector de dato espectrotemporal para componentes de información concatenados de una secuencia de pasos de sensor. El análisis puede proceder así en el puramente caso espectral. Sin embargo, este planteamiento directo guía la información de muy alto dimensionalidad, así que métodos de extracción de características, tales como los descritos en capítulos anteriores, son muy importantes.

Cambios de escena, ambos cambios en características espectrales o cambios en clasificación, son de inmediato interés en muchas aplicaciones. A menudo imágenes de la misma escena, medidos en diferentes tiempos, son compartidos pixel por pixel; comparaciones regionales son también importantes. Comparación de características espectrales requieren que la información que los datos primero sean normalizados, por ejemplo, calibrados en unidades de radiación o reflectancia, en orden a ser comparados. Cuando las imágenes clasificadas son para ser comparadas, cuidadosamente es requerido que las clases en las diferentes escenas son realmente comparadas. En general, detección de cambio por métodos digitales es muy sensitivo para la exactitud de el registro de las imágenes involucradas.

2.9.3.3. Razonamiento contextual generalizado

En adición a las relaciones espaciales y temporales, otros factores pueden contribuir a el conocimiento contextual acerca de la información sensoriada remotamente a ser analizada. Algunos ejemplos incluyen territorio(topografía), variables climatológicas, e información geopolítica. Estos a menudo pueden tener una influencia significativa sobre la utilidad de la información derivada de la información sensorial remota y seria usada para el alcance posible.

2.9.4. Extracción de característica

Extracción de característica es una aplicación sensorial remota de reconocimiento de patrones que tiene tres principales propósitos:

(1) Reducir la alta dimensionalidad de información imagen multivariable a dos o tres componentes dentro en orden a plasmarlos en papel, traduciéndolos en imagen desde fotográficamente, o viéndolos sobre una pantalla de despliegue de imagen a color;

(2) Reducir la información dimensionalmente en orden a ganar velocidad de procesamiento; y

(3) Extraer y activar rasgos característicos de las clases de interés las cuales serían de otra manera obscurecidas por la alta dimensionalidad de la información o por variaciones de medio ambiente incidentales.

Muchas de las tranformaciones las cuales han sido desarrolladas para este propósito han sido derivadas sobre una bases puramente empíricas. Radios simples y radios de sumas y diferencias de bandas multiespectrales, sobre una bases de pixel por pixel, han sido encontradas para enfatizar características de territorio de particular interés en aplicaciones tales como estudios agrícolas y estudios geológicos. Donde la información multitemporal están disponibles, las trayectorias temporales de algunos de estas características han sido encontrados a ser particularmente usados.

La transformación lineal basada en la expansión Karhunen-Loève (también llamado análisis de componentes principales) han sido encontrados utilizados por mejoramientos de imagen monocroma y de color en una amplio rango de aplicaciones. Esto es usualmente utilizado en un modo no supervisado en el que la transformación es determinada por la información de imagen multivariable sin relacionar a la discriminación de crecimiento cubierto. Otro método, llamado, análisis canonical, el cual usa también ingieanalisis para determinar una dimensionalmente-reducción transformación lineal, es mas usada para maximizar discriminalidad de clase. Este método es usado en un modo supervisado, la transformación es basada en estadísticas de segundo-orden de la clases desarrolladas cubiertas así determinado de la información de entrenamiento.

Otra entera familia de métodos supervisados para optima derivación, transformaciones dimensionalmente-reducientes han sido desarrolladas específicamente en el contexto de aplicaciones sensoriales remotamente. Estos métodos son basados en el concepto de minimización directa de probabilidad de clasificador de error o maximizando una medida de separabilidad estática interclase, tal como divergencia.

Un numero de rasgos espaciales, incluyendo textura y sintaxis (estructura), han ya sido discutidas. Forma, como textura, ha sido caracterizada por un gran numero de diferentes planteamientos , uno de los mas exitosos siendo análisis Fourier de objetos limites. La resultante representación de frecuencia espacial tiende a ser invariante a la rotación y traslación, una extremadamente importante propiedad en aplicaciones basadas en discriminación de forma.

2.9.5. Evaluación de Resultados

El aspecto espacial de una imagen, la cual enriquece su potencial contenido de información, también complica grandemente la tarea de valoramiento de exactitud con la cual ha sido clasificada. Propiamente determinando la exactitud de una imagen clasificada, o, mas particularmente, un mapa temático, requiere ejemplificación estática de la área la cual ha sido clasificada. En general, ambos , un nivel de exactitud y un intervalo de confidencialidad para estos nivel son necesarios. Para mas aplicaciones, estos resultados pueden ser obtenidos solamente en términos de exactitud total, pero en otros resultados precisos son necesitados para algunos o todos las categorías individuales representadas por la imagen o mapa temático.

2.10. Extracción de información geográfica

El procesamiento de imagen y las operaciones de reconocimiento de patrones descritas usualmente no producen el producto final buscado por el usuario. Típicamente, aplicando estas operaciones a la información sensoriada remota provee únicamente un conjunto de la información sobre la cual el usuario debe dibujar en orden para formular un plan o tomar una decisión. La tecnología computacional ha hecho esto posible para integrar información sensorial remotamente dentro de sistemas comprensivos de información geográfica los cuales significativamente automatizan la compilación de información geográfica y facilitan la extracción de información desde tal información.

Sistemas de información geográfica difieren marcadamente de otros sistemas de información en su explícito enfoque sobre entidades espacial y relaciones. El almacenamiento eficiente y manejo de volúmenes extremadamente grandes de información son particularmente ediciones criticas en tales sistemas, así son la fundición de múltiples fuentes de información, las cuales involucran formas múltiples de interfase y formatos de información.

(1) Entrada de información: digitalización manual y automática, encodigación, y edición de información.

(2) Almacenamiento y recuperación: registro y enlazamiento referencial, catalogalización, actualización, procesamiento de dudas.

(3) Análisis: modelamiento de escena, clasificación, cubrimiento, vecindario y computación regional, distancia y medidas de área.

(4) Generación de reporte: sumarización, generación de mapa cubiertos, incluyendo temas y anotaciones, formateo tabular.

CAPITULO III

ALGORITMOS PARA RECONOCIMIENTO

INTRODUCCION

Los modelos de redes neuronales están especificados por la topología de la red, características del nodo, y las reglas de aprendizaje o entrenamiento.

Estas reglas especifican un conjunto inicial de pesos e indica cómo éstos deben ser adaptados para mejorar el desempeño. Tanto los procedimientos de diseño como las reglas de entrenamiento son el tópico de mucha investigación actual. Los beneficios potenciales de las redes neuronales se extienden más allá de las altas tasas de computación proveídas por el paralelismo masivo. Las redes neuronales típicamente proveen más alto grado de tolerancia a las fallas que las computadoras secuenciales de Von Neumann porque hay muchos nodos de procesamiento, cada uno con conexiones primarias locales. El daño a unos pocos nodos o ligas en una u otra manera no necesariamente disminuyen el desempeño general significativamente.

Muchos de los algoritmos de redes neuronales también adaptan pesos de conexión para mejorar el desempeño basados en resultados actuales. La adaptación o aprendizaje es uno de los puntos focales de la investigación de redea neuronales actuales. La habilidad de adaptarse y continuar aprendiendo es esencial en áreas de entrenamiento son limitados y nuevos hablantes, nuevas palabras, nuevos dialectos, nuevas frases, y nuevos ambientes son continuamente encontrados. La adaptación también provee un grado de dificultad para compensar variaciones pequeñas en características de los elementos procesados. Las técnicas de estadísticas tradicionales no son adaptativas, pero típicamente procesan todos los datos de entrenamiento simultáneamente antes de ser usados con nuevos datos.

Las redes de entrenamiento con supervisión tal como la red Hopfield[1] y los perceptrones son usados como memorias asociativas y como clasificadores. Estas redes son proveídas con información lateral o etiquetas que especifican la clase correcta para nuevos patrones de entrada durante el entrenamiento. Muchos de los clasificadores estadísticos tradicionales, tales como clasificadores Gaussianos[2] datos de entrenamiento etiquetados. Las redes entrenadas sin supervisión, tal como las redes formativas de mapas de características de Kohonen[3], son usadas como cuantizadores vectoriales o para formar racimos. La red Hamming[4] es una implementación de red neuronal del clasificador optimo para patrones binarios corrompidos por ruido aleatorio.

3.1. RED HOPFIELD

La red Hopfield[5][6][7], mostrada en la fig. 3.1., es mas apropiada cuando tenemos representaciones exactas binarias son posibles como con imágenes en blanco y negro donde los elementos de entrada son valores pixeles, o con texto ASCII donde los valores de entrada pueden representar bits en el formato ASCII de 8 bits de cada carácter.

[1] J.J. Hopfield, *"Neural Networks and Phisycal Systems with Emergent Collective Computational Abilities"*, Proc. Natl. Acad. Sci. USA Vol. 79, 2554-2558, April 1982.

[2] R.O. Duda and P.E. Hart, *"Pattern Classification and Scene Analysis"*, John Wiley & Sons, Ney York 1973.

[3] T. Kohonen, *"Self-Organization and Associative Memory"*, Springer-Verlag, Berlin 1984.

[4] R.P Lippmann, B.Gold, and M.L. Malpass, *"A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification"*, MIT Lincoln Laboratory Technical Report, TR-769, to be published.

[5] J.J. Hopfield, *"Neural Networks and Phisycal Systems with Emergent Collective Computational Abilities"*, Proc. Natl. Acad. Sci. USA Vol. 79, 2554-2558, April 1982.

[6] J.J. Hopfield, *"Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons"*, Proc. Natl. Acad. Sci. USA, Vol. 91, 3088-3092, May 1984.

[7] J.J. Hopfield and D.W. Tank, *"Computing with Neural Circuits: A Model"*, Science, Vol. 233, 625-633, August 1986.

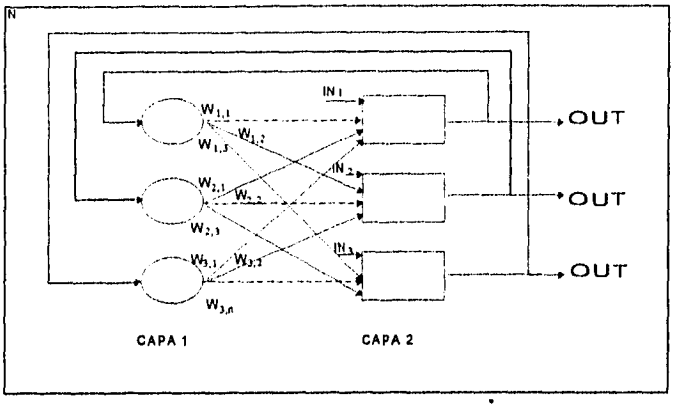


Fig. 3.1. Red Hopfield

Estas redes son menos apropiadas cuando los valores de entrada son realmente continuos, porque un problema fundamental de representación debe ser direccionado para convertir cantidades analógicas a valores binarios. Esta red puede ser usada como una memoria asociativa o para resolver problemas de optimización.

La Fig. 3.1. tiene N nodos conteniendo fuertes no linealidades limitantes y entradas binarias y salidas tomando los valores +1 y -1.

La salida de cada nodo es realimentada a todos los otros nodos a través de pesos. La operación de ésta red es descrita en la tabla 3.1.

1. Asignar pesos.	$tij = \sum_{s=0}^{M-1} x^s ix^s j$	$i \neq j$	o i=1 si $0 \leq i, j \leq M-1$
2. Inicializar con modelos desconocidos de entrada.	$\mu i(0) = xi$	$0 \leq j < N-1$	Donde $\mu i(t)$ es la salida del nodo i en el tiempo t.
3. Itera hasta que converja.	$\mu i(t+1) = fn\left[\sum_{j=0}^{N-1} tij\,\mu i(t)\right]$	$0 \leq j \leq M-1$	
4. Ir al paso 2	El proceso es repetido hasta que los nodos de salida no cambien.		

Tabla 3.1. Algoritmo Hopfield

Primero, los pesos son puestos usando los valores recibidos dados por los patrones de ejemplo de todas las clases. Entonces, un patrón desconocido es impuesto a la red en el tiempo 0 mediante el cual forzará la red a igualar el patrón desconocido. Enseguida de ésta inicialización, la red itera en tiempo discreto usando la fórmula dada.

Se considera que la red converge cuando las salidas no cambian en sucesivas iteraciones. El patrón especificado por los nodos de salida después de la convergencia es la salida de la red.

Hopfield[8] y otros han probado que esta red converge cuando los pesos son simétricos (tij=tji) y los nodos de salida son actualizados asíncronamente usando las ecuaciones de la tabla 1. Cuando la red Hopfield es usada como memoria asociativa, la red de salida, después de la convergencia es usada directamente como la memoria completamente restaurada.

Cuando la red Hopfield es usada como clasificador, la salida después de la convergencia debe ser comparada a los M ejemplares para determinar si ésta corresponde a algún ejemplar exactamente. Si lo hace, la salida es aquella clase cuyo ejemplar correspondió al patrón de salida. Por el contrario, entonces ocurre un resultado de "no correspondencia".

La red Hopfield tiene dos grandes limitaciones cuando es usada como una memoria de contenido direccionable. Primero, el número de patrones que pueden ser almacenados y recordados exactamente está severamente limitado. Si son almacenados demasiados patrones, a un nuevo patrón falso diferente de los patrones ejemplares. Tal patrón falso producirá una salida "no corresponde" cuando la red es usada como clasificador.

Hopfield[9] mostró que esto no ocurre frecuentemente cuando patrones ejemplares son generados aleatoriamente y el número de clases (M) es menor que 0.15 veces el número de elementos de entrada o nodos en la red (N). El número de clases es así mantenido abajo de 0.15 N.

Una segunda limitación de la red Hopfield es que un patrón ejemplar es considerado inestable si es aplicando en el tiempo 0 y la red converge a algún otro ejemplar. Este problema puede ser eliminado y el desempeño puede ser mejorado por un cierto número de procedimientos de ortogonalización.

[8] J.J. Hopfield,*"Neural Networks and Phisyscal Systems with Emergent Collective Computational Abilities"*, Proc. Natl. Acad. Sci. USA Vol. 79, 2554-2558, April 1982.

[9] J.J. Hopfield,*"Neural Networks and Phisyscal Systems with Emergent Collective Computational Abilities"*, Proc. Natl. Acad. Sci. USA Vol. 79, 2554-2558, April 1982.

3.2. RED HAMMING

La red Hopfield es probada frecuentemente en problemas donde las entradas son generadas para seleccionar un ejemplar y cambiar los valores de los bits aleatoria e independientemente con una probabilidad dada[10][11][12]. Este es un problema clásico en teoría de comunicaciones, que ocurre cuando señales binarias de longitud fija son mandadas a través de un canal simétrico binario sin memoria. El clasificador de mínimo error óptimo en éste caso calcula la distancia Hammming al ejemplar para cada clase y selecciona aquella clase con la mínima distancia Hamming. La distancia Hamming es el número de bits en la entrada que no corresponden a los correspondientes bits ejemplares. Una red, la cual será llamada una red Hamming, implementa éste algoritmo usando correspondientes de redes neuronales, como se ilustra en la fig. 3.2.

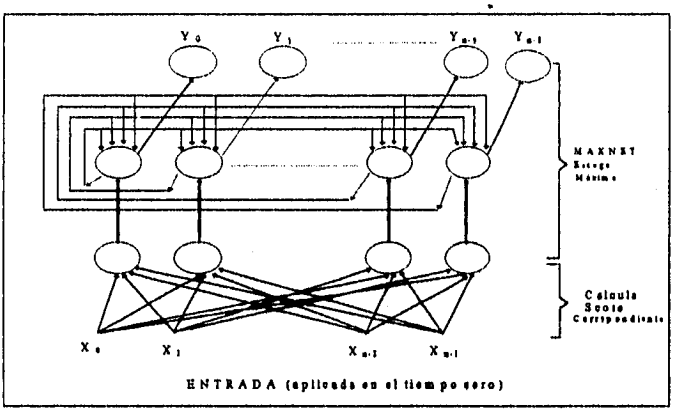


Fig. 3.2. Red Hamming

[10] J.J. Hopfield,*"Neural Networks and Phisyscal Systems with Emergent Collective Computational Abilities"*, Proc. Natl. Acad. Sci. USA Vol. 79, 2554-2558, April 1982.

[11] B. Gold, *"Hopfield Model Applied to Vowel and Consonant discrimination"*, MIT Lincoln Laboratory Technical Report, TRT-747, ADA169742, June 1986.

[12] D.J. Wallace, *"Memory and Learning in a Class of Neural Models"*, in B. Bunk and K. H. Mutter (Eds.) Proceeding of the Workshop on Lattice Gauge Theory, Wuppertal, 1985, Plenum 1986.

La operación de la red Hamming es descrita en la tabla 3.2.

1. Asignar Pesos de Conexión	capa 1=wij=xij/2 capa w: $tij = \begin{cases} 1, & k = 1 \\ -\varepsilon, & k \neq 1 \end{cases}$	0j=N/2 0≤j≤M-1 $\varepsilon < 1/M$ $0 \leq k$ $1 \leq M-1$	wij, tki= peso θ=umbral xij = elemento i = ejemplar j
2. Inicializar con modelo desconocido	$\mu i(0) = ft\left[\sum_{i=0}^{N-1} wijxj - \theta j\right]$	$0 \leq j \leq M-1$	$i(t)^{\mu}$=salida del nodo xi=entrada ft=función
3. Itere hasta que converja.	$\mu j(t+1) = ft\left[\mu j(t) - \varepsilon \sum_{k \neq j} \mu k(t)\right]$	$0 \leq j$ $k \leq M-1$	
4. Ir al paso 2			

Tabla 3.2. Algoritmo de Hamming

Los pesos y los umbrales son primero colocados en la subred inicial de tal modo que los números de puntos correspondientes generados por las salidas de los nodos intermedios de la Fig. b son iguales a N menos las distancias Hamming a los patrones ejemplares. Estos números de puntos correspondientes tendrán un rango desde 0 hasta el número de elementos en la entrada (N) y son más altos para aquellos números correspondientes a clases con ejemplares que mejor corresponden a la entrada.

Los umbrales y pesos en la subred MAXNET son fijos. Todos los umbrales han sido puestos a cero y los pesos entre nodos son inhibitorios con un valor de -ε donde $\varepsilon = 1/M$.

Después de que los pesos y umbrales han sido puestos, un patrón de entrada binario con N elementos es presentado en la parte de abajo de la red Hamming. Debe ser presentado lo suficiente como para permitir que las salidas con números de puntos correspondientes a la primera subred se pongan en su lugar e inicializar los valores de salida de la MAXNET.

La entrada es entonces quitada y la MAXNET itera hasta que la salida de un sólo nodo es positiva. La clasificación es entonces completa y la clase seleccionada es aquella que corresponda al nodo con una salida positiva.

Las simulaciones con diferentes probabilidades de invertir bits en patrones de entrada y con diferentes números de clases y elementos en los patrones de entrada han demostrado que la MAXNET típicamente converge en menos de 10 iteraciones. Además puede probarse que la MAXNET siempre convergerá y encontrará el nodo con el máximo valor cuando $\varepsilon < 1/M$[13].

La red Hamming tiene un número de ventajas sobre la red Hopfield. Implementa el clasificador de mínimo error cuando los errores de bits son aleatorios e independientes, y así el desempeño de la red Hopfield puede ser peor o equivalente a la red Hamming en tales situaciones. Comparaciones entre las dos redes en problemas tales como reconocimiento de caracteres, reconocimiento de patrones aleatorios, y recuperación bibliográfica han demostrado ésta diferencia en desempeño[14]. La red Hamming requiere menores conexiones que la red Hopfield crece como el cuadrado del número de entradas, mientras que el número de conexiones en la red Hamming crece linealmente.

La red Hamming puede ser modificada para ser un clasificador de mínimo error cuando los errores son generados por invertir elementos de entrada de +1 a -1 y de -1 a +1 asimétricamente con diferentes probabilidades y cuando los valores de específicos elementos de entrada son desconocidos.

Finalmente, la red Hamming no sufre de falsos patrones de salida que pueden producir un resultado de "no correspondencia".

3.3. EL CLASIFICADOR GROSSBERG-CARPENTER

Carpenter y Grossberg (3), en el desarrollo de su teoría de resonancia adaptativa, han diseñado una red la cual forma racimos y es entrenada sin supervisión. Esta red implementa un algoritmo de arracimamiento que es muy similar al algoritmo líder selecciona la primer entrada como el ejemplar para el primer racimo. La próxima entrada es comparada al primer ejemplar arracimado. Este "sigue al líder" y es arracimado con el primero si la distancia al primero es menor que un umbral. De otra manera este es el ejemplar para un nuevo racimo. Este proceso es repetido, para todas las siguientes entradas. Así, el número de racimos crece con el tiempo y depende tanto del umbral como de la distancia métrica usada para comparar entradas a ejemplares de racimos.

[13] R.P Lippmann, B.Gold, and M.L. Malpass, *"A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification"*, MIT Lincoln Laboratory Technical Report, TR-769, to be published.

[14] R.P Lippmann, B.Gold, and M.L. Malpass, *"A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification"*, MIT Lincoln Laboratory Technical Report, TR-769, to be published.

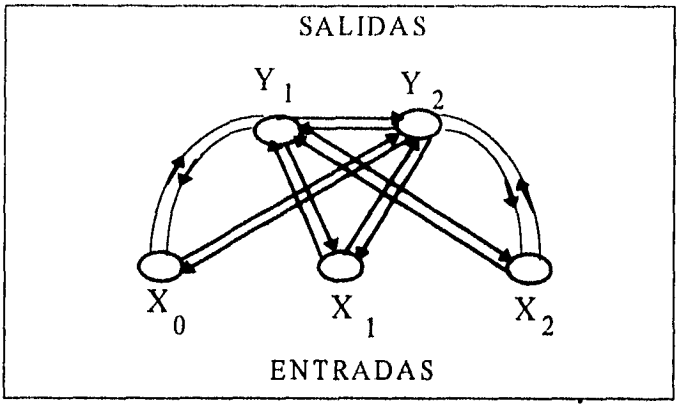


Fig. 3.3. Clasificador Carpenter Grossberg

Los mayores componentes de una red clasificadora Carpenter-Grossberg y con tres entradas y dos nodos de salida esta presentada en la Fig. 3.3. La estructura de esta red es similar a la de la red Hamming. Los puntos que corresponden son computados usando conexiones de alimentación en adelanto y los valores máximos son amplificados usando inhibición lateral formando parte de los nodos de salida. Esta red difiere de la red Hamming en que las conexiones de alimentación en retraso son hechas de los nodos de salida a los nodos de entrada. Los mecanismos también son dados para apagar los nodos de salida con un valor máximo, y comparar los ejemplares a la entrada para la prueba de umbral requerida por el algoritmo líder. Esta red esta completamente descrita usando ecuaciones diferenciales no-lineales, incluyendo la realimentación extensiva, y se ha demostrado que es estable(3).

En operación típica, se puede demostrar que las ecuaciones diferenciales implementan el algoritmo de arracimamiento presentado en la tabla 3.3. El algoritmo asume que el "aprendizaje rápido" es usado como en las simulaciones presentadas en (3) y así los elementos en ambas entradas y ejemplares almacenados toman solo los valores de 0 y 1. La red es inicializada por poner todos los ejemplares representados por pesos de conexión a cero. Además, un umbral de correspondencia llamado "de vigilancia" cuyos rangos están entre 0.0 y 1.0 debe ser puesto. Este umbral determina que tan cerca un nuevo patrón de entrada se asemeja a un ejemplar almacenado para ser considerado similar.

1. Inicialización	tij(0)=1 bij(0)=1//1+N	$0 \le j \le N-1$ $0 \le j \le M-1$ $0 \le p \le 1$	bij, tij = pesos p= umbral de vigilancia
2. Aplicar nueva entrada			
3. Calcular puntajes	$\mu j = \sum_{i=0}^{N-1} bij(t)xi$	$0 \le j \le M-1$	mj=salida del nodo xi= i-ésimo elemento de la entrada
4. Elegir el mejor ejemplar	$\mu j = \max\{\mu i\}$		
5. Prueba de vigilancia	(1) $\lvert x\rvert = \sum_{i=0}^{N-1} xi$ (2) $\lvert T \bullet x\rvert = \sum_{i=0}^{N-1} tijxj$	Si $\frac{\lvert T \bullet x\rvert}{\lvert x\rvert} > p$	Si: Ir a paso 6 No: Ir a paso 7
6. Desabilita el ejemplar mas parecido. La salida del nodo elegido como mas parecido en el paso 4 es temporalmente puesto a cero y no toma parte en la maximización del paso 4. Ir al paso 3.			
7. Adapta el ejemplar mas parecido.	$bij(t+1) = \frac{tij(t)xi}{0.5 + \sum_{i=0}^{N-1} tij(t)xi}$		
8. Ir al paso 2 (habilitar los nodos desabilitados).			

Tabla 3.3. Algoritmo del Clasificador Grossberg-Carpenter

Un valor cercano a uno requiere una correspondencia cercana y valores menores aceptan una correspondencia mas pobre. Nuevas entradas son presentadas secuencialmente en la parte de abajo de la red, igual que en la red Hamming. Después de la presentación la entrada es comparada a todos los ejemplares almacenados en paralelo como en la red Hamming para producir correspondencias. El ejemplar con el más alto número de puntos de correspondencia es seleccionado usando inhibición lateral. Este es entonces comparado a la entrada mediante el cómputo de dividir el producto punto de la entrada y el ejemplar con mejor correspondencia (número de bits en uno en común) dividido por el número de bits en 1 en la entrada. Si esta división es mas grande que el umbral de vigilancia, entonces se considera que la entrada es similar al mejor ejemplar correspondiente y ése ejemplar es actualizado mediante ejecutar una operación AND entre éstos bits y aquellos de la entrada. Si el resultado es menor que el umbral de vigilancia entonces se considera que la entrada es diferente que todos los ejemplares y se almacena como nuevo ejemplar. Cada nuevo ejemplar adicional, requiere un nodo y 2N conexiones para computar números de puntos correspondientes. Este clasificador tiene un problema, puede desempeñarse bien con patrones perfectos de entrada pero que aún una pequeña cantidad de ruido puede causar problemas. Sin ruido, el umbral de vigilancia puede ser puesto de tal manera que los dos patrones que son los más similares, sean considerados diferentes. Con ruido, sin embargo, éste nivel puede ser demasiado alto y el número de ejemplares almacenados puede rápidamente crecer hasta que todos los modelos disponibles hayan sido usados. Son necesarias modificaciones para mejorar el desempeño de éste algoritmo con ruido. Esto puede incluir pesos de adaptación mas lentos y el cambio del umbral de vigilancia durante el entrenamiento y pruebas.

3.4. EL ALGORITMO LMS O ALGORITMO WIDROW-HOFF

El algoritmo LMS o algoritmos Widrow-Hoff[15] es idéntico al procedimiento de convergencia del perceptrón, excepto que la linealidad de límite fijo es hecha lineal o reemplazada por una no-linealidad de lógica de umbral. Los pesos son entonces corregidos en cada prueba por una cantidad que depende de la diferencia entre la entrada deseada y la real. Un clasificador que use el algoritmo de entrenamiento LMS puede usar salidas deseadas de 1 para clase A y de 0 para clase B. Durante la operación la entrada puede ser asignada a la clase A sólo si la salida fué arriba de 0.5. Las regiones de decisión formadas por los perceptrones son similares a aquellas formadas por un clasificador de máxima probabilidad el cuál asume que las entradas no tienen correlación y la distribución para diferentes clases difiere sólo en valores medios. Este tipo de clasificador Gaussiano y los pesos Euclidianos o la distancia métrica Euclidiana es frecuentemente usada en

[15] Widrow, and M.E. Hoff *"Adaptive Switching Circuits"*, 1960 IRE WESCON Conv. Record, Part 4, 96-104 August 1960.

reconocimiento del habla donde hay datos de entrenamiento limitado y las entradas han sido ortogonalizadas por una transformación adaptable[16]. La tabla 3.4. muestra como los pesos y el umbral en un perceptrón pueden ser seleccionados de tal forma que la estructura del perceptrón compute la diferencia entre probabilidades logarítmicas requeridas por tal clasificador Gaussiano[17].

Si $M_{Ai}+\sigma^2_{Ai}$ son la media y la varianza de la entrada Xi cuando la entrada es la de la clase A y $M_{Bi}+\sigma^2_{Bi}$ son la media y la varianza de la entrada Xj para la clase B y $\sigma_{2i}=\sigma^2_{Ai}=\sigma_{Bj}$, entonces los valores requeridos por un clasificador de máximos, son monotónicamente relacionados a:

$$L_A = \sum_{i=0}^{N-1} \frac{(x_i - M_{Ai})^2}{\sigma^2_i} = \sum \frac{x^2_i}{\sigma^2_i} + 2\sum \frac{M_{Ai}x_i}{\sigma^2_i} - \sum \frac{M^2_{Bi}}{\sigma^2_i}$$

y a:

$$L_B = \sum_{i=0}^{N-1} \frac{(x_i - M_{Bi})^2}{\sigma^2_i} = \sum \frac{x^2_i}{\sigma^2_i} + 2\sum \frac{M_{Bi}x_i}{\sigma^2_i} - \sum \frac{M^2_{Bi}}{\sigma^2_i}$$

TERMINO I TERMINO II TERMINO III

Tabla 3.4. Un clasificador Gaussiano implementado utilizando la estructura del perceptrón

Estructuras similares al perceptrón pueden ser también usadas para desempeñar las computaciones lineales requeridas por una transformación karhunen Loeve[18]. Estas computaciones pueden ser usadas para transformar un conjunto de N+K entradas Gaussianas correlacionadas dentro de un reducido conjunto de N entradas no relacionadas que pueden ser usadas con éste clasificador Gaussiano.

Un clasificador de máxima semejanza debe calcular L_A y seleccionar la clase con la mayor semejanza. Ya que el Termino I en éstas ecuaciones es idéntico para L_A y L_B, puede suprimirse. El termino II es un producto de las veces de la entrada de los pesos y puede ser calculado por un perceptrón y el término III es una constante que puede ser obtenida del umbral en un nodo del perceptrón. Un clasificador Gaussiano puede entonces ser formado por utilizar el perceptrón estándar para calcular L_A-L_B por hacer:

16 T. Parsons, *"Voice and Speech Processing"*, Mc Graw Hill, New York 1986.
17 R.O. Duda and P.E. Hart, *"Pattern Classification and Scene Analysis"*, John Wiley & Sons, Ney York 1973.
18 T. Parsons, *"Voice and Speech Processing"*, Mc Graw Hill, New York 1986.

$$W = \frac{2(M_{Ai} + M_{Bi})}{\sigma^2_i}$$

y

$$= \sum_{i=1}^{N-1} \frac{M^2_{Ai} + M^2_{Bi}}{\sigma^2_i}$$

3.5. EL ALGORITMO DE BACKPROPAGATION

El algoritmo de Backpropagation descrito en la tabla 3.5. Es una generalización del algoritmo LMS. Usa una técnica de búsqueda de gradiente para minimizar una función costeada igual a la diferencia cuadrática media entre las salidas de la red deseada y las reales. La salida deseada de todos los nodos es normalmente "Baja" (0 o 0.1) a menos que ése nodo corresponda a la clase correspondiente a la entrada actual, en cuyo caso es "Alta" (1.0 o 0.9).

1. Inicializa pesos con pequeños valores random.	
2. Presenta entrada y salida deseada.	
3. Calcula salida actual, utilizando función de no linealidad (compresora).	
4. Adapta los pesos.	$= OUT(1-OUT)(TARGET-OUT)$ $\Delta w_{pq,k} = \eta \delta_{q,k} OUT_{p,j}$ $w_{pq,k(n+1)} = w_{pq,k(n)} + \Delta w_{pq,k}$
5. Adapta los pesos de las capas ocultas.	$_{p,j} = OUT_{pj}(1-OUT_{pj})(\sum \delta_{p,k} w_{pq,k})$
6. Ir al paso 2.	

Tabla 3.5. Algoritmo de Backpropagation

La red es entrenada por seleccionar inicialmente pequeños pesos aleatorios y umbrales internos y entonces presentar todos los datos de entrenamiento repetidamente. Los pesos son ajustados después de cada prueba usando la información lateral especificando la clase correcta hasta que los pesos converjan y la función costeada es reducida a un valor aceptable. Un componente esencial del algoritmo es el método iterativo descrito en la Fig. 3.6.

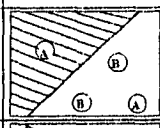
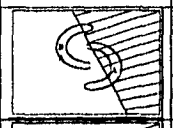
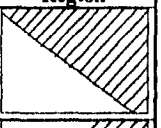
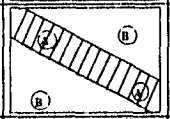
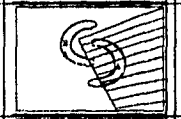
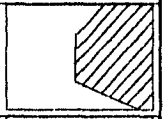
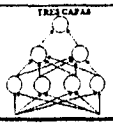
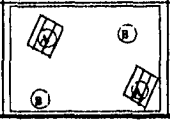
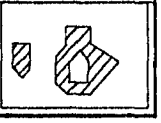
Estructura	Tipo de Región de decisión	Problemas exclusivos	Clases con Regiones	Muestra General de Región
UNA CAPA 	Half Plane Sounded by hyperplane			
DOS CAPAS 	Converge, abre y cierra la región.			
TRES CAPAS 	arbitraría (complementa el límite del número de nodos)			

Fig. 3.6. Tipos de regiones de decisión que pueden ser formados por perceptrones unicapa y multicapa con una y dos capas de unidades ocultas y dos entradas.

3.6. PRINCIPIOS DE LOS MODELOS ADALINE Y MADALINE

Los métodos de procesamiento de señales tuvieron su desarrollo dentro de la rama de la ingeniería en la llegada de una nueva rama en la electrónica, la comunicación. En sus inicios, para llevar a cabo este tipo de comunicación se requería de filtros analógicos, los cuales consistían de circuitos RLC (resistor-inductor-capacitor) que fueron diseñados con el propósito de eliminar el ruido que existía en las señales de comunicación.

Hoy en día, los sistemas de procesamientos de señales junto con sus avances se han involucrado dentro de una tecnología multifacética, haciendo énfasis principalmente en la evolución que hubo desde la implementación de los primeros circuitos sintonizadores (RLC) hasta la llegada de los procesadores de señales digitales (DSP's).

Las bases para las industrias de la comunicación residen en el diseño e implementación de filtros, cuya función es la de eliminar el ruido existente en las señales de información generadas y/o recibidas por ellas.

Para el desarrollo de los procesadores de señales digitales se penso en varios modelos de redes neuronales artificiales, de las cuales se tomo de acuerdo a sus características el primer perceptrón, que consistía de dos niveles, por lo que las investigaciones no se detuvieron con el desarrollo de este perceptrón y optaron por el diseño de un perceptrón mas preciso, el cual consistió de múltiples niveles. Dentro de este conjunto de perceptrones Bernard Widrow y un grupo de investigadores tomaron principalmente dos modelos para su análisis y desarrollo, a los cuales denominaron perceptrones Adaline y madeline. El Adaline (ADAptive LINear element "Elemento Lineal Adaptivo") también es conocido como Unidad Umbral Lineal y constituye el bloque básico de la estructura de una red Madaline (many Adaline "Muchas Adaline"). De acuerdo al modelo Adaline, su estructura se constituye de una sola neurona localizada en el nivel F2 en tanto el modelo Madeline posee cualquier número mayor de una neurona en el mismo nivel, como se aprecia en la figura 3.7.

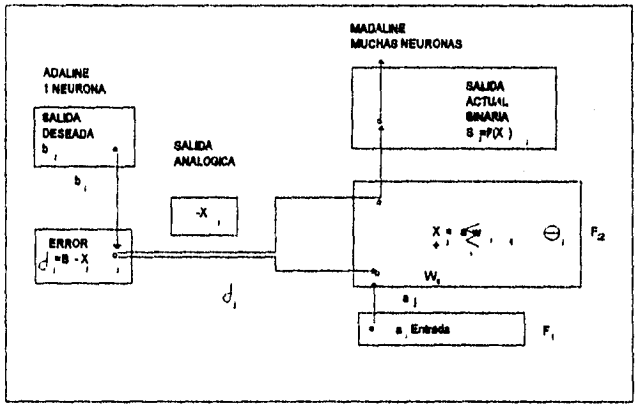


Fig. 3.7. Comparación entre el modelo ADALINE/MADALINE y el Perceptrón

En la figura 3.7. se puede observar que las áreas resaltadas muestran la diferencia principal entre los modelos Adaline/Madaline y el Perceptrón de alimentación hacia

adelante de Rosenblatt, que consiste en que el modelo Adaline/Madaline compara la salida analógica X_j que genera la red con la salida b_j que se pretende obtener. Esta comparación proporciona una indexación de errores derivada de la relación entre una salida binaria y una salida deseada. De acuerdo a la figura 3.7., el error b_j-x_j=d_j representa la retroalimentación que sirve para ajustar los pesos, que utilizando la regla de corrección de Error de retroalimentación de Rosemblatt[19] queda de la siguiente forma:

$$\frac{dw_{ij}}{dt} = \alpha \delta_j \frac{a_i}{|a|} \quad \text{..... (1)}$$

Esta regla minimiza el error cuadrático medio representado por:

$$\sum_j \delta_j \quad \text{..... (2)}$$

con la regla anterior obtenemos el promedio de todas las entradas.

Los modelos Adaline y Madaline nos proporcionan muchos ejemplos sobre los avances tecnológicos generados por las investigaciones de las redes neuronales artificiales.
A continuación daremos una descripción mas amplia acerca de los modelos básicos del Adaline y Madaline.

3.6.1. Adaline

Combinador Lineal Adaptivo.

La red neuronal artificial denominada Adaline, es un dispositivo conformado únicamente por un elemento de procesamiento, por lo tanto, en términos técnicos, el Adaline no es considerado como una red neuronal artificial, sin embargo, aunque pequeña, su estructura es muy importante, debido a que se forma de un Combinador Lineal Adaptivo, por lo que el estudio de este tipo de redes, requiere previamente de un análisis de estos combinadores.

19 F.Rosenblatt. *"Principles of Neurodynamics: Perceptrons and the Theory of brain Mechanism"*. Washington, D.C.: Spartan Books, 1962.

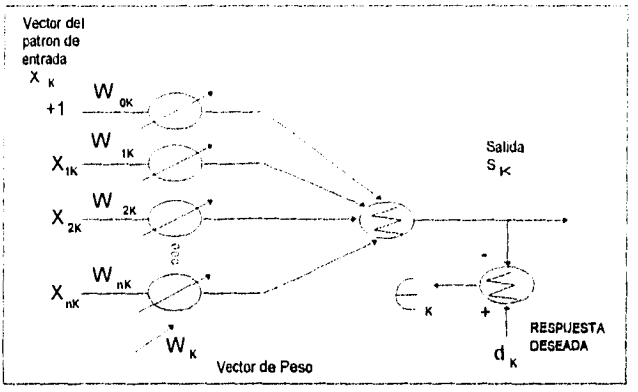


Fig. 3.8. Esquema de un combinador Lineal Adaptivo

En la figura 3.8. se observa un Combinador Lineal Adaptivo, en el que la salida representa la combinación lineal de las entradas.
Ahora bien, si implementamos este combinador a un sistema de tiempos discretos, recibirá en el tiempo marcado como K un vector con la señal de entrada, representado por la expresión:

$$X_K = [X_{0K}, X_{1K}, X_{2K}, ..., X_{nK}] \quad \text{..... (3)}$$

y una respuesta deseada dk; en tanto, una entrada especial se utiliza para efectos de entrenamiento y aprendizaje. El vector de entrada esta compuesto principalmente por un conjunto de pesos marcados con sus respectivo coeficiente, que se expresa por:

$$W_K = [W_{0K}, W_{1K}, W_{2K}, ..., W_{NK}] \quad \text{..... (4)}$$

entonces la suma de las entradas y los pesos es procesada, produciendo así una salida lineal representada por el producto:

$$S_K = X_K W_K \quad \text{..... (5)}$$

de donde los componentes de X_n pueden tomar cualquier valor analógico continuo o en su defecto valores binarios. Por su parte, los pesos son variables, esencialmente, son considerados como variables continuas y pueden tomar valores ya sean positivos o negativos. Durante el proceso de entrenamiento de la red, tanto los patrones de entradas como las correspondientes respuestas deseadas, se presentan ante el combinador lineal. Después, un algoritmo de entrenamiento ajusta de manera automática los pesos, de

manera que las respuestas de salida puedan ser consideradas como las respuestas deseadas.

Para esto, tenemos que en las aplicaciones en donde intervengan procesamiento de señales, el método mas común para la adaptación de pesos es el Algoritmo LMS (Least Mean Square) , que frecuentemente es llamada Regla Delta Widrow-Hoff[20]. Dicho algoritmo tiene como tarea minimizar el total de la suma de los cuadrados de los errores lineales, para esto, el error lineal E_k se encuentra definido por la diferencia entre la respuesta deseada d_k y la salida lineal S_k durante el tiempo K, siendo esto necesario para la adaptación de los pesos.

Cuando el combinador lineal adaptivo esta implantando en una red neuronal artificial de arquitectura multinivel, por lo menos una señal de error esta disponible, aunque no de manera directa, por cada combinador lineal individual, y muchos de los procedimientos deberán ser proporcionados para adaptar sus vectores de pesos.

ADALINE.

El bloque básico de construcción que es frecuentemente usado para el diseño de las redes neuronales artificiales, es el Elemento Lineal Adaptivo mejor conocido como ADALINE[21], que en sus principios fue llamado Neurona Lineal Adaptiva, como se observa en la figura 3.9.

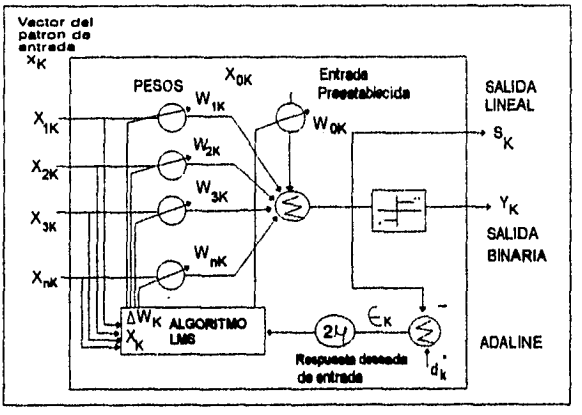


Fig. 3.9. Red neuronal artificial con arquitectura ADALINE

[20] D.E. Rumelhart and J.L. McClelland. *"Learning Internal Representations by Error propagation"*. In Parallel Distributed Processing. vol. 1. ch. VIII. Eds. Cambridge, Ma:M.I.T. Press, 1986.
[21] B.Widrow and M.E. Hoff. Jr. *"Adaptive Switching Circuits"*. IRE Western Electric Show and Convention Record, Part IV, 1960.

Muchos de los investigadores coinciden en que la estructura del ADALINE es semejante a la descripción general de un elemento de procesamiento, para esto, existen 2 modificaciones que se requieren para hacer que la estructura de dicho elemento de procesamiento cumpla con las características de una red ADALINE, las cuales son:

1.- Adicionar al conjunto de conexiones que llegan al combinador lineal adaptivo una conexión mas, con el peso marcado como W_0, el cual siempre tendrá un valor de entrada igual a uno.

2.- Esta consiste en la colocación de una condición bipolar sobre la salida; dicha condición, funciona de la siguiente forma: si la salida del combinador lineal adaptivo es positivo, la salida dela red tendrá un valor igual a +1, si por el contrario, la salida del combinador lineal adaptivo es negativo, la salida al final de la red será -1.

Estas características se aprecian en la figura 3.10.

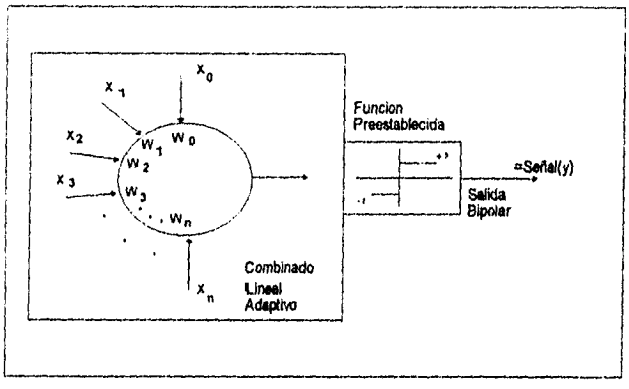


Fig. 3.10. Elemento de procesamiento modificado para funcionar como una estructura ADALINE

Para comprender mejor el funcionamiento de una red neuronal de tipo ADALINE, tomaremos como referencia la red neuronal artificial que se observa en la fig. 3.9., la cual es capaz de implementar ciertas funciones lógicas.

En ella tenemos 2^n posibles patrones de entrada; para esto, se pretende que una implementación lógica general sea capaz de clasificar cada patrón, cualquiera que sea su valor, +1 o -1, de acuerdo a la respuesta deseada. De esta manera tenemos 2^{2n} posibles funciones lógicas que conectan las n entradas a una sola salida binaria.

Debido a lo anterior, una red neuronal de tipo ADALINE tiene la capacidad de procesar únicamente un pequeño subconjunto de estas funciones, las cuales son conocidas como Funciones Lógicas Separables Linealmente o Funciones Lógicas Preestablecidas[22], que pueden ser obtenidas mediante todas las variaciones posibles sobre los pesos.

Por otro lado, tenemos un caso particular, en el cual el elemento llamado ADALINE posee únicamente 2 entradas, como se observa en la figura 3.11. Así también, en la figura 3.12. podemos apreciar todas las posibles entradas binarias a dicho elemento, representadas por cuatro puntos extensos, localizados dentro del espacio correspondiente al vector del patrón de entrada. En este espacio, las componentes de dicho vector se encuentran situados a lo largo de los ejes de coordenadas.

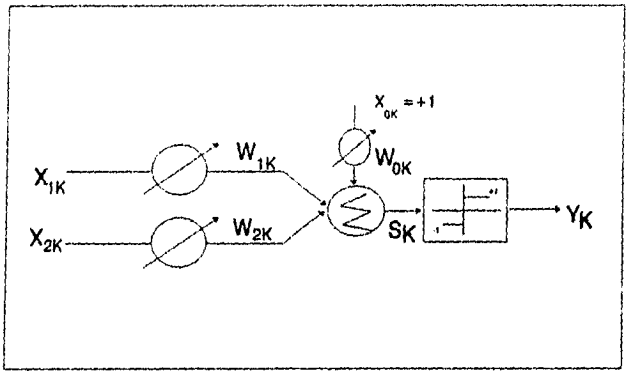


Fig. 3.11. Elemento ADALINE con 2 entradas.

22 B. Widrow and M.E. Hoff. Jr. *"Adaptive Switching Circuits"*. IRE Western Electric Show and Convention Record, Part Iv, 1960.

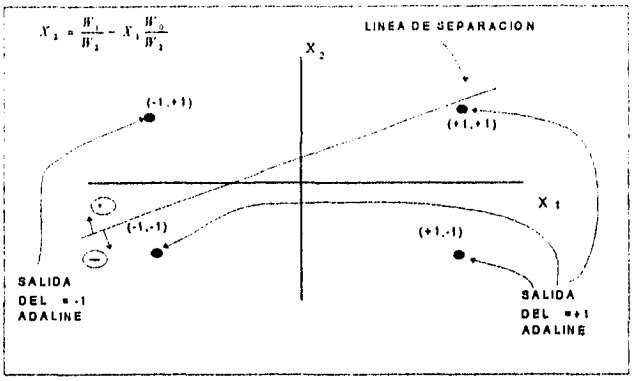


Fig. 3.12. Número posible de entradas binarias para el elemento ADALINE de 2 entradas.

De acuerdo alas figuras anteriores, durante su funcionamiento, el elemento ADALINE lleva a cabo una tarea muy importante, que consiste en separar a los patrones de entrada en 2 categorías que depende de los valores que tengan los pesos. Para esto, tenemos que una condición preestablecida en estado crítico se presenta cuando el valor de la salida lineal es igual a cero, es decir:

$$S = X_1W_1 + X_2W_2 + W_0 = 0 \quad \text{..... (6)}$$

de la ecuación anterior tenemos que:

$$X_2 = -\frac{W_1}{W_2}X_1 - \frac{W_0}{W_2} \quad \text{..... (7)}$$

siendo esta relación lineal, apreciada en la figura 2.32., la cual contempla tanto una línea inclinada de separación como una intersección, representadas ambas por:

$$Pendiente = -\frac{W_1}{W_2}$$
$$Interseccion = -\frac{W_0}{W_2} \quad \text{..... (8)}$$

Los pesos marcados como W_0, W_1 y W_2 son los que determinan la pendiente, la intersección y el lado de la línea de separación que corresponde a la salida positiva, en tanto en el lado opuesto, la línea de separación corresponde a la salida negativa.

De acuerdo a las expresiones anteriores decimos que en las redes ADALINE que contemplan en su estructura cuatro pesos, su límite de separación será siempre en plano, no siendo así para las ADALINE que contemplen mas de cuatro pesos, ya que en este caso su límite de separación será un hiperplano. Como lo señalamos anteriormente, cuando el peso adicional marcado como Wo toma el valor de cero, tenemos que la separación hiperplana será homogénea, es decir, pasara dentro del espacio del vector del patrón y por el origen.

De la figura 3.12. obtenemos que los patrones binarios de entrada están clasificados de la siguiente manera:

(+1,+1) ---> +1
(+1,-1) ---> +1
(-1,-1) ---> +1
(-1,+1) ---> +1

De acuerdo a los resultados anteriores observamos que la red ADALINE cumple con la función de separabilidad lineal. Por el contrario, si no cumpliera con esta función, estaríamos hablando de que lleva a cabo una función NOR exclusiva con 2 entradas, y en la cual los patrones binarios de entrada estarían clasificados de la siguiente forma:

(+1,+1) ---> +1
(+1,-1) ---> -1
(-1,-1) ---> +1
(-1,+1) ---> -1

De esta manera, observamos que una red ADALINE de estructura simple sin una fase de preprocesamiento, no tiene la capacidad de implementar una función del tipo NOR exclusiva.

En tanto una red ADALINE sencilla pero con 2 entradas, es capaz de realizar 14 de las 16 posibles funciones lógicas; que si por el contrario, la red posee un número mayor de entradas, únicamente una pequeña fracción de las funciones lógicas se podrán realizar, pero siempre siendo estas linealmente separables.

Finalmente tenemos que las combinaciones de los elementos o también llamados nodos de los elementos, tiene su mayor aplicación en las redes cuyas funciones no son linealmente separables.

3.6.2. Madaline

Como lo analizamos en el tema anterior, uno de los principales problemas en el desarrollo de las redes, se presenta en las redes ADALINE de 2 niveles, debido a que estas no pueden procesar funciones como es el caso de la OR exclusiva.

Para solucionar este problema surgieron los primeros modelos de redes neuronales artificiales cuyos niveles se encontraban compuestos por elementos adaptivos múltiples, es decir, una combinación de redes ADALINE, como se observa en la figura 3.13.; estos modelos de redes neuronales artificiales fueron llamados MADALINE (Many ADAline "Muchos ADALINE") por Widrow[23] y Hoff[24].

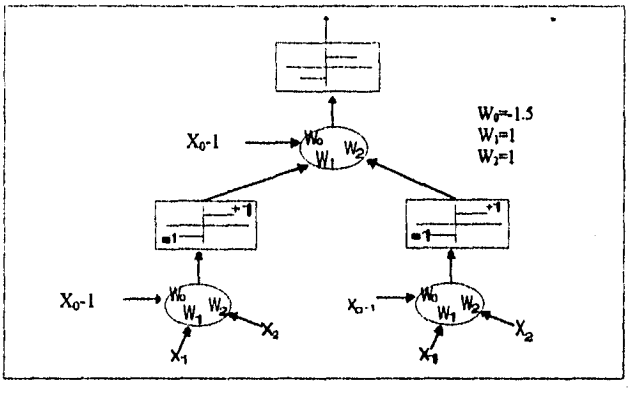


Fig. 3.13. Red neuronal artificial con arquitectura MADALINE

Algunos análisis matemáticos enfocados al estudio de la estructura del MADALINE fueron desarrollados en las tesis de Ridway[25], Hoff y Glanz[26], en las que coincidieron en que las arquitecturas multinivel MADALINE eran las mas aproximadas a lo que es la estructura real de una red neuronal artificial, como la que se aprecia en la figura 3.14.

---

[23] B.Widrow. *"Generalization and Information Storage in Networks of ADALINE Neurons"*. Eds. Washington, D.C.: Spartan Books, 1962.

[24] M.E. Hoff, Jr. *"Learning Phenomena in Networks of Adaptive Switching Circuits"*. Ph D. Thesis, Techs. Rep., Stanford Electron Labs., Stanford. C.A. May., 1965.

[25] W.C. Ridgway III, *"An Adaptive Logic System with Generalizing properties"* Ph. D. Thesis Tech. Rep., Stanford Electron Labs., Stanford, C.A. April, 1962.

[26] P.H. Glanz *"Statiscal Extrapolation in Certain Adaptive Pattern-Recognition Systems"*. Ph. D. Thesis, Tech. Rep., Stanford Electron, Labs., Stanford, CA., May, 1965.

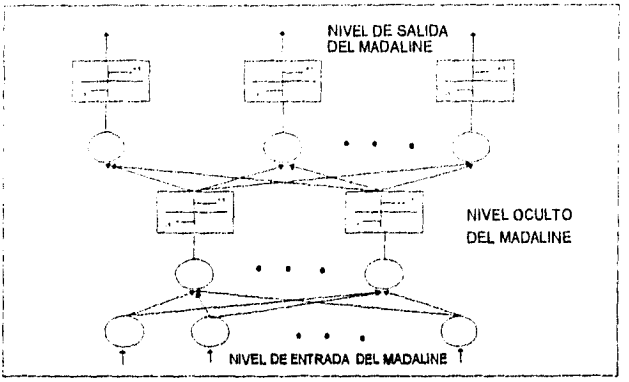


Fig. 3.14. Estructura real de una red neuronal artificial formulada con MADALINES

De acuerdo a la figura anterior, tenemos que la red MADALINE podría ser presentada con un vector de entrada de una dimensión extensa, como por ejemplo, los valores de los pixeles de una imagen rastreada pueden ser tomados como los valores de los patrones de entrada. En este caso, con un entrenamiento adecuado, la red podría enviar como respuesta un 1 binario a cualquiera de los diferentes nodos de salida, en donde cada respuesta corresponde a una de las porciones de la imagen de entrada. Para un patrón de entrada, un nodo deberá tener una salida con valor +1, si el patrón de entrada corresponde al de la clase representada por dicho nodo, de manera que los nodos restantes que forman a la red deberán tener un valor de -1.

Cuando una red neuronal de este tipo ha sido entrenada con una identificación previa de los patrones de entrada, el vector de salida deseado es ya conocido; lo que realmente es desconocido, es el valor de la salida deseada para un nodo conocido el cual se encuentra localizado en uno de los niveles ocultos.

Durante el desarrollo de estas redes, tenemos que al principio de la década de los 60's se construyó una red con arquitectura MADALINE formada por 1000 conexiones de peso; la cual fué utilizada en el área de reconocimiento de patrones; en ella, los pesos que se utilizaron para su elaboración fueron fabricados con resistores, que son resistencias eléctricamente variables; este tipo de redes fueron desarrolladas por Widrow y Hoff, y cuya configuración fue la siguiente:

Las entradas que son introducidas por la retina se conectan a un nivel formado por elementos ADALINE adaptivos, en tanto, las salidas eran conectadas a un dispositivo lógico estable, el cual finalmente generaba la salida del sistema. Un ejemplo de este tipo

de redes se muestra en la figura 3.15., en donde se conectan dos redes ADALINE a un dispositivo lógico tipo AND para así proporcionar una salida.

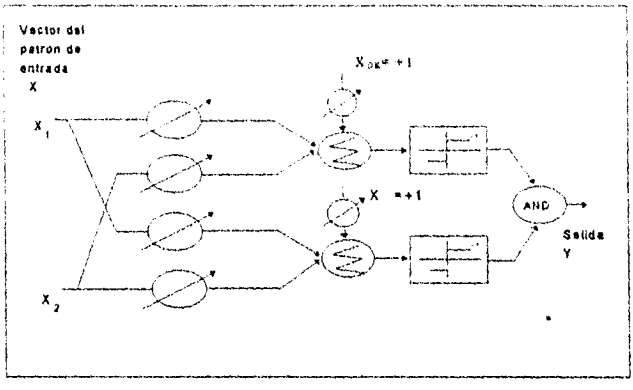


Fig. 3.15. Red neuronal MADALINE formulada con 2 elementoe ADALINE conectados a un dispositivo lógico de tipo AND.

Con una adecuada selección de los pesos, obtenemos que el límite de separación dentro del espacio del patrón de entrada para la red neuronal ilustrada en la figura 3.15., deberá ser similar a la que se observa en la figura 3.16.

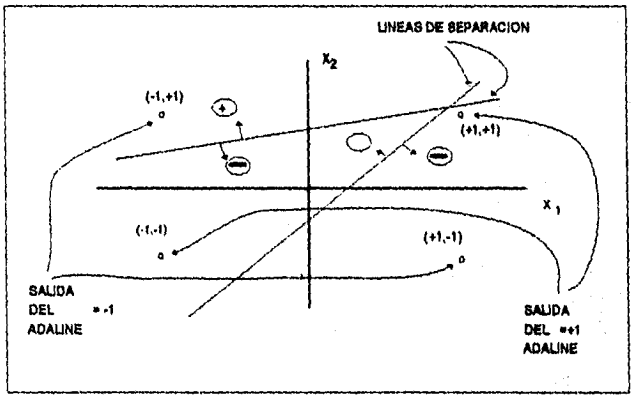


Fig. 3.16. Gráfica donde se observa el límite de separación para la red de la figura 3.15.

Es importante mencionar que este limite de separación es capaz de implementar la función NOR exclusiva. Muchos de estas redes fueron construidas con un numero mayor de entradas, así como los elementos ADALINE en el primer nivel, y un numero variado de dispositivos lógicos estables, de entre los que destacan el dispositivo AND, el OR y el de la Mayoría de Elementos Tomados del segundo nivel. Estos tres dispositivos o funciones son consideradas como funciones lógicas preestablecidas.

Para las redes neuronales artificiales que involucran los valores de pesos conocidos, se relacionan con cualquiera de las tres funciones anteriores, pero con un factor muy importante, que las selecciones de los pesos no son las únicas; esto se puede apreciar en la figura 3.17. También tenemos que durante los años 60's las redes de tipo MADALINE tenían una estructura que consistía de elementos adaptivos en el primer nivel (de entrada), y de funciones preestablecidas estables en el segundo (de salida), aunque hoy en día las redes neuronales con alimentación hacia adelante, frecuentemente involucran en su estructura muchos niveles, y regularmente la mayoría de estos niveles son del tipo adaptivo.

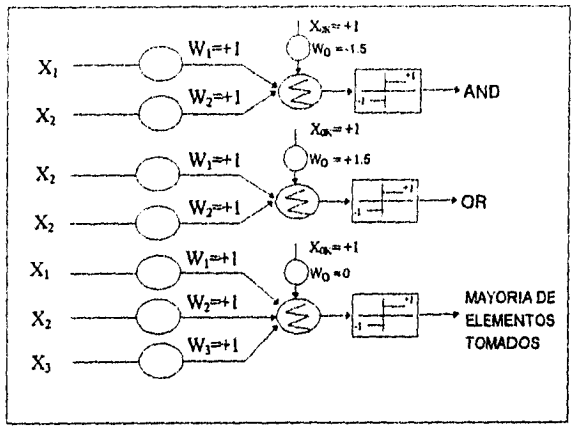


Fig. 3.17. Dispositivos lógicos estables : AND, OR y el de la mayoría de elementos tomados.

En la figura 3.18. se observa una red del tipo MADALINE, la cual esta formada de tres niveles, que a su vez involucran diversos elementos ADALINE, los cuales se encuentran conectados en su totalidad, es decir, cada elemento ADALINE recibe en sus entradas todas las salidas provenientes del nivel procedente.

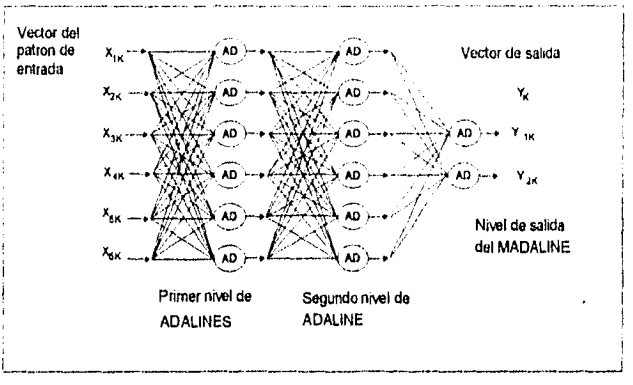


Fig. 3.18. Red Neuronal artificial de tipo MADALINE, con alimentación hacia adelante y formada por 2 niveles ADALINE.

Durante su funcionamiento, la respuesta de cada elemento de salida es comparado con su correspondiente respuesta deseada, en tanto, las señales de error asociadas con cada uno de estos elementos son parcialmente procesadas, de esta manera, la adaptación del nivel de salida es directa. Además, uno de los problemas fundamentales de las redes multiniveles con arquitectura MADALINE, se encuentran en la obtención de las señales de error para los elementos de las redes ADALINE que se encuentran en los niveles ocultos, pero esto no es un factor determinante para decir que una red con alimentación hacia adelante no deba tener una estructura multinivel.

Finalmente, retomando el problema que se presenta frecuentemente en los sistemas de reconocimiento de patrones, tenemos que la invariancia que existe en la salida de la red, ante los cambios presentes tanto en la posición como en el tamaño del patrón de entrada o de la imagen es mínima.

Para esto, existen varias técnicas que han sido utilizadas para solucionar problemas, entre los que se encuentran: el movimiento de traslación, rotación y escala de imágenes, así como la eliminación de los efectos de invariancia en el tiempo. Estas cuatro propiedades de las redes con estructura MADALINE, son esenciales para cualquier sistema que se desee aplicar dentro del campo de la visión artificial como es el reconocimiento de patrones o los censores infrarrojos, esto se debe, a que los humanos no siempre tienen la capacidad de reconocer al momento ciertos objetos, sino hasta que han sido rotados u orientados, o en su defecto han sufrido una reducción o ampliación en su escala.
Además, dentro de los métodos de entrenamiento de redes, el ya mencionado algoritmo LMS tiene una característica muy importante, que dependiendo de ésta, sabemos a que tipo de redes es aplicable, ya que se basa en que solo puede operar sobre las salidas

analógicas provenientes de los combinadores lineales adaptivos, y no siendo así, operable para los valores de salida de tipo bipolar que proporcionan las redes de estructura ADALINE; debido a esto, con una estrategia diferente de entrenamiento, puede ser desarrollada, entrenada y aplicada una red con arquitectura MADALINE.

3.7. ALGORITMO DE BACKPROPAGATION MEJORADO

3.7.1. Antecedentes

La Backpropagation fué formalizada primeramente por Werbos (1972)[27], posteriormente por Parker (1982)[28] y por último por Rumelhart, Hinton y Williams (1985)[29]

El algoritmo Backpropagation, inventado independientemente en tres esfuerzos de investigación separados (Werbos 1974; Parker 1982 y Rumelhart, Hinton y Williams 1986) da un método sistemático para entrenar redes multicapa, sobrepasando así las limitaciones presentadas por Minsky. El algoritmo Backpropagation tiene sus problemas. Primero, no hay garantía de que la red pueda ser entrenada en un período finito de tiempo. Muchos esfuerzos de entrenamiento fallan después de consumir gran cantidad de tiempo de computación. Cuando esto ocurre, el intento de entrenamiento debe ser repetido, sin garantía de que los resultados serán mejores. Tampoco se puede asegurar que la red se entrenará con la mejor configuración posible.

La invención del algoritmo Backpropagation jugó un papel importante en el resurgimiento del interés en las redes neuronales artificiales. Backpropagation[30][31] es un método sistemático para entrenar redes artificiales multicapa. Tiene un fundamento matemático que es sólido pero no es muy práctico. Sin tomar en cuenta sus limitaciones, Backpropagation ha expandido dramáticamente al rango de problemas en los cuales pueden aplicarse las redes neuronales artificiales, y ha generado muchas demostraciones

---

27 P. Werbos. *"Beyord Regression: New Tools for Prediction and Analysis in the Behavioral Sciences"*. PhD Thesis, Harvard, Cambridge, MA, August 1974.

28 D.B. Parker. *"Learning Logic Techinical Report TR-47, Center for Computational Research in Economics and Management Science"*, MIT, Cambridge, MA, April 1985.

29 James McClelland and David Rumelhart. *"Parallel Distributed Processing"*, volumes 1 and 2. MIT Press, Cambridge. Ma, 1986.

30 D. E. Rumelhart, G. E. Hinton and R. J. Williams, *"Learning Internal Representation by Error Propagation"* in D.E. Rumelhart & J. L. McClelland (Eds.) Parallel Distributed Processing: Explorations in the Microstructure of Recognition, Vol. 1: Foundations. MIT Press, 1986.

31 D. E. Rumelhart, and J. L. McClelland, *"Parallel Distributed Processing: Explorations in the Microstructure of Cognition"*, MIT Press, 1986.

exitosa de su poder. Backpropagation tiene una historia interesante. Rumelhart, Hinton, y Williams (1986) presentaron una descripción clara y concisa del algoritmo de Backpropagation. Parker (1982), a su vez se había adelantado a Rumelhart. Un poco antes de esto, Werbos (1972) había ya descrito el método. Rumelhart y Parker podrían haber ahorrado un gran trabajo y esfuerzos si hubieran estado pendientes del trabajo de "Werbos".

Aunque duplicación similar de esfuerzos se ha encontrado en virtualmente toda disciplina científica, en las redes neuronales artificiales el problema es particularmente severo debido a la estructura interdisciplinaria del sujeto a cuestión . La Investigación en Redes Neuronales es publicada en libros y revistas de tantos diversos campos que aún el investigador mas diligente puede no estar al día en investigaciones de éste campo.

3.7.2. El Análisis De Sensibilidad

Una de las principales diferencias que existe entre las redes neuronales y los sistemas expertos es la forma de interiorizar la información que contienen. En sistemas expertos tradicionales, la información es mantenida en forma de reglas.

Cuando un sistema experto llega a una conclusión, uno puede ir a su base de conocimiento para ver cuál de sus reglas fué activada para encontrar la conclusión final. En caso de una red neuronal, éste tipo de explicación no es fácil de obtener porque las redes almacenan la información en forma de pesos o "fuerzas sinápticas". Los pesos en una red neuronal tienen valores numéricos más que simbólicos haciendo difícil el extraer una explicación simbólica.

Sin embargo, es posible aprender más acerca de cómo las entradas afectan la salida de una red neuronal. Esto es posible mediante la aplicación de una técnica proveniente del campo de análisis estadístico no paramétrico llamada "análisis de sensibilidad"[32].

El análisis de sensibilidad[33] inspecciona cada una de las entradas y determina cuando y de que tipo de impacto un pequeño cambio en la entrada puede tener en la salida. Si un pequeño cambio en una entrada en particular ocasiona un cambio drástico en una salida en particular, ésa entrada debe ser considerada como uno de los factores principales en la salida resultante.

---

[32] F. L. Lewis, *"Optimal Estimation"*, John Wiley & Sons, New York 1986.

[33] J. S. Denker, AIP Conference Proceedings 151, *Neural Networks for Computing*, Snowbird Utah, AIP, 1986.

El proceso real de aplicar análisis de sensibilidad consiste en empezar con la primera entrada de la red y añadir una pequeña cantidad, recomputar las salidas, sustraer una pequeña cantidad de la entrada de la red, recomputar la salida, tomar entonces la deferencia entre las dos salidas (el cambio de las salidas) y dividirlo entre las diferencias de las dos entradas (en total). Este resultado en M valores, uno para cada salida. Este proceso es repetido para cada una de las entradas de la red. El resultado es una matriz M x N donde M es el número de salidas, y N el número de entradas. La cantidad más pequeña de cambio en la entrada es llamada "incertidumbre".

Como comentario, ésta técnica básica es usada internamente en una variedad de redes neuronales para determinar cuál elemento de proceso debe ser modificado. En particular, es usado en ciertas variantes de Madeline, y también en ciertas técnicas de optimización de redes.

3.7.3. Configuraciones De Red Para El Algoritmo Backpropagation

La figura 3.19. muestra la neurona usada como el bloque estructural fundamental para las redes Backpropagation. Un conjunto de entradas es aplicado, ya sea desde afuera o desde una capa anterior. Cada uno de éstos conjuntos es multiplicado por un peso, y los productos son sumados. Estas sumas de productos es llamado NET y debe de ser calculado para cada neurona de la red. Después de que NET es calculado, una función de activación F es aplicado para modificarlo, produciendo así la señal de salida OUT.

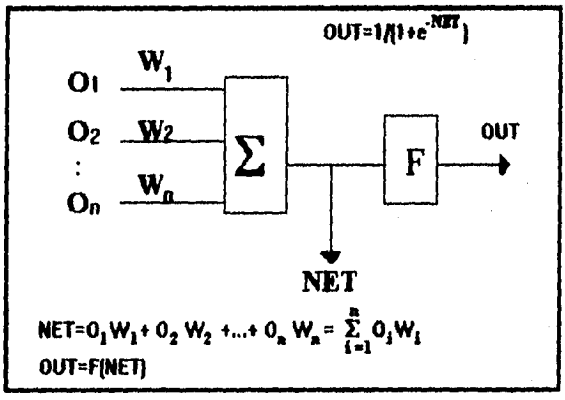


Fig. 3.19. Red Artificial con función de activación.

La figura 3.20. muestra la función de activación usualmente utilizada para Backpropagation.

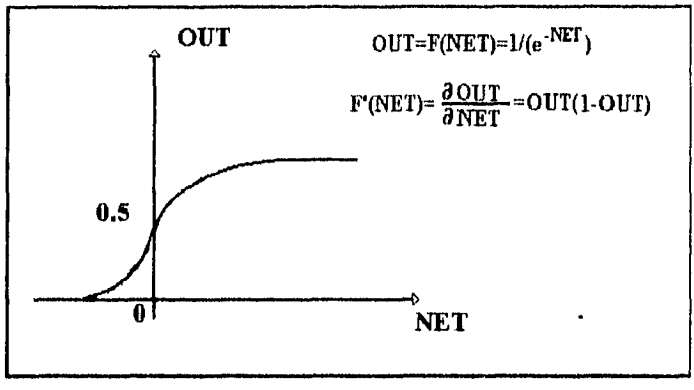


Fig. 3.20. Función sigmoidal de Activación

$$OUT=1/(1+e^{-NET}) \quad \text{..... (9)}$$

Como se muestra en la ecuación 10, ésta función, llamada una sigmoide, es deseable que posea derivada de primer grado, un hecho que utilizamos en implementar el algoritmo de Backpropagation:

$$\frac{\partial OUT}{\partial NET} = OUT(1-OUT) \quad \text{..... (10)}$$

Algunas veces llamada función logística o simplemente función compresora, el sigmoide comprime el rango de NET de tal forma que OUT cae entre 0 y 1.
Las redes multicapa tienen mayor poder representacional que las redes unicapa sólo cuando es introducida una no linealidad. La función compresora produce la no linealidad necesitada.

Existen muchas funciones que pueden ser usadas; el algoritmo de Backpropagation sólo requiere que la función sea completamente diferenciable. El sigmoide satisface éste requerimiento. Tiene la ventaja adicional de que provee una forma de control automático de ganancia. Para señales pequeñas, (NET cercano a cero) la tangente de la curva entrada/salida es muy inclinada, produciendo una gran ganancia. Si la magnitud de la señal se hace mas grande, la ganancia decrece. En ésta forma las señales grandes pueden

ser acomodadas por la red sin saturación, mientras que las señales pequeñas se les permite pasar a través sin atenuación excesiva.

3.7.4. La Red Multicapa

La figura 3.21. muestra una red multicapa hecha para entrenar con Backpropagation. El primer conjunto de neuronas (conectadas a las entradas) sirven sólo como puntos de distribución; no desempeñan sumatoria de entrada. La señal de entrada es simplemente pasada a través hacia los pesos en sus salidas. Cada neurona en capas subsecuentes producen las señales NET y OUT descritas arriba.

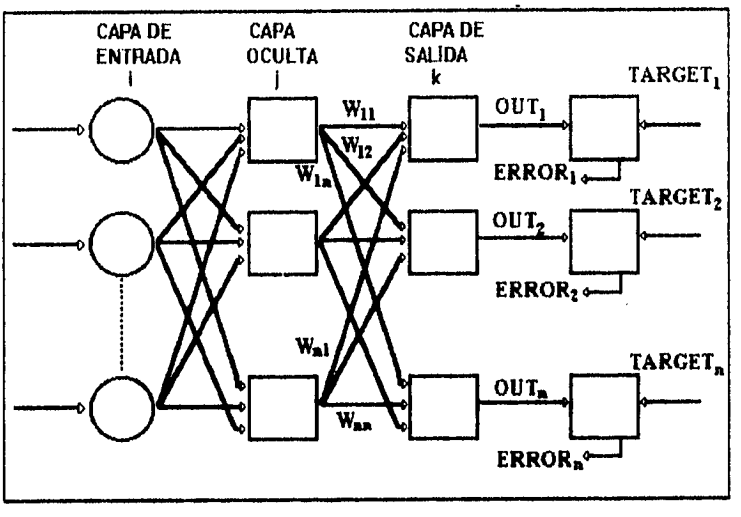


Fig. 3.21. Red Backpropagation de dos capas.

Backpropagation puede ser aplicada a redes con cualquier número de capas; sin embargo, sólo dos capas de pesos son necesarias para demostrar el algoritmo. En éste punto de discusión, sólo redes de alimentación en adelanto son consideradas. También es posible aplicar Backpropagation a redes con retroalimentación.

El objetivo de entrenar la red es ajustar los pesos de tal forma que la aplicación de un conjunto de entradas produce el conjunto de salidas deseado. Por razones de brevedad, éstos conjuntos de entrada-salida pueden ser referidos como vectores. El entrenamiento asume que cada vector de entrada es aparejado con un vector objetivo representando la

salida deseada; juntos éstos vectores son llamados un par de entrenamiento. Usualmente, una red es entrenada sobre un número de pares de entrenamiento. Por ejemplo, la parte de entrada de un par de entrenamiento puede consistir de un patrón de unos y ceros representando una imagen binaria de una letra o del alfabeto.

Si una línea se pasa del cuadrado, la correspondiente entrada de la neurona es uno; de otra manera, ésa entrada de la neurona es cero. La salida pudiera ser un número que represente a la letra A, o tal vez otro conjunto de unos y ceros que pudieran ser usados para producir las letras del alfabeto, se requerirían 26 pares de entrenamiento. El grupo de pares de entrenamiento es llamado grupo de entrenamiento. Antes de empezar el entrenamiento todos los pesos deben ser inicializados a pequeños números aleatorios.

Esto asegura que la red no está saturada por grandes valores de los pesos, y previene otras patologías del entrenamiento. Por ejemplo, si todos los pesos empiezan con valores iguales y el desempeño deseado requiere valores diferentes, la red no aprenderá. Entrenar la red de Backpropagation requiere los pasos siguientes:

1. Seleccionar el próximo par de entrenamiento del conjunto de entrenamiento; aplicar el vector de entrada a la entrada de la red.
2. Calcula la salida de la red.
3. Calcular el error entre la salida de la red y la salida deseada el vector objetivo del par de entrenamiento.
4. Ajustar los pesos de la red en una forma que minimice el error.
5. Repetir los pasos del 1 al 4 para cada vector en el conjunto de entrenamiento hasta que el error para el conjunto entero sea aceptablemente bajo.

Las operaciones requeridas en los pasos 1 y 2 son similares a la manera en la cual la red entrenada será usada; esto es, un vector de entrada es usado y la salida es calculada. Los cálculos son desempeñados capa por capa.

Refiriéndonos a la figura 3.21., son calculadas primero las salidas de las neuronas en la capa j ; estas son entonces usadas como entradas a la capa k ;las neuronas de la salida de la capa k son calculadas y eso constituye el vector de salida de la red.

En el paso 3, cada una de las salidas de la red etiquetada como OUT en la figura 3.21., es substraída de su correspondiente componente del vector objetivo para producir un error. Este error es usado en el paso 4 para ajustar los pesos de la red, donde la polaridad y magnitud de los cambios de los pesos es determinada por el algoritmo de entrenamiento. Después de las suficientes repeticiones de éstos cuatro pasos, el error entre las salidas reales y las salidas objetivo debe ser reducido a un valor aceptable, entonces se dice que la red está entrenada. En éste punto la red es utilizada para recognición y los pesos no son cambiados.

Puede verse que los pasos 1 y 2 constituyen un "paso hacia adelante" en el que la señal se propaga desde la entrada de la red hacia su salida. Los pasos 3 y 4 son un "paso hacia tras", aquí, la señal de error calculada se propaga hacia atrás (Backward) a través de la red donde es usada para ajustar los pesos. Estos dos pasos son ahora expandidos y expresados en una forma un poco más matemática.

Los pasos de 1 y 2 pueden ser expresados en forma de vector como sigue : un vector de entrada X es aplicado y un vector de salida Y es producido. El par vector objetivo X y Y vienen del conjunto de entrenamiento. El cálculo es efectuado en X para producir el vector de salida Y. El cálculo de redes multicapa es hecho capa por capa, empezando en la capa más cercana a las entradas. El valor NET de cada neurona en la primera capa es calculado como la suma pesada de las entradas de sus neuronas. La función de activación F entonces "comprime" NET para producir el valor OUT para cada neurona en la capa. Una vez que el conjunto de salida de una capa es encontrado, sirve como entrada para la próxima capa. El proceso es repetido, capa por capa, hastã que el conjunto final de salidas de la red es producido.

Este proceso puede ser establecido en notación vectorial. Los pesos entre neuronas pueden ser considerados como la matriz W. Por ejemplo; el peso de la neurona 8 en la capa 2 a la neurona 5 en la capa 3 es designado W 8,5. Mejor que utilizar la suma de productos, el vector NET para una capa N pude ser expresado como el producto de X y W. En notación vectorial N = XW. Aplicando la función F al vector NET de N, componente por componente, produce el vector de salida 0. Así, para una capa dada, la siguiente expresión describe el proceso de cálculo.

$$O = F(XW) \quad \text{..... (11)}$$

El vector de salida para una capa es el vector de entrada de la próxima, entonces el cálculo de las salidas de la capa final requiere la aplicación de la ecuación 11 para cada capa, desde la entrada de la red hasta su salida .

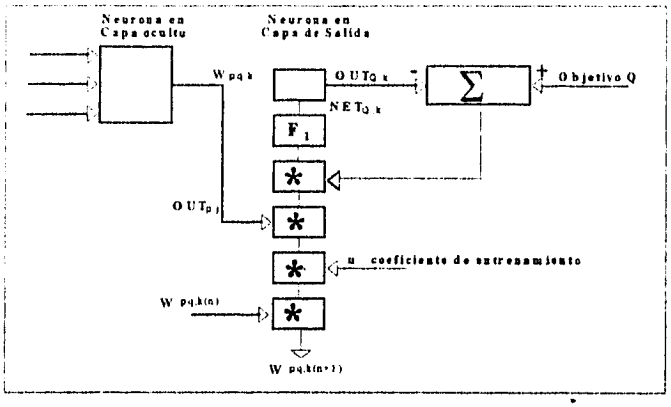


Fig. 3.22. Entrenamiento de un peso en la capa de salida.

3.7.5. Ajustar Los Pesos De La Capa De Salida

Ya que un valor objetivo es disponible para cada neurona en la capa de salida, ajustar los pesos asociados es efectuado fácilmente, utilizando una modificación de la regla delta.

Las capas interiores son referidas como "capas escondidas", ya que sus salidas no tienen valores objetivo para comparación; de aquí, el entrenamiento es más complicado. La figura 3.22. muestra el proceso de entrenamiento, para un solo peso desde la neurona *p* en la capa escondida *j* a la neurona q en la capa de salida *k*, para producir una señal de ERROR. Esto es multiplicado por la derivada de la función compresora OUT (1-OUT) calculada para la neurona *k* de ésa capa, produciendo entonces el valor δ.

$$\delta = OUT\,(1\text{-}OUT)\,(TARGET\text{-}OUT) \quad \ldots\ldots (12)$$

Entonces δ es multiplicado por OUT desde una neurona j, la neurona fuente para el peso en cuestión. Este producto es a su vez multiplicado por un coeficiente de entrenamiento n (típicamente de 0.01 a 1.0) y el resultado es sumado al peso. Un proceso idéntico es efectuado para cada peso procedente de una neurona en la capa oculta hacia una neurona en la capa de salida.

$$\Delta w_{pq,k} = n\delta_{q,k}\, OUT_{p,j} \quad \ldots\ldots (13)$$

$$w_{pq,k}^{(n+1)} = w_{pq,k}^{(n)} + \Delta w_{pq,k} \quad \ldots\ldots (14)$$

Donde:

$w_{pq,k}^{(n)}$ es el valor de un peso desde la neurona p en la capa escondida a la neurona q en la capa de salida en el paso n (antes del ajuste); note que la k indica que el peso es asociado con su capa destino.

$w_{pq,k}^{(n+1)}$ es el valor del peso en el paso $n+1$ (después del ajuste).

$\delta_{q,k}$ es el valor de δ para la neurona que en la capa escondida j.

p y q se refieren a neuronas específicas, donde j y k se refieren a capas.

3.7.6. Ajusta Los Pesos De Las Capas Ocultas.

Las capas ocultas no tienen vector objetivo, entonces el proceso de entrenamiento descrito arriba no pude ser usado. Esta falla de un objetivo de entrenamiento impidió los esfuerzos de entrenar redes multicapa hasta que Backpropagation proveyó de un algoritmo utilizable.

Backpropagation entrena las capas escondidas mediante propagar el error de salida hacia atrás a través del de la red capa por capa, ajustando los pesos en cada capa. La ecuación 13 y 14 son utilizadas para todas las capas, tanto de salida como escondidas; sin embargo, para capas escondidas δ debe ser generada sin el beneficio de un vector objetivo.

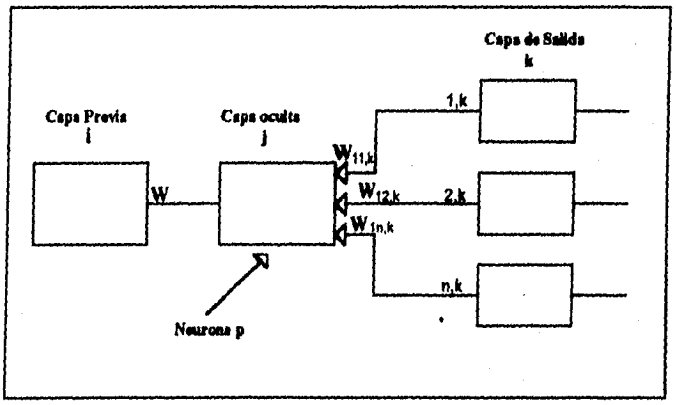


Fig. 3.23. Ajustando los pesos de las capas ocultas.

Primero, δ es calculado para cada neurona en la capa de salida, como en la ecuación 12. Esto es usado para ajustar los pesos que alimentan la capa de salida, entonces se propaga hacia atrás a través de los mismos pesos para generar un valor de d para cada neurona en la primera capa de salida. Estos valores de δ son utilizados a su vez para ajustar los pesos de ésta capa escondida y, en forma similar, es propagada hacia atrás a todas las capas precedentes.

Considere una neurona en la capa oculta justo antes de la capa de salida. En el paso en adelanto, ésta neurona propaga su valor de salida hacia las neuronas en la capa de salida a través de los pesos de interconexión. Durante el entrenamiento, éstos pesos operan en reversa, pasando el valor de d desde la capa de salida hacia la capa oculta. Cada uno de éstos pesos es multiplicado por el valor de d de la neurona a la cuál se conecta en la capa de salida. El valor de d necesario para la neurona de la capa oculta es producido por sumar todos éstos productos y multiplicarlos por la derivada de la función compresora:

$$\delta_{p,j} = OUT_{pj}\,(1\text{-}OUT)\,(\Sigma\delta_{q,k}\,w_{pq,k}) \qquad (15)$$

ya que con δ en la mano, los pesos que alimentan la primera capa oculta pueden ser ajustados utilizando las ecuaciones 13 y 14, modificando los índices para indicar las capas correctas. Para cada neurona en una capa oculta dada, los valores de δ deben ser calculados, y todos los pesos asociados con ésa capa deben ser ajustados. Esto es repetido, regresando a hacia la capa de entrada, capa por capa, hasta que todos los pesos son ajustados.

Con notación vectorial, la operación de propagar el error hacia atrás puede ser expresada en forma mucho más compacto. Sea D_k el conjunto de δs y el conjunto de pesos para la capa de salida a arreglar W_k. Para llegar a D_k (δ de la capa oculta), el vector δ, es decir D_k, se sigue éstos dos pasos:

1. Multiplicar el vector δ de la capa de salida Dk por la transpuesta de la matriz de pesos que conecta la capa oculta a la capa de salida (W_k^t).

2. Multiplicar cada componente del producto resultante por la derivada de la función compresora para la neurona correspondiente en la capa oculta.

Simbólicamente:

$$D_j = D_k W_k^t \$ [O_j \$ (I - O_j)] \qquad (16)$$

donde $ indica la multiplicación componente a componente de dos vectores. O_j es el vector de salida para la capa oculta j, es I es un vector cuyos todos componentes son 1.

3.7.7. Añadiendo Una Neurona De Dirección.

En muchos casos, es deseable proveer cada neurona con una dirección de entrenamiento. Esto desfasa el origen de la función logística, produciendo un efecto que es similar al ajustar el umbral de la neurona del perceptrón, permitiendo así una convergencia más rápida del proceso de entrenamiento. Esta característica es fácilmente incorporada dentro del algoritmo de entrenamiento; un peso conectado a +1 es añadido a cada neurona. Este peso es entrenable en la misma manera que lo son los otros pesos, excepto que la fuente es siempre +1 en vez de ser la salida de una neurona de una capa previa.

3.7.8. Momentun

Rumelhart, Hinton y Williams (1986)[34] describen un método para mejorar el tiempo de entrenamiento del algoritmo de Backpropagation, mientras se mejora la estabilidad del proceso.
Llamado Momentun, el método consta añadir un término al ajuste del peso que es proporcional a la cantidad del cambio de peso previo. Una vez que un ajuste es efectuado, es "recordado" y sirve para modificar todos los ajustes de peso subsecuentes. Las ecuaciones de ajuste son modificadas como sigue:

$$\Delta w_{pq,k}^{(n+1)} = \eta(\delta_{p,k} OUT_{pj}) + \alpha[\Delta w_{pq,k}^{(n)}] \quad \ldots\ldots(17)$$

donde α, el coeficiente de momentun, es comúnmente puesto a 0.9.

Utilizando el método de momentun, la red tiende a seguir el fin de canales angostos en la superficie de error (si existen) mas que cruzar rápidamente de lado a lado. Este método parece trabajar bien en algunos problemas, pero tiene poco efecto o efecto negativo en otros.

Sejnowski y Rosenberg (1987)[35] describen un método similar basado en un suavizado exponencial que puede ser superior en algunas aplicaciones.

$$\Delta w_{pq,k}^{(n+1)} = \alpha \Delta w_{pq,k}^{(n)} + (1-\alpha)\delta_{p,k} OUT_{pj} \quad \ldots\ldots(18)$$

---

[34] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *"Learning Internal Representation by Error Propagation"* in D.E. Rumelhart & J. L. McClelland (Eds.) Parallel Distributed Processing: Explorations in the Microstructure of Recognition, Vol. 1: Foundations. MIT Press, 1986.
[35] T. Sejnowski and C. R. Rosenberg, *"NETtalk: a Parallel Network That Learns to Read Aloud"*, John Hopkins Univ. Technical Report JHU/EECS-8601, 1986.

entonces el cambio de pesos es computado:

$$W_{pq,k}^{(n+1)}=W_{pq,k}^{(n)}+n\Delta W_{pq,k}^{(n+1)} \quad \ldots\ldots(19)$$

donde α es un coeficiente en el rango de 0.0 a 1.0. Si α es 0.0, entonces el suavizador es mínimo; el ajuste entero de peso viene del nuevo cambio calculado. Si α es 1.0, el nuevo ajuste es ignorado y el cálculo previo es repetido entre 0 y 1 hay una región donde el ajuste del peso es frenado por un coeficiente proporcional a α. Otra vez n es el coeficiente de entrenamiento, sirviendo para ajustar el tamaño del promedio de cambio de peso.

3.7.9. El Simulador De Backpropagation

En esta sección, se describen en detalle los algoritmos necesarios para instrumentar un simulador de Backpropagation.

En una Backpropagation, el flujo de señales es bidireccional, pero solamente en una dirección a la vez. Durante el entrenamiento, hay dos tipos de señales presente en la red durante el primer medio ciclo, modula el flujo de las señales de salida de entrada a salida; durante el segundo medio ciclo, el flujo de señales de error de la capa de salida a la capa de entrada. Dentro del modo productivo, únicamente el feedforward (alimentación hacia adelante), modula las señales de salida a utilizar.

Diversas suposiciones pueden incorporarse dentro de la designación de la simulación. Primero, la salida funciona sobre todas las unidades de las capas ocultas -y salidas- es supuesta como la función sigmoidal. Esta imposición es también implícita en el pseudocódigo para calcular el termino de error para cada unidad. Además, nosotros podemos incluir el termino momentum en los pesos calculados hasta el momento. Estas suposiciones implican la necesidad de almacenar los pesos actuales en una iteración, para usarles en la próxima iteración. Finalmente, valores parciales pueden no estar incluidos en los cálculos.

En este modelo de red, la unidad de entrada es únicamente el fan-out de un proceso. Eso es, la unidad en la capa de entrada no ejecuta conversión de datos en el patrón de entrada de red. Ellos simplemente actúan como retención de los componentes para el vector de entrada dentro de la estructura de la red. Así, el proceso de entrenamiento comienza con la aplicación externa de patrones de entrada a la capa de entrada de unidades. Adelante las señales se propagan hasta encontrarse acordes para la siguiente secuencia de actividades:

1. Localice el primer proceso de unidad en la capa superior inmediata al flujo de la corriente de la capa.
2. Poner la corriente de entrada total a cero.
3. Computar el producto de la primera conexión de pesos de entrada y la salida para la transmisión de unidad.
4. Suma el producto en el valor acumulado total.
5. Repite los pasos 3 y 4 por cada entrada conectada.
6. Computa el valor de salida para esas unidades aplicando en la salida la siguiente función.

$$f(x) = \frac{1}{1+e^{-x}} \qquad \text{donde x = entrada total.}$$

7. Repite los pasos 2 al 6 por cada unidad de la capa.
8. Repite los pasos 1 al 7 por cada capa de la red.

Una vez calculado el valor de salida para cada una de las unidades de la red, el valor computado para la unidad en la capa de salida es comparada con la salida del patrón deseado, elemento por elemento. En cada unidad de salida, un valor de error es calculado. Este termino de error es el retroalimentado hacia todas las unidades en la estructura de la red a través de la siguiente secuencia de pasos:

1. Localice el primer proceso de unidad en la capa inmediata posterior de la capa de salida.
2. Poner la corriente de error total a cero.
3. Computar el producto de el primer peso de salida conectada y el error proporcionado por la unidad en la capa superior.
4. Sumar el producto al error acumulado.
5. Repetir los pasos 3 y 4 por cada conexión de salida.
6. Multiplica el error acumulado por $o(1\text{-}o)$, donde o es el valor de salida para la unidad de capa oculta producida durante la operación feedforward.
7. Repite los pasos 2 al 6 por cada unidad de la capa.
8. Repite los pasos 1 al 7 por cada capa.
9. Localice el primer proceso de unidad en la capa superior de la capa.
10. Compute el peso cambiando el valor de la primera conexión de entrada de la unidad por la suma del error acumulado en la unidad del valor de entrada por unidad.
11. Modifique el cambio de peso de los términos por la suma de el termino momentum igual a la fracción de los cambios de pesos para las iteraciones previas.
12. Salve los nuevos valores de los pesos y los viejos valores cámbielos por las conexiones.
13. Cambie las conexiones de peso por la suma de los nuevos pesos y cambie el valor de las viejas conexiones de pesos.
14. Repite los pasos 10 al 13 por cada conexión de entrada de la unidad.
15. Repite los pasos 10 al 14 por cada unidad de cada capa.

16. Repite los pasos 10 al 15 por cada capa de la red.

El procedimiento básico de entrenamiento de la red es incorporada en la siguiente descripción:

1. Aplicar un vector de entrada a la red y calcular su correspondiente valor de salida.
2. Comparar las salidas actuales con las salidas correctas y determinar una medida de el error.
3. Comparar en cual dirección (+ o -) cambia cada peso en orden a reducir el error.
4. Determinar la cantidad por la cual cambiar cada peso.
5. Aplicar las correcciones a los pesos.
6. Repetir los pasos 1 al 5 con todos los vectores de entrenamiento hasta el error por todos los vectores en el conjunto de entrenamiento es reducido a un valor aceptable.

Ahora definiremos estos mismos pasos con sus correspondientes ecuaciones para un solo parte de entrenamiento:

1.- Aplicar el vector de entrada Xp=(Xp1,Xp2,...,Xpi)t a las unidades de entrada.
2.- Calcular el valor de entrada a la unidad de capa oculta:

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

3.- Calcular las salidas desde la capa oculta

$$i_{pj} = f_j^h(net_{pj}^h)$$

4.- Mover a la capa de salida. Calcular el valor de entrada a la red para la cada unidad.

$$net_{pj}^{o} = \sum_{j=1}^{L} w_{ji}^{o} i_{pj}$$

5.- Calcular las salidas

$$O_{pk} = f_k^o(net_{pj}^o)$$

6.- Calcular los términos de error para las unidades de salida:

$$\delta_{pk}^{o} = (y_{pk} - O_{pk}) f_k^{o}(net_{pk}^{o})$$

7.- Calcular los términos de error para las unidades ocultas:

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

Nótese que los términos de error en las unidades ocultas son calculados antes que se hagan las correcciones de pesos a las unidades de salida ocultas.

8.- Actualización de los pesos sobre la capa de salida
$w^o_{kj}(t+1) = w^o_{kj}(t) + \eta \delta^o_{pk} i_{pj}$

9.- Actualizando pesos sobre las capas ocultas:
$w^h_{ji}(t+1) = w^h_{ji}(t) + \eta \delta^h_{pj} x_i$

El orden de los pesos actualizados sobre una capa individual no es importante.
Pero será seguro calcular el termino de error.

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta^2_{pk}$$

3.7.10. Estructuras de Datos de BPN

Comenzamos esta sección del simulador de Backpropagation con una presentación de las estructuras de datos de red que requeriremos. Aunque la Backpropagation es similar en estructura a la red de Madeline, es también diferente en el uso de varios parámetros adicionales que tienen que estar almacenados en una conexión de base de unidad de red. Basado en nuestro conocimiento de como opera la backpropagation, ahora propondremos un registro de información que definirá la estructura de nivel superior del simulador.

```
Record BNP=
   INUNITS: ^layer;          {localiza capa de entrada}
   OUTUNITS: ^layer;         {localiza unidades de salida}
   LAYER: layer[];           {tamaño de la red}
   alpha,                    {el termino momentum}
   eta: float;               {la tasa de aprendizaje}
end record;
```

La figura 3.2 ilustra la relación entre el registro de red y toda la estructura subordinada.

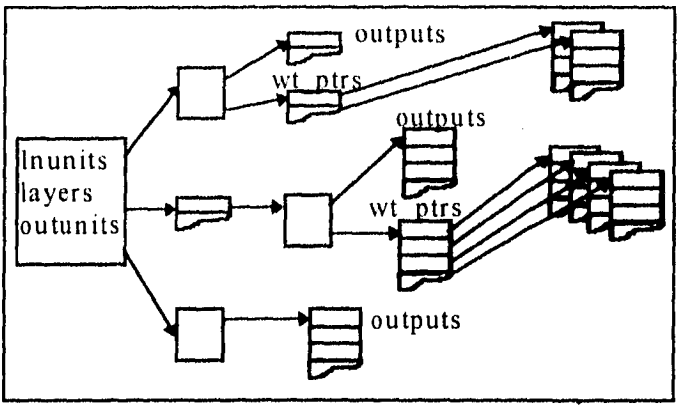


Fig.3.2. La estructura de datos de Backpropagation está mostrada sin los arreglos para el error y el término delta para claridad. La red está definida por unos apuntadores de contenido de registros a la estructura subordinada, así como parámetros de red específicos. En este diagrama, solamente tres capas son ilustradas, muchas más capas ocultas pueden ester añadidas por extensiones simples de la capa posterior del arreglo.

La inspección de la Backpropagation registra que esta estructura nos permite diseñar o crear redes que contienen algo más que simplemente tres capas de unidades. En la práctica, Backpropagation requiere más de tres capas para resolver un problema.

Es obvió que el registro de backpropagation contiene la información de mayor interés global de las unidades en la red específicamente, la alfa (α) y los términos de eta (η). Ahora se definirá la estructura de capa que utilizaremos para construir el resto de la red, desde la base para localizar toda la información utilizada por las unidades en cada capa. Para definir la estructura de capa, tenemos que recordar que la Backpropagation tiene dos diferentes tipos de operación, y que diferente información es necesitada en cada fase. Así, la estructura de capa contiene apuntadores a dos diferentes conjuntos de arreglos: uno utilizado durante la propagación, y uno utilizado durante el error de propagación. En base a estas consideraciones, ahora definiremos la estructura de capa para la backpropagation.

```
Record layer =
   salidas: ^float[];          {localiza arreglo de salida}
   weights: ^float[];          {localiza arreglo de conexión}
   errors: ^float[];           {localiza términos de error por capa}
   last_delta: ^float[];       {localiza termino previo de delta}
end record;
```

Durante la fase de propagación hacia adelante, la red utilizará la información contenida en las salidas y los pesos (arreglos), de la misma manera que el simulador de Adeline. Sin embargo, durante la fase de backpropagation, el simulador requiere acceso para un arreglo de términos de error (uno por cada unidad de la capa) y para la lista de parámetros de cambio utilizados durante el aprendizaje previo pasa (almacenados en una base de conexión). Combinando el acceso del mecanismo a todos estos términos en la estructura de capa, podemos continuar el proceso de manera eficiente al menos durante la fase de adelanto. Lamentablemente, la actividad durante la backpropagation puede ser ineficiente, debido a que estaremos accediendo diferentes arreglos más bien que accediendo ubicaciones secuenciales con los arreglos.

Los siguientes cuatro algoritmos instrumentan el feedforward de señales de propagación en nuestra red. El primer procedimiento servirá como la rutina de interface entre la computadora y la simulación de la backpropagation. Suponemos que el usuario haya definido un arreglo de números de punto flotante que indican el patrón para estar aplicado a la red como entradas.

```
procedure set_inputs (INPUTS, NET_IN: ^float[])
{ copian los valores de entrada en la capa de entrada total}

var
   temp1: ^float[];                    {apuntador local}
   temp2: ^float[];                    {apuntador local}
i: integer;                            {contador}

begin
   temp1 = NET_IN;                     {localiza la capa de entrada}
   temp2 = INPUTS;                     {localiza los valores de entrada}
   for i = 1 to length(NET_IN) do      {para todo valor de entrada}
   temp1[i] = temp2[i];                {copia entrada en entrada total}
   end do;
end;
```

La rutina desempeña la propagación de la señal hacia adelante entre las dos capas, localizadas por el apuntador. Esta rutina encarna los cálculos hechos las ecuaciones 1 y 2 para la capa oculta y las ecuaciones 3 4 para la capa de salida.

```
Procedure propagate_layer (LOWER, UPPER: ^layer)
{ propaga las señales de la capa inferior a la superior}

var
   inputs: ^float[];                    {tamaño de la capa de entrada}
   current: ^float[];                   {tamaño de la capa de corriente}
   connects: ^float[];                  {paso a través de entradas}
   sum: real;                           {producto acumulado}
   i,j: integer;                        {contadores}

begin
   inputs = LOWER^.salidas;             {localiza capa inferior}
   current = UPPE^.salidas;             {localiza capa superior}
   for i = 1 to length(current) do      {paso a través de las unidades de la capa}
      sum = 0;                          {acumulador igual a cerò}
      connets = UPPER^.weigth^[i];      {encuentra el comienzo del arregl wt. }
      for j = 1 to length(inputs) do    {paso a través de las entradas de la
      unidad}
      sum = sum + inputs[j]*connects[j]; {acumula productos}
      end do;
      current[i] = 1.0 / (1.0 + exp(-sum)); {genera salidas}
   end do;
end;
```

El próximo procedimiento desempeña la propagación de la señal por el interior de la red. Esta asume que el contenido de la capa de entrenamiento de patrones, será puesta allí en un nivel oculto llamado a set_inputs.

```
procedure propagate_forward(NET:BNP)    {desempeño de la propagación de la señal
                                         por la red}
var
upper:^layer;                           {apuntador a la capa superior}
lower:^layer;                           {apuntador a la capa inferior}
i:integer;                              {contador de capas}

begin
for i=1 to length(NET.layers)do         {para todas las capas}
    lower=NET.layers[i];                {apuntador igual a la capa de entrada}
    upper=NET.layers[i+1];              {apuntador igual a la próxima capa}
    propagate_layer(lower,upper);       {propagación hacia adelante}
end do;
end;
```

El final de la rutina necesaria para la propagación hacia adelante desea extraer los valores de salida generados por la red y los copia dentro de un arreglo externo especifico que es llamado en el programa. Las rutinas son el complemento de la rutina set_input descrita anteriormente.

```
procedure get_salidas(NET_OUTS,SALIDAS:^float[])    {Copia los valores de salida
                                                      de la red dentro de una
                                                      salida especifica}

var
    temp1:^float[];                                  {apuntador local}
    temp2:^float[];                                  {apuntador local}

bergin
    temp1=NET_OUTS;                                  {localiza la próxima capa de
                                                      salida}
    temp2=SALIDAS;                                   {localiza los valores de
                                                      salida del arreglo}
    for i=1 to length(NET_OUTS) do                   {para todas las salida}
        temp2[i]=temp1[i];                           {copia las salidas de la red}
        end do;                                      {hasta el arreglo de salida}
    do;
```

3.7.11. Mejoras Al Algoritmo Backpropagation

Parker (1987) describe un método para mejorar la velocidad de convergencia del algoritmo de backpropagation. Llamado backpropagation de segundo orden, utiliza segundas derivadas para producir una estimación más exacta del cambio de peso correcto.

Parker ha demostrado que el algoritmo es óptimo en el sentido de que utilizando derivadas mas altas que de segundo orden no mejorarán el resultado estimado.

Los requerimientos computacionales se incrementan comparados con la backpropagation de primer orden, y mas resultados de pruebas son necesarios para demostrar que el costo adicional es justificado.

Stornetta y Huberman (1987) describen un método decepcionantemente simple para mejorar las características de las redes Backpropagation. Ellos apuntan que el rango dinámico de entradas y salidas convencional de 0 a 1 no es óptimo. Ya que la magnitud

de un ajuste de peso $\Delta Wpq,k$ es proporcional al nivel de salida de la neurona que origina OUT_{pj}, un nivel de 0 resulta en no modificar el peso.

Con vectores de entrada binarios, la mitad de las entradas, en promedio, serán 0, y los pesos conectados a ella no serán entrenados. La solución reside en cambiar el rango de entradas a 0.5 y añadir una direccional a la función compresora para modificar el rango de salida de la neurona a 0.5. La nueva función compresora es como sigue:

$$OUT=-0.5+1/(e^{-NET}+1) \quad \ldots\ldots(20)$$

Los tiempos de convergencia fueron reducidos en un promedio de 30 a 50% con éstos cambios fácilmente implementados, esto es un ejemplo de las modificaciones prácticas que pueden traer mejoras sustanciales en el backpropagation.

Pineda (1988) y Almeida (1987) han descrito métodos para aplicar bäckpropagation a redes recurrentes, esto es, redes cuyas salidas retroalimentan sus entradas. Ellos han demostrado que el aprendizaje puede ocurrir muy rápidamente en tales sistemas y el criterio de estabilidad es fácilmente satisfecho.

3.7.12. Algunas Aplicaciones

Backpropagation ha sido aplicado a una variedad de aplicaciones de investigación; unas pocas son descritas para demostrar el poder de éste método.

NEC en Japón ha anunciado recientemente que ha aplicado Backpropagation a un nuevo sistema de reconocimiento óptico de caracteres, mejorando así la eficiencia en un 99%. Esta mejora fué lograda a través de una combinación de algoritmos convencionales con una red backpropagation proveyendo verificación adicional.

Sejnowski y Rosemberg (1987)[36] produjeron un éxito sensacional con NetTalk, un sistema que convierte inglés impreso en habla altamente inteligente. Su grabación del proceso de entrenamiento tiene un fuerte parecido con los sonidos de un niño de varias etapas del proceso de aprender a hablar.

Burr (1987) ha utilizado Backpropagation en reconocimiento por máquina del inglés manuscrito. Los caracteres son normalizados por tamaño, son colocados en una cuadrícula, y se hacen proyecciones de las líneas a través de los cuadrados de la

[36] T. Sejnowski and C. R. Rosenberg, *"NETtalk: a Parallel Network That Learns to Read Aloud"*, John Hopkins Univ. Technical Report JHU/EECS-8601, 1986.

cuadrícula. Estas proyecciones forman entonces las entradas a una red Backpropagation. El reporta exactitudes del 99.7% cuando se utiliza con un diccionario filtro.

Cotrell, Munro y Zipser (1987) reportan una exitosa aplicación de comprensión de imágenes en el cual las mismas fueron representadas con un bit por pixel, una mejora de ocho veces sobre los datos de entrada.

El algoritmo de Backpropagation ha sido probado con un número de problemas determinísticos tales como el problema de la OR-exclusiva, en problemas relacionados a síntesis de voz y reconocimiento y en problemas relacionados a reconocimiento de patrones visuales.

Se ha visto que funciona bien en muchos casos y para encontrar buenas soluciones a los problemas propuestos. Una demostración del poder de éste algoritmo fué hecha por Sejnowski. Entrenó una red de dos capas tipo perceptrón, con 120 unidades escondidas y mas de 20,000 pesos para formar letras para reglas de transcripción fonética. La entrada de ésta red fué un código binario indicando ésas letras en una ventana deslizante de siete letras de largo que fué movida sobre una transcripción de texto hablando. La salida deseada era un código binario indicando la transcripción fonética de la letra en el centro de la ventana.

Después de 50 veces a través de un diálogo conteniendo 1024 palabras, el porcentaje de error de transcripción fué sólo 5%. Este porcentaje se incrementó a 22% para una continuación de ése diálogo que no fué usado para el entrenamiento.

3. 8. PROGRAMAS DE APRENDIZAJE SUPERVISADO Y NO SUPERVISADO.

A) Programa generalizado de Regla Delta (Aprendizaje Supervisado)

Este segmento contiene una programa en lenguaje C para usarse como punto de partida para la exploracion del procedimiento de la Regla Delta Generalizada para el aprendizaje supervisado. El programa provee codigo y apoyo a las siguientes tareas:

1. Especificacion de arquitectura de red.
2. Aprendizaje de los Pesos y entradas con uso de entrenamiento de conjunto de patrones.
3. Uso de la red para obtencion de valores de salida para nuevos patrones, ambos para propositos de clasificacion o para estimacion de valores de atributos asociados.

El programa se ejecuta bajo PC-DOS/MS-DOS y puede ser compilado con Microsoft C.

/** DESCRIPCION DEL PROGRAMA:

ESTE PROGRAMA PERMITE AL USUARIO CONSTRUIR UNA RED CON REGLA DELTA GENERALIZADA UTILIZANDO APRENDIZAJE SUPERVISADO.
El USUARIO PUEDE ESPECIFICAR LA CANTIDAD DE ENTRADA Y UNIDADES DE SALIDA, NUMERO DE CAPAS Y NUMERO DE UNIDADES OCULTAS EN CADA CAPA.

DESPUES DE QUE LA RED ESTA CONSTRUIDA,TIENE LUGAR EL APRENDIZAJE DE LA RED. CON UN CONJUNTO DADO DE MUESTRAS ENTRENADORAS, UN USUARIO ESPECIFICA VALORES DE LA ETA (TASA DE APRENDIZAJE), LA ALFA (TASA DEL MOMENTUM), ERRORES DE TOLERANCIA MAXIMOS Y NUMERO MAXIMO DE ITERACIONES.

DESPUES DEL APRENDIZAJE, TODA LA INFORMACION RELEVANTE A LA ESTRUCTURA DE RED, INCLUYENDO PESOS Y UMBRALES SON ALMACENADOS EN UN ARCHIVOS.

LAS SALIDAS PUEDEN ESTAR GENERADAS POR NUEVOS PATRONES LEIDOS DE ALGUNARCHIVO Y RECONSTRUYENDO LA RED.

ENTRENANDO MUESTRAS DE CONJUNTO Y MUESTRAS ADICIONALES PARA PROCESAMIENTO ESTAN SON ALMACENADO EN UN ARCHIVOS. */

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#ifndef VAX
/* para declaración de malloc() en PC o compatible
#include <malloc.h>
#endif
```

```
/* definición de constantes utilizadas a través de funciones* /

#define NMXUNIT  50                    /* max no. de unidades en una capa */
#define NMXHLR   5                     /* max no. de capas ocultadas  */
#define NMXOATTR 50                    /* max no. de salidas futuras  */
#define NMXINP   200                   /* max no. de entradas muestra  */
#define NMXIATTR 50                    /* max no. de entradas futuras  */
#define SEXIT    3                     /* salida exitosa               */
#define RESTRT   2                     /* reinicio          */
#define FEXIT    1                     /* salida en fallo        */
#define CONTNE   0                     /* continúa calculando      */

/* Base de Datos: declaraciones de variables* /

float eta;                             /** tasa de aprendizaje          **/
float alpha;                           /** tasa del momentum            **/
float err_curr;                        /** normalizá error del sistema **/
float maxe;                    /** maximo error permitido por el sistema **/
float maxep;                   /** maximo error permitido por el patrón **/
float *wtptr[NMXHLR+1];
float *outptr[NMXHLR+2];
float *errptr[NMXHLR+2];
float *delw[NMXHLR+1];
float target[NMXINP][NMXOATTR];
float input[NMXINP][NMXIATTR], ep[NMXINP];
float outpt[NMXINP][NMXOATTR];
int nuinit[NMXHLR+2], nhlayer, ninput, ninattr, noutattr;
int result, cnt, cnt_num;
int nsnew, nsold;
char task_name[20];
FILE *fp1, *fp2, *fp3, *fopen();
int fplot10;

  /*  generación de números aleatorios (computos independientes)  */
long randseed = 568731L;
int random()
{
 randseed = 15625L * randseed + 22221L;
 return((randseed >> 16) & 0x7FFF);
}

/* asignación de almacenamiento dinamico para la red */
```

```
unit()
{
 int len1, len2, i, k;
 float *p1, *p2, *p3, *p4;
 len1=len2=0;
 nunit[nhlayer+2]=0;

 for (i=0; i<(nhlayer+2); i++) {
   len1+= (nunit[i]+1)*nunit[i+1];
   len2+= nunit[i]+1;
 }
              /* pesos */
 p1=(float *) calloc(len1+1,sizeof(float));
              /* salidas  */
 p2=(float *) calloc(len2+1,sizeof(float));
              /* errores  */
 p3=(float *) calloc(len2+1,sizeof(float));
              /* delw     */
 p4=(float *) calloc(len1+1,sizeof(float));

        /* adaptación de apuntadores iniciales */

wtptr[0]  = p1;
outptr[0] = p2;
errptr[0] = p3;
delw[0]   = p4;

        /* adaptación del resto de apuntadores  */

for (i=1 ; i<(nhlayer+1); i++)  {
   wtptr[i] = wtptr[i-1]+nunit[i]*(nunit[i-1]+1);
   delw[i] = delw[i-1]+nunit[i]*(nunit[i-1]+1);
}
for (i=1;i<(nhlayer+2); i++)  {
   salida[i]=outptr[i-1]+nunit[i-1]+1;
   errptr[i]=errptr[i-1]+nunit[i-1]+1;
}
```

```
        /* adaptación de salidas de umbral */
for (i=0; i<nhlayer + 1; i++)    {
  *(outptr[i] + nunit[i]) = 1.0;
 }
}
            /* inicialización de pesos con valores aleatorios entre -0.5 y +0.5 */
initwt()
{
  int i,j;

  for (j=0;  j<nhlayer+1; j++)
    for (i=0; i<(nunit[j]+1) * nunit[j+1]; i++)  {
      *(wtptr[j]+i)=random()/pow(2.0,15,0)-0.5;
      *(delw[j]+1)=0.0;
  }
}
          /* especifica arquitectura de valores y parámetros de la red de aprendizaje */
set_up()
{
 int i;

 eta=0.9;
 printf("\n Tasa de Momentum (valor por omisión=0.9)?: ");
 scanf("%f",&eta);

 alpha=0.7;
 printf("\n Tasa de aprendizaje alfa (valor por omisión=0.7)?: ");
 scanf("%f",&alpha);

 maxe=0.01; maxep=0.001;
 printf("\n Maximo error total (valor por omisión =0.01)?: ");
 scanf("%f",&maxe);

 printf("\nMaximo error individual (valor por omisión =0.001)?: ");
 scanf("%f",&maxep);

 cnt_num=1000;
 printf("\nMaximo número de iteraciones (valor por omisión=1000)?: ");
 scanf("%d", &cnt_num);

 printf("\nNúmero de capas ocultas?: ");
 scanf("%d", &nhlayer);

 for (i=0; i<nhlayer; i++)  {
```

```
    printf("\n\tNúmero de unidades para capas ocultas %d?: ",
        i+1);
    scanf("%d",&nunit[i+1]);
  }
  printf("\nCreación del archivo de error? Si 1 ó No 0 : ");
  scanf("%d",&fplot10);

  printf("\nComienzo");

  nunit[nhlayer+1]=noutattr;
  nunit[0]=ninattr;
  }
    /* lectura del archivo para la arquitectura de la red y los parámetros de aprendizaje.
    El nombre del archivo tiene sufijo _v.dat  */

  wtread (taskname)
  char *taskname;
  {
   int i,j,c;
   char var_file_name[20];

   strcpy(var_file_name, taskname);
   strcat(var_file_name,"_v.dat");
   if ((fp1=fopen(var_file_name,"r"))==NULL)
   {
     perror("\n No puede abrirse fichero de datos ");
     exit(0);
   }

  fscanf(fp1,"%d%d%d%f%f%d%d",&ninput,&noutattr,&ninattr,&eta,&alpha,&nhlayer,&
  cnt_num);

   for (i=0; i<nhlayer+2; i++)
     fscanf (fp1, "%d", &nunit[i]);

   if ((c=fclose(fp1))  !=0)
    printf("\n Archivo no puede cerrarse  %d ",c);
  }
          /* Lectura de los pesos y aumbrales contenidos en un archivo.
           El archivo tiene sufijo _w.dat  */

  wtread(taskname)
  char *taskname;
  {
```

```
  int i,j,c;
  char wt_file_name[20];

  strcpy(wt_file_name,taskname);
  strcat(wt_file_name,"_w.dat");

  if((fp2=fopen(wt_file_name,"r"))==NULL)
  {
   perror("\n El archivo no se puede abrir ");
   exit(0);
   }

   for (i=0;i<nhlayer+1; i++) {
    for (j=0; j<(nunit[i]+1)*nunit[i+1]; j++) {
      fscanf (fp2,"%f", (wtptr[i]+j));
   }
}
 if ((c=fclose(fp2)) !=0)
   printf("\n El archivo no puede ser cerrado %d ", c);
 }

        /* Creación del archivo para la arquitectura y los parámetros de aprendizaje.
        El archivo tiene sufijo _v.dat */

dwrite(taskname)
char *taskname;
{
   int i,j,c;
char var_file_name[20];
strcpy(var_file_name,taskname);
strcat(var_file_name, "_v.dat");

if ((fp1 =fopen(var_file_name,"w+")) ==NULL)
 {
   perror(" El archivo no puede abrirse ");
   exit(0);
   }
fprintf (fp1, "%u %u %u %f %f %u %u\n",ninput,noutattr,ninattr,eta,alpha,nhlayer,
nt_num);

   for (i=o; i<nhlayer+2;i++) {
      fprintf(fp1, "%d ,nunit[i]);
  }
  fprintf(fp1, "\n%d  %f", cnt,err_curr);
```

```
 fprintf(fp1, "\n);
 for(i=o;i<ninput;i++)
 {
   for(j=o;j<noutattr;j++)
     fprintf(fp1,"%f    ",outpt[i][j]);
   fprintf(fp1, "\n");
 }
 if ((c=fclose(fp1)) !=O)
   printf("\nEl archivo no puede ser cerrado %d ", c);
 }
```

/*Creación del archivo para salvar los pesos y umbrales aprendidos en el entrenamiento. El archivo tiene sufijo _w.dat */

```
 wtwrite(taskname)
 char *taskname;
 {
 int  i, j, c, k;
 char wt_file_name[20];
 strcpy(wt_file_name,taskname);
 strcat(wt_file_name, "_w.dat");
 if (( fp2 =fopen(wt_file_name,"w+")) ==NULL)
 {
   perror("El archivo no puede abrirse");
   exit(o);
 }

k=o;
for (i=o; i<nhlayer+1; i++)
  for( j=o; j <(nunit[i] +1) *nunit[i+1]; j++) {
    if(k==8) {
      k=o;
      fprintf(fp2,"\n");
    }
    fprintf(fp2, "%f ", *(Wtptr[i] +j));
    k++;
    }
    if ((c=fclose(fp2)) !=o)
      printf("\nEl archivo no puede cerrarse %d ", c);
    }

          /* bottom_up cálculo de la red para patrones de entrada */
  void forward(i)
  {
```

```
    int  m, n, p, offset;
    float net;

                /* Cálculo de la salida para el nivel de entrada*/
   for (n=o; m<ninattr;m++)
     *(outptr[o]+m) = input[i] [m];

    /*Cálculos para la salida de la capa oculta*/
   for (m=1; m<nhlayer+2;m++) {
    for (n=0; n<nunit[m]; n++) {
     net=0.0;
     for (p=0; p<nunit[m-1]+1;p++) {
      offset = (nunit[m-1]+1)*n+p;
      net += *(wtptr[m-1]+offset)
        *(*(outptr[m-1]+p));
  }
  *(outptr[m]+n)=1/(1+exp(-net));
 }
}
for (n=0; n<nunit[nhlayer+1];n++)
 outpt[i][n] = *(outptr[nhlayer+1]+n);
}

      /* Varias condiciones son revisadas para ver si el aprendizaje debería terminar  */

int introspective (nfrom,nto)
int nfrom;
int nto;
{
  int i,flag;
      /* realizar maximo de iteraciones ?*/
  if (cnt>=cnt_num) return(FEXIT);
      /* error para cada patrón pequeño suficiente? */
nsnew=0;
flag=1;
for (i=nfrom; (i<nto) && (flag==1); i++)  {
 if (ep[i] <=maxep) nsnew++;
 else flag = 0;
}
if (flag==1) return(SEXIT);

         /* Error total del sistema suficientemente pequeño? */
if (err_curr <= maxe) return (SEXIT);
return(CONTNE);
```

```
}

   /* Umbral está tratado como pesos de enlace de un nodo virtual cuyo valor de salida
es unidad */

int rumelhart(from_snum,to_snum)
int from_snum;
int to_snum;
{
 int i,j,k,m,n,p,offset,index;
 float out;
 char *err_file="criter.dat";

nsold=0;
cnt=0;
result=CONTNE;

if(fplot10==1)
 if((fp3=fopen(err_file,"w"))==NUL)
 {
  perror("No puede abrirse el archivo de error");
  exit(0);
  }
  do {
  err_curr=0.0;
    /* Para cada patrón  */
    for(i=from_snum; i<to_snum;i++) {
      /* bottom_up calculos */
      forward(i);
        /* top_down propagación de error */
        /* salida_level error  */
      for (m=0; m<nunit[nhlayer+1];m++) {
       out=*(outptr[nhlayer+1]+m);
       *(errptr[nhlayer+1]+m)=(target[i][m]-out)
                    * (1-out)*out;
}
    /* error de la capa de entrada y capas ocultas */
  for (m=nhlayer+1; m>=1; m--) {
   for (n=0; n<nunit[m-1]+1; n++) {
    *(errptr[m-1]+n)=0.0;
    for(p=0; p<nunit[m]; p++) {
       offset=(nunit[m-1]+1)*p+n;
       *(delw[m-1]+offset)=eta * (*(errptr[m]+p))
         *(*(outptr[m-1]+n))
```

```
            +alpha*(*(delw[m-1]+offset));
            *(errptr[m-1]+n) += *(errptr[m]+p)
                  *(*(wtptr[m-1]+offset));
      }
      *(errptr[m-1]+n)=*(errptr[m-1]+n)*
        (1-*(outptr[m-1]+n))*(*(outptr[m-1]
        +n));
    }
  }
    /* cambio de pesos */
  for (m=1; m<nhlayer+2; m++)  {
   for(n=0; n<nunit[m]; n++)  {
     for (p=0; p<nunit[m-1]+1;p++)  {
       offset=(nunit[m-1]+1)*n+p;
       *(wtptr[m-1]+offset) += *(delw[m-1]
       +offset);
     }
   }
  }
  ep[i]=0.0;
  for (m=0; m<nunit[nhlayer+1]; m++)  {
   ep[i] += fabs((target[i][m]-
        *(outptr[nhlayer+1]+m)));
  }
  err_curr +=ep[i];
 }     /* normalización del error del sistema  */
 err_curr=0.5 * err_curr/ninput;

    /** salvar errores en archivo para dibujar el error del sistema con fplot10  **/
 if (fplot10==1)
 fprintf(fp3,"%1d,%2.9f\n",cnt,err_curr);
 cnt++;

      /* revisar condiciones para terminar aprendizaje  */
  result=introspective(from_snum,to_snum);
  } while (result==CONTNE);

/* Actualiza salidas con cambios de pesos */
for (i=from_snum; i<to_snum;i++) forward(i);

for (i=0;i<nhalayer+1; i++) {
 index=0;
 for (j=0; j<nunit[i+1]; j++)
 {
```

```
   printf("\n\n pesos entre unidad %d de capa %d",
    j,i+1);
   printf(" y unidades de capa %d\n",i);
   for(k=0;k<nunit[i];k++)
    printf(" %f",*(wtptr[i]+index++));
   printf("\n Umbral de unidad %d de capa %d
   if %f",
        j,i+1,  *(wtptr[i]+index++));
   }
  }
  for (i=0; i<ninput;i++)
   for (j=0;j<noutattr;j++)
    printf("\n\n muestra %d salida %d=%f objeto %d=%f",i,j,outpt[i][j],j,target[i][j]);
printf ("\n\nEl número total de iteraciones es %d",cnt);
printf ("\nEl sistema fue normalizado con un error de  %f\n\n\n",
 err_curr);
 return(result);
}
   /* lectura de los datos de entrada de el archivo especificado por el usuario durante la
sesión interactiva */
user_session()
{
  int  i,j,showdata;
  char fnam[20],dtype[20];
  FILE *fp;

printf("\n Comienzo de la sesión de aprendizaje");

     /* para tarea con nombre task_name, archivo de datos de entrada de la tarea es
automáticamente establecer para ser task_name.dat por el programa.
 */
pintf("\n\t Entrada del nombre de tarea: ");
scanf("%s", task_name);

printf("\n Cuantos caracteres distintos existen en el patrón de entrada?; ");
scanf ("%d", &ninattr);

printf("\n Cuantas unidades de salida?: ");
scanf("%d",&noutattr);

printf("\n Número Total de Entradas Muestra?: ");
scanf("%d",&ninput);

strcpy(fnam,task_name);
```

```
strcat(fnam,".dat");
printf("\n El nombre del archivo de entrada es %s ", fnam);
if((fp=fopen(fnam,"r"))==NUL)
{
 printf("\n El archivo  %s no existe",fnam);
 exit(0);
}
printf("\n Quiere usted mirar la información que va a leer?");
printf("\n Responda Yes o No : ");
scanf("%s",dtype);
showdata=((dtype[0]=='y') || (dtype[0] == 'Y'));
for (i=0;i<ninput;i++)  {
  for (j=0; j<ninattr;j++)  {
    fscanf(fp,"%f",&input[i][j]);
    if (showdata) printf("%f  ",input[i][j]);
   }
   for (j=0; j<noutattr;j++)  {
    fscanf(fp,"%f",&target[i][j]);
    if (showdata) printf("%f\n",target[i][j]);
  }
 }
 if(i=fclose(fp)) !=0)
 {
  printf("\nFile cannot be closed "d",i);
  exit(0);
 }
}

   /* cuerpo principal del aprendizaje  */

learning()
{
 int result;
 user_session();
 set_up();
 init();
 do {
  initwt();
  result=rumelhart(0,ninput);
}while (result==RESTRT);
if (result==FEXIT)
{
 printf("\n Maximo numero de iteraciones que se alcanzó, ");
 printf("\n  pero fallo al disminuir el sistema");
```

```
  printf("\n error suficientememte");
}
dwrite(task_name);
wtwrite(task_name);
}

/*  cuerpo principal de la generación de salida */
salida_generation()
{
 int i,m,nejemplo;
 char ans[10];
 char dfile[20];

/* Si la tarea estuviera en la memoria, los ficheros de datos para la tarea no b¿necesitan
ser leidos. Pero si fuera una nueva tarea, unos ficheros de datos deberían estar leidos para
reconstruir la red  */

printf("\nGeneración de salidas para un nuevo patrón");
printf("\n\t Presentación del nombre de la tares %s", task_name);
printf("\n\t Trabaja con una tarea diferente? ");
printf("\n\t  Responsa Yes o No : ");
scanf("%s",ans);
if ((ans[0]=='y') || (ans[0]=='Y'))
{
 printf("\n\t Teclea el nombre de la tarea: ");
 scanf(%s",task_name);
 dread(task_name);
 init();
 wtread(task_name);
}
     /* Datos de Entrada para geración de salida están creados */

printf("\nNombre del archivo de patrones ");
printf(" esta procesando: ");
scanf("%s",dfile);
if ((fp1=fopen(dfile,"r"))==NULL)
{
 perror("No puede abrirse archivo");
 exit(0);
 }
 printf("\nNúmero de patrones para procesar: ");
 scanf("%d", &nejemplo);

for(i=0;i<nejemplo;i++)
```

```
  for(m=0;m<ninattr;m++)
   fscanf(fp1,"%f",&input[i][m]);

     /* Salida generada por los cálculos realizados */
   for(i=0;i<nejemplo;i++)
   {
    forward(i);
     for (m=0;m<noutattr;m++)
      printf("\n ejemplo %d salida %d=%f",
       i,m,*(outptr[nhlayer+1]+m));
     printf("\n");
   }
   printf("\nSalidas generadas ");

   if((i=fclose(fp1)) !=0)
    printf("\n El archivo no puede ser cerrado %d",i);
  }

/***************** MAIN ********************/

main()
{
 char select[20],cont[10];
  strcpy(task_name, "**********");

  do {
   printf("\n** Seleccionar L(Aprendizaje) or O(Salida Generada)
   **\n");
   do {
    scanf("%s",select);
    switch(select[0]) {
     case 'o':
     case 'O':
      salida_generation();
      break;
     case 'l':
     case 'L':
      learning();
      break;
     default():
      printf("\n No seleccionó algunas de las dos opciones ");
      break;
  }
}while ((select[0]!='o') && (select[0]!='O')
```

```
    &&(select[0]!='I') && (select[0]!='L'));
  printf("\n Desea usted continuar? ");
  scanf("%s",cont);
} while((cont[0]=='y') || (cont[0]=='Y'));
  printf("\n Esto a finalizado. ");
  printf("\n Adios ");
}
```

Ejemplo:

Archivo de datos de entrada por paridad-3 de aprendizaje: par3.dat.

```
0  0  0     0.9
0  0  1     0.1
0  1  0     0.1
0  1  1     0.9
1  0  0     0.1
1  0  1     0.9
1  1  0     0.9
1  1  1     0.1
```

Archivo de Datos para la Generacion de Salida: data.dat.

```
0    0    0
1    1    1
0.0  0.0  0.2
0.0  0.0  0.8
0.0  0.2  0.0
0.0  0.8  0.0
0.0  0.2  0.2
0.0  0.8  0.8
```

Salida de la corrida del programa.

```
** Seleccionar L(Aprendizaje) o O( Salida Generada) **
L

Comienza sesion de Aprendizaje
Nombre de la tarea: par3

Número de entradas del patrón de entrada?: 3

Número de unidades de salida?: 1
```

Total de número de entradas ejemplo?: 8

Nombre del archivo de la tarea: part3.dat
Desea leer los datos del proceso de ajuste?

Conteste yes o no: y

0.000000	0.000000	0.000000	0.900000
0.000000	0.000000	1.000000	0.100000
0.000000	1.000000	0.000000	0.100000
0.000000	1.000000	1.000000	0.900000
1.000000	0.000000	0.000000	0.100000
1.000000	0.000000	1.000000	0.900000
1.000000	1.000000	0.000000	0.900000
1.000000	1.000000	1.000000	0.100000

Valor momentum eta (valor por omisión=0.9)?: 0.9

Valor de aprendizaje alpha (valor por omisión=0.7)?: 0.7

Max total error (valor por omisión=0.01)?: 0.000001

Max individual errpr (valor por omisión=0.001)?: 0.0000001

Max número de iteraciones(valor por omisión=1000)?: 2000

Número de capas ocultas?: 1

Número de unidades ocultas por capa oculta ?: 3

Crear archivo de error? Si =1, o No=0: 1

Comienzo de Ejecución

pesos de las unidades 0 de la capa 1 y unidad de capa 0
-2.755273 -2.755553 -2.756222
El resultado de la unidad 0 y la capa 1 es 6.599238

pesos de las unidades 1 de la capa 1 y unidad de capa 0
-5.482076 -5.487528 -5.497730
El resultado de la unidad 1 y la capa 1 es 7.316314

pesos de las unidades 2 de la capa 1 y unidad de capa 0

-4.870187 -4.894925 -4.942179
El resultado de la unidad 2 y la capa 1 es 1.474228

pesos de las unidades 0 de la capa 2 y unidad de capa 1
7.695147 -7.613980 6.758780
El resultado de la unidad 0 y la capa 2 es -3.380892

ejemplo 0 salida 0 = 0.899753 target 0 = 0.900000

ejemplo 1 salida 0 = 0.100304 target 0 = 0.100000

ejemplo 2 salida 0 = 0.100339 target 0 = 0.100000

ejemplo 3 salida 0 = 0.899036 target 0 = 0.900000

ejemplo 4 salida 0 = 0.100365 target 0 = 0.100000

ejemplo 5 salida 0 = 0.898985 target 0 = 0.900000

ejemplo 6 salida 0 = 0.898907 target 0 = 0.900000

ejemplo 7 salida 0 = 0.103369 target 0 = 0.100000

Total de iteraciones 1384
El sistema se narmalizo con un error de 0.000001

Desea usted continua? y

** Seleccionar L(Aprendizaje) o O(Salida Generada) **
O

Generación de salidas para los nuevos patrones
La Tarea es la par3
Desea trabajar con diferente tarea?
Conteste o no : n

Nombre del archivo para geración de salida: data.dat

Numero de patrones a procesar: 8

ejemplo 0 salida 0 = 0.899753

ejemplo 1 salida 0 = 0.103369

ejemplo 2 salida 0 = 0.707187

ejemplo 3 salida 0 = 0.077218

ejemplo 4 salida 0 = 0.710287

ejemplo 5 salida 0 = 0.078334

ejemplo 6 salida 0 = 0.326657

ejemplo 7 salida 0 = 0.893754

Salidas generadas correctamente
Desea usted continuar? n

Eso es todo.
Adios

b) Aprendizaje No supervisado basado en el descubrimiento de estructuras de grupos. (Clustering)

Programa en C para utilizarse como punto de partida para el aprindizaje no supervisado por medio del descubrimiento de estruturas de grupos.

En este programa, los agrupamientos usan la distancia metrica Euclidiana para determinar la distancia entre patrones y centros de grupos. El programa corre bajo PS-DOS/MS-DOS y fue compilado con Microsoft C.

```
/* DESCRIPCION DEL PROGRAMA

ESTE PROGRAMA REALIZA EL APRENDIZAJE NO SUPERVISADO
UTILIZANDO LA DISTANCIA EUCLIDIANA METRICA. EL NUMERO MAXIMO
DE RACIMOS QUE PUEDEN CREARSE ES IGUAL AL NUMERO DE PATRONES
DE ENTRADA.

LA ENTRADA DE PATRONES Y PESOS APRENDIDOS ESTAN ALMACENADO
EN ARCHIVOS. * /


#include <stdio.h>
#include <ctype.h>
```

```
#include <math.h>

#define NMXPATTERN  100                 /* max número de entrada nuestra */
#define NMXATTR     100                 /* max número de atributos de entrada  */
int ninput;                             /* número de entrada de patrones  */
int ninattr;                            /* número deof input attributes  */
float threshold;                        /* umbral */
int testing;                            /* 0=training, 1=testing*/
float pattern[NMXPATTERN][NMXATTR];
float x[NMXATTR];                       /* entrada de patrones */
float ed[NMXPATTERN];                   /* Distancia Euclidina  */
int active_nodes;                       /* número de racimos actuales */
FILE *in,*out;
int debug_falg;
int total_time;                         /* número de iteraciones */

     /** obtención de información atravéd de un usuario  **/
user_session()
{
 int i,j;
 char file_name[20];

 printf("\n Despliege de Distancia Euclidiana en cada iteración?");
 printf("\n Yes=1,No=0: ");
 scanf("%d",&debug_flag);

 printf("\n\n Umbral: ");
 scanf("%f",&threshold);
 printf("\Nombre del archivo de datos de entrada: ");
 scanf(%s",file_name);
 if ((in=fopen(file_name,"r"))==NULL)
 {
   printf("\nNo puede abrirse el archivo\n");
   exit(0);
 }
 printf("\nCuantos patrones de entrada de entrenamiento?: ");
 scanf("%d",&ninput);
 printf("\nCuantos atributos de entrada): ");
 scanf("%d", &ninattr);

testing=0;
total_time=0;
}
/* Lee desde archivo de entrada la información del patrón */
```

```
read_all_patterns()
{
 int i,j;
 for(i=0;i<ninput;i++)
   for (j=0;j<ninattr; j++)
     fscanf(in,"%e",&pattern[i][j]);
   fclose(in);
 }
/* Creación del primer racimo con el primer patrón */
first_node()
{
 int i;
    /* bottom_up pesos=primer patrón */
 for(i=0;i<ninattr; i++)
   b[0][i]=pattern[0][i];

 active_nodes=1;   /* número de racimos */
 cluster_tbl[0][0]=1; /*número de miembros en cada racimo*/
}
     /* creación de un nuevo racimo  */
form_new_node(input_no)
int input_no;       /* entrada del número de patrones */
{
 int i;
 for(i=0;i<ninattr;i++)
   b[active_nodes][i]=pattern[input_no][i];

  cluster_tbl[active_nodes][0]=1; /* número de miembros=1  */
  cluster_tbl[active_nodes][1]=input_no;
  active_nodes++;    /* increm,ento del número de racimos */
  }
     /* Cálculo del la distancia Euclidiana para cada patron del racimo centro */
compute_euc_dist()
{
 int i,j;
 if(debug_flag==1)
  printf("\nDistancia Euclidiana de cada racimo centro\n");
 for(j=0;j<active_nodes;j++)
 {
  ed[j]=0.0;
  for (i=0;i<ninattr; i++)
    ed[j]=ed[j]+((b[j][i]-x[i])*(b[j][i]-x[i]));
  if(debug_flag==1) printf("%.6f", ed[j]);
  }
```

```
   if (debug_flag==1) printf ("\n");
  }
/* escoge el racimo más cercano al patrón y regresa el número de racimo si el patrón esta
dentro del radio del umbral. Si no hay ninguno disponible, entonces regresa- 99*


int compare_min_ed(input_no)
{
 int i, cluster_no;
 float min;

 min=10000;

 for (i=0;i<active_nodes;i++)
  if(ed[i] < min)
  {
   min=ed[i];
   cluster_no=i;
  }
 if (debug_flag==1)
  printf("Ed=%.3f Node=%d Pat=%d \n",
   sqrt(ed[cluster_no]),cluster_no,input_no);

 if (sqrt(ed[cluster_no])<=threshold)
  return(cluster_no);
 else return(-99);
}

/* incluye un nuevo miembro en el racimo actualizando pesos */

update_wts(cluster_no,input_no)
int cluster_no;
int input_no;
{
 int i,no_member;
 float n,m;
 n=cluster_tbl[cluster_no][0];
 m=n+1;
 if (testing==0)
 {
  for(i=0;i<ninattr;i++)
   b[cluster_no][i]=((n/m)*b[cluster_no][i])
              +((1/m)*x[i]);
 }
```

```
 no_member=++(cluster_tbl[cluster_no][0]);
 cluster_tbl[cluster_no][no_member]=input_no;
}
report()
{
 int i,j,k,nbcol;

 printf("\n\n");
 printf("+------+-----+----------------------------+\n");
 printf("| Nodo |Contador|      Número de Patrón     |\n");
 printf("+------+-----+----------------------------+\n");

 k=0;
 nbcol=6;
 for (i=0;i<ninput; i++)
 {
  if (cluster_tbl[i][0]==0) break;
  printf("| %-4d | %-3d |",i,cluster_tbl[i][0]);
  for (j=1;j<ninput+1;j++)
  {
   if ((cluster_tbl[i][j] !=0)
     || ((i==0)&&(j==1)))
  {
    if(k>nbcol)
    {
     printf8"|\n|    |   |");
     k=0;
    }
    printf(" %-3d",cluster_tbl[i][j]);
    k++;
  }
}
 for(j=k;j<=nbcol;j++) printf("    ");
 printf("|\n");
 printf("+------+-----+----------------------------+\n");
 k=0;
 }
}

print_bot_up_wts()
{
 int i,j;
 char *file_name='weights.dat";
```

```
 if ((out=fopen(file_name,"w"))==NULL)
 {
  printf("\n No puede abrirse archivo de datos de entrada\n");
  exit(0);
 }
 printf("\n\nBottom_up_weights");
 for(i=0;i<active_nodes; i++)
 {
  printf("\n Racimos %d\n  ",i);
  for(j=0;j<ninattr;j++)
  {
   printf("%.3f",b[i][j]);
   fprintf(out,"%.3f",b[i][j]);
  }
  printf("\n");
  }
  fclose(out);
}
main()
{
int cluster_no,q;
int i;

user_session();
read_all_patterns();
first_node();

for (q=1;q<ninput;q++)
{
 for (i=0;i<ninattr; i++)
  x[i]=pattern[q][i];
 compute_euc_dist();
 if((cluster_no=compare_min_ed(q)) >=0)
  update_wts(cluster_no,q);
 else
  form_new_node(q);
}

 report();    /*reporte de resultados */
 print_bot_up_wts();
}
```

Ejemplo:

Archivo de entrada de datos para agrupamiento: 8x7.dat

```
0.0 1.0 2.0 3.0 4.0 5.0 6.0
0.0 1.0 2.0 3.0 4.0 5.0 5.0
0.0 1.0 2.0 3.0 4.0 4.0 4.0
6.0 5.0 4.0 3.0 2.0 1.0 0.0
6.0 5.0 4.0 3.0 2.0 1.0 1.0
6.0 5.0 4.0 3.0 2.0 2.0 2.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Salida de ejecucion del programa

Despliege de la distancia Euclidiana para cada iteración Yes = 1,No = 0 ? : 0

Please enter threshold: 3
Número del arcivo de entradas: 8x7.dat
Número de entradas por patron?: 8
Número de entradas de atributos?: 7

Nodo	Contador	Número de Patrón
0	3	0 1 2
1	3	3 4 5
2	2	6 7

Bottom_up pesos
Cluster 0
0.000 1.000 2.000 3.000 4.000 4.667 5.000

Racimo 1
6.000 5.000 4.000 3.000 2.000 1.333 1.000

racimo 2
0.500 0.500 0.500 0.5000 0.500 0.500 0.500

3. 9. PROBLEMAS Y TIPOS DE SOLUCIONES DE ALGUNOS ALGORITMOS

Paradigma	Problema y tipo de solución	Necesidad tiene aprendizaje	Línea apagado de aprendizaje	Patrón Matching	Tipo de Aprendizaje	Limitaciones	Implementación
AG	Datos de clasificación no supervisados	Si	No	Si	Hebbian	Almacenamiento limitado	Optica y electrónica
SG	Información humana procesada	Si	No	Si	Hebbian	Almacenamiento limitado	Electrónica
ART1	Clasificación compleja de patrones	Si	No	Si		Restricciones binarias	Computadora Neuronal
DH	Procesamiento de voz (habla)	No	Si	No	Hebbian	Restricciones binarias	Optica electrónica VLSI
CH	Optimización combinacional	No	Si	Si	Hebbian	Almacenamiento limitado	Optica electrónica VLSI
BAM	Reconocimiento de Imágenes	No	Si	Si	Hebbian	Extensivo apagado de linea de aprendizaje.	Optica y Computadoras Neuronales
TAM	Almacenamiento especial temporal de patrones.	No	Si	No	Hebbian	Extensivo apagado de línea de aprendizaje.	Simulación de computadora
ABAM	Procesamiento analógico de patrones en tiempos continuos	Si	No	Si	Hebbian	Almacenamiento limitado	Computadora Neuronal
CABAM	Optimización combinacional	Si	Si	Si	Hebbian	Almacenamiento limitado	Simulación de computadora
FCM	Optimización combinacional	Si	Si	No	Hebbian	Extensivo apagado de línea de aprendizaje.	Simulación de computadora

Tabla 3.6. Aprendizaje no supervisado: Llamada Feedback en Sistemas de Redes Neuronales Artificiales

Paradigma	Problema y tipo de solución	Entrenable línea aprendizaje	Línea apagado de aprendizaje	Patrón Matching	Tipo de Aprendizaje	Limitaciones	Implementación
LM	Proceso de Monitores	No	Si	Si	Hebbian	No reporta	Mecanismos electrónico magnéticos
DR	Predicción	No	Si	Si	Hebbian	Restricciones en tiempo discreto	Simulación de Robots
LAM	Sistema de Control	No	Si	Si	Hebbian	Almacenamiento limitado	Simulación de computadoras
OLAM	Procesamiento de Señales	No	Si	Si	Hebbian	Extensivo apagado de línea de aprendizaje	Optica
FAM	Procesamiento del conocimiento	Si	No	Si	Hebbian	Almacenamiento limitado	Simulación de Computadoras y chip de silicon

Tabla 3.7. Aprendizaje no supervisado: Llamada Feedforward en Sistemas de Redes Neuronales Artificiales

Paradigma	Problema y tipo de solución	Entrenable línea aprendizaje	Línea apagado de aprendizaje	Patrón Matching	Tipo de Aprendizaje	Limitaciones	Implementación
BSB		No	Si	Si	Corrección de error	Extensivo apagado de línea de aprendizaje	Simulación de computadora.

Tabla 3.8. Aprendizaje supervisado: Llamada Feedforward en Sistemas de Redes Neuronales Artificiales

Paradigma	Problema y tipo de solución	Necesidad hora aprendizaje	Línea apagada de aprendizaje	Patrón Matching	Tipo de Aprendizaje	Limitaciones	Implementación
Perceptrón	Predicción	No	Si	Si	Corrección de error.	No reportadas	Maquina Mark I perceptrón
ADELINE/ MADELINE	Eliminación de ruido predicción	No	Si	Si	Corrección de error.	Extensivo apagado de línea de aprendizaje	Optica Magnética
BP	Reconocimiento criptico de caracteres.	No	Si	Si	Corrección de error.	Extensivo apagado de línea de aprendizaje	Optica Electrónica VLSI
AVQ	Organización de datos por si mismo.	No	Si	Si	Corrección de error.	Extensivo apagado de línea de aprendizaje	Computadora Neuronal VLSI
CPN	Programación por si mismo.	No	Si	Si	Hebbian	Carece de control w/o del mapa	Computadora Neuronal
BM	Optimización combinacional	No	Si	Si	Hebbian y simulación de reconocimiento	Extensivo apagado de línea de aprendizaje	Optica electrónica VLSI
CM	Optimización combinacional	No	Si	Si	Hebbian y simulación de reconocimiento	Extensivo apagado de línea de aprendizaje	Optica
AHC	Predicción de control	Si	No	No	Corrección de error y recompensa/ maltrato	Restringidas entradas bipolares	Simulación de computadora
ARP		Si	No	Si	Información estocastica	Extensivo apagado de línea de aprendizaje	Simulación de computadora
SNMF	Reconocimiento del habla e imagen.	No	Si	Si	Hebbian	Extensivo apagado de línea de aprendizaje	Simulación de computadora

Tabla 3.9. Aprendizaje supervisado: Llamada Feedforward en Sistemas de Redes Neuronales Artificiales

CAPITULO IV

APLICACION DE LA RED NEURONAL PARA EL RECONOCIMIENTO DE HUELLAS

En este capitulo se describen los elementos que se utilizaron para la modelación de la Red Neuronal, y los pasos que se realizaron para dar solución al problema planteado.

4.1. RECONOCIMIENTO DE HUELLAS DIGITALES.

Agencias Gubernamentales de Estados Unidos mantienen archivos de más de 200 millones de huellas digitales. La División de Identificación de la Investigación Federal de Bureau, por ejemplo, opera el más grande archivo de huellas digitales (600 millones). Esta División recibe arriba de 30,000 solicitudes de investigación diariamente.

Para adoptar información de este volumen, algunos 1,400 técnicos y empleados son requeridos para desempeñar la meticulosa tarea de clasificación y subsecuentemente búsqueda de un par de huellas.

Hace algunos años la FBI se interesó por el desarrollo automático de sistemas para identificar huellas digitales. Como ejemplo del esfuerzo en esta área es el sistema prototipo, llamado FINDER, desarrollado por la Calspan Corporation por el FBI. Este sistema detecta y localiza características únicas en una impresión.

Las características leídas no son densas estructuras, tales como arcos, curvas o espirales usados en la clasificación primaria, pero bastante minucioso el final y horquillas de ondulaciones mostradas en la figura 4.1.

Fig. 4.1. Muestra -las terminales acanaladas (cuadrados) y bifurcaciones (circulos)- utilizada por el sistema FINDER para identificación de huellas dactilares.

Un diagrama a bloques del sistema es mostrado en la figura 4.2 FINDER y opera como sigue:

El operador carga la tarjeta de huellas digitales estándar dentro del manejador automático, el cual mueve y precisa la posición de las impresoras bajo el "ojo" del sistema del scaner.

Cada huella dactilar es digitalizada dentro de una matriz de 750x750 puntos donde cada punto es uno de 16 sombras de grises. El scan funciona bajo el control del propósito general de la computadora.

La información del scaner es alimentada dentro del filtro de ondulación cumbre en alta velocidad, paralelo, 2 algoritmos de procesamiento dimensional que secuencialmente examinan cada punto de los 750x750 de la matriz.

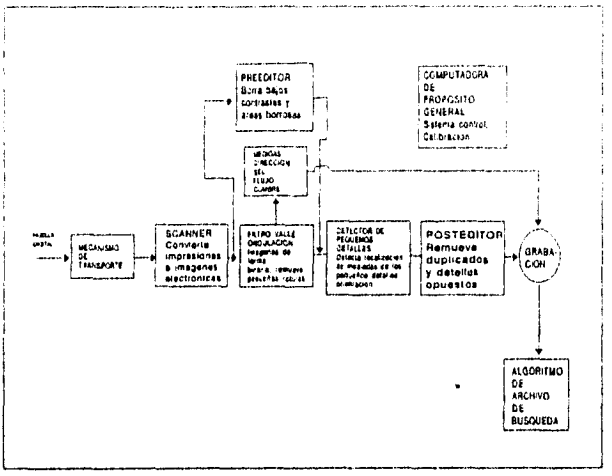


Fig. 4.2. El diagrama a bloques del sistema de huellas dactilares FINDER.

La salida de este algoritmo es una imagen binaria mejorada, como se muestra en la figura 4.3. Este algoritmo también graba la dirección curva de cada punto de la huella digital para un procesamiento subsecuente. Sobre algunas porciones de muchas huellas digitales, la estructura de ondulación clara no puede ser desarrollada para leer confiablemente. Estas tareas son tomadas en el preeditor, el cual excluye a ellas lo más lejos de la consideración de la fuente original.

12	2	6	15	14	13	15	6	8	9	6	5	12	0	12	2	15	11	12	12	0	6
15	12	12	12	13	11	2	2	5	6	7	9	8	9	12	11	15	14	14	9	6	5
12	10	1	5	14	5	5	15	10	15	8	5	8	0	2	3	5	6	8	9	12	4
12	11	3	9	15	9	6	15	11	12	9	6	9	1	5	6	7	8	9	12	15	2
11	12	5	2	16	6	9	12	12	1	12	7	5	12	4	5	6	11	12	13	14	2
9	13	6	2	8	4	10	12	11	3	14	9	4	0	15	12	13	14	15	14	9	5
6	6	8	4	9	7	12	10	15	6	14	7	2	0	14	12	14	14	8	9	14	6
5	5	9	5	11	6	12	11	6	6	12	9	1	2	12	13	15	13	4	5	7	11
	4	10	9	15	9	15	12	4	8	11	5	11	3	13	12	14	12	2	2	5	12
1	0	11	7	14	8	15	15	8	12	12	1	12	4	10	15	12	11	12	12	12	15
5	6	15	8	13	5	11	6	9	13	13	3	12	5	11	14	11	10	15	5	6	6
8	9	0	9	15	4	10	9	7	15	14	4	13	6	12	12	11	9	10	9	4	8
9	2	14	10	14	5	11	6	8	16	15	6	14	8	13	14	9	5	10	4	6	9
11	4	5	11	11	2	12	9	9	14	15	8	15	8	15	15	8	0	5	8	6	7
14	5	6	11	2	11	13	9	10	13	14	9	12	6	15	15	6	4	12	5		6
15	2	12	12	5	15	14	4	11	14	12	4	14	15	14	14	4	9	15	2	1	4
5	6	15	8	13	5	11	6	9	13	13	3	12	5	11	14	11	10	15	5	6	6
1	3	5	15	15	14	13	12	10	9	0	5	4	3	6	7	8	8	12	13	5	6
12	15	15	12	14	11	10	0	9	1	5	6	4	6	9	8	1	2	5	6	12	14
11	6	7	15	6	16	5	5	12	15	14	0	12	14	15	15	2	15	10	12	0	1
8	9	6	14	8	19	6	4	15	15	6	5	11	12	14	15	1	11	12	14	13	11

Fig. 4.3. La impresión de una sección de salida del rastreador. En esta representación numérica negro tiene 0 y blanco 15.

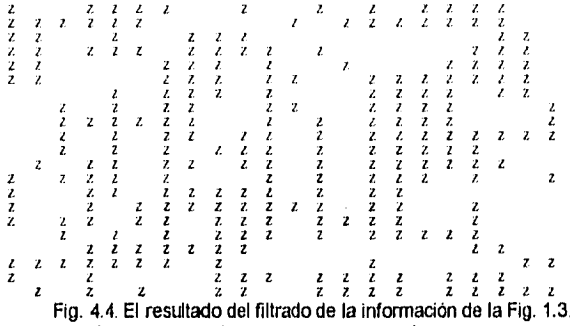


Fig. 4.4. El resultado del filtrado de la información de la Fig. 1.3. Se representan los puntos negros con z's.

Las pruebas son hechas por blancuras, oscuridades, por cadenas de estructura insuficientes o contrastes para permitir la detección de una información segura.

El siguiente paso en el proceso es la detección real de pequeños detalles. Este es acompañado por un algoritmo el cual sigue detrás del filtro de cadena. Este detecta pequeños detalles con suministros en la edición posterior. Primero, el área y el perímetro de los pequeños detalles detectándose son comparados con los umbrales que son característicos de verdaderos pequeños detalles para eliminar obviamente falsos datos. La duplicación de pequeños detalles son entonces fusiones. Si un pequeño detalles particular es detectado más de una vez, el que sea más largo es retenido. Por uso de una técnica, solo los pequeños detalles que están contiguos del que fue considerado son incluidos en el registro, reduciendo grandemente el tiempo. El siguiente, oponiéndose y cancelando pequeños detalles tales como resultado de pausa en estructura de cadena que son removidos. La lista de pequeños detalles es ahora escogida en factores de forma y calidad que permiten umbrales ciertos. Las etapas finales en el proceso de edición posterior determinan si un pequeño detalle es parte de una agrupación de pequeños detalles o si el ángulo pequeños detalles es significativamente diferente de la orientación de la estructura de cadena local. La prueba de agrupación descarta grupos de pequeños detalles tales como resultado de una marca de tecleada.

Si más de un número especifico de pequeños detalles son encontrados cerca del pequeño detalle que esta siendo analizado, el último es borrado como falso. Si el pequeño detalle pasa esta prueba, realiza una prueba lógica de anomalía de ángulo, usando una red de dirección de cadena de datos conectados durante el proceso. Dependiendo de la desviación del ángulo de cadena promedio; el pequeño detalle es también aceptado, rechazado o, si este es ligeramente fuera de línea; es ajustado al promedio de los ángulos de cadena de alrededor.

Finalmente, aproximadamente 2500 bits de datos definiendo la sobrevivencia de los pequeños detalles, todas las ediciones posteriores son grabadas en una cinta magnética para comprobación eventual contra pequeños detalles para impresiones en archivo.

4.2. SOLUCION AL PROBLEMA

Tomando como referencia lo anterior procedimos a dar solución al problema plateado. Para lo cual primeramente tuvimos que decidir el tipo de lenguajue, el algoritmo a utilizar y el tipo de red (asociativa o heteroasosiativa).

4.2.1. EL LENGUAJE DE PROGRAMACION

La descripción del código de una red neuronal, en forma de lenguaje de programación, se le denomina Neurosoftware.

David Zipser fue el primero en proponer explícitamente la idea de que un lenguaje formal debe ser creado en el propósito de describir la estructura funcional de las redes neuronales.

El principal atractivo de los lenguajes neurosoftware es que se puede construir la estructura común de todas las redes neuronales dentro del lenguaje. Así entonces, cada lenguaje puede ofrecer un incremento significativo en la facilidad de uso y eficiencia sobre las expresiones, es decir, las descripciones escritas en un lenguaje optimizado para la expresión de algoritmos (tales como el C, LISP, C++, PASCAL, etc.).

Los lenguajes neurosoftware generalmente se construyen en torno a un modelo de red neuronal. Para la construcción del lenguaje alrededor de un modelo, el programador es libre de tener que expresar repetidamente el conocimiento estructural contenido en el modelo. El uso de un modelo también simplifica la edición y modificación de una red, puesto que únicamente los cambios específicos por si mismos necesitan ser corregidos, y no los efectos colaterales de estos cambios. En realidad, los lenguajes para redes neuronales se prestan para los métodos de orientación a objetos, que actualmente se utilizan para eliminar las deficiencias del desarrollo tradicional de software (Programación estructurada).

A partir de diversos tipos de pseudocódigo con enfoque a redes neuronales, como el lenguaje AXON, se han desarrollado algoritmos codificados en varios lenguajes neurosoftware especializados para computadoras (también especializadas), en centros de investigación de Universidades (p.e. Cambridge) e Institutos científicos y tecnológicos (p.e. el Centro de Investigación SIEMENS, en Alemania, y la NASA, en EUA).

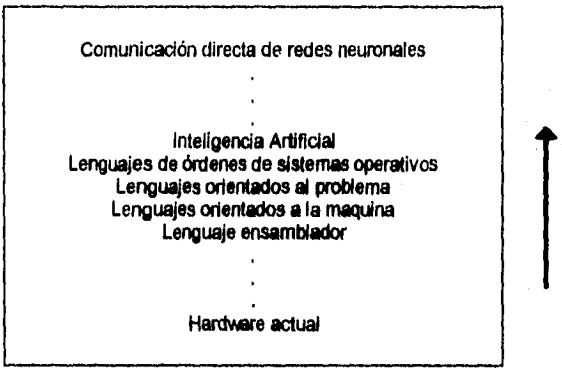
Sin embargo, por consecuencia del nuevo auge que está tomando la investigación y, sobre todo, el potencial de aplicación de las redes neuronales, se ha logrado desarrollar software más práctico que puede usarse en equipos de cómputo convencionales, de costo menor, y por ende, de mayor uso actualmente en muchas industrias. Este tipo de neurosoftware se ha codificado en los lenguajes de programación de mayor uso en el desarrollo de sistemas como son el C, el C++ y ocasionalmente el pascal.

Lenguaje C

C es lenguaje relativamente de bajo nivel, que permite especificar cada detalle en la lógica de un algoritmo para lograr la máxima eficiencia de la computadora. Pero también es un lenguaje de alto nivel que puede ocultar los detalles de la arquitectura de la computadora, incrementando así la eficiencia de la programación.

Relación con otros lenguajes.

Podemos definir esta jerarquía:



Comenzamos en la parte inferior de la jerarquía y nos desplazamos hacia arriba, los lenguajes van de lo tangible y empírico hacia lo evasivo y teórico. Los puntos representan los mayores adelantos con muchos pasos suprimidos.

El lenguaje C pertenece a un nivel superior al de los lenguajes orientados a la máquina, pero está un nivel por debajo de la mayoría de los lenguajes orientados al problema. Está suficientemente cerca de la computadora para dar un gran control sobre los detalles de la implementación de una aplicación, pero lo bastante lejos para ignorar los detalles del hardware. Por ello, el lenguaje C es simultáneamente un lenguaje de alto y bajo nivel.

Ventajas de C

A continuación se mencionarán algunas de las ventajas que tiene el lenguaje C, que a nuestro criterio son las más importantes:

- Pequeño tamaño.- C tiene menos reglas de sintaxis que muchos otros lenguajes y podemos escribir un compilador de C de alta calidad que trabaje solamente con 256K de memoria total.

- Velocidad.- El código C producido por la mayoría de los compiladores tiende a ser muy eficiente. La combinación de un lenguaje pequeño, un sistema de ejecución pequeño y un lenguaje cercano al hardware hace que la velocidad de ejecución de muchos programas en C se aproxime a la de sus equivalentes en lenguaje ensamblador.

- Un lenguaje estructurado.- C incluye todas las estructuras de control que esperaríamos de un lenguaje moderno (bucles *for*, construcciones *if* e *if-else*, instrucciones *case* y bucles *while)*. También permite separar código y datos controlando su ámbito, por ejemplo, proporciona variables locales para este propósito y llamadas por valor para la independencia de datos en las subrutinas.

- Manipulación de bits.- En programación de sistemas frecuentemente necesitamos manipular objetos a nivel de bits, el lenguaje proporciona un amplio conjuento de operadores de manipulación de bits.

- Portabilidad.- La portabilidad es una medida de la facilidad para transformar un programa que se ejecuta en una computadora o sistema operativo para que se ejecute en otra computadora o sistema operativo. Actualmente, los programas escritos en C están entre los más portables en el mundo de las computadoras.

Es muy difícil encontrar estas carcateristicas en otros lenguajes de programación, por lo anterior decidimos utilizar el lenguaje C para programar la red neuronal. Además es uno de los que mejor manejamos.

4.2.3. EL ALGORITMO

Para poder seleccionar un algoritmo de solución es necesario tomar en cuenta la topología de la red que se va a utilizar y el tipo de solución que se requiere.

En el capitulo 3, se describieron brevemente algunos algoritmos y se presentaron 4 tablas con los algoritmos mas usados con sus correspondientes problemas y tipos de solución. Dentro de los algoritmos que podían resolver nuestro problema "Reconocimiento de

huellas" (reconocimiento espacial), se encuentran los algoritmos: no supervisado (BAM) y supervisados (ADELINE/MADELINE, BP, BM, CM y OLAM).

Decidimos usar el de backpropagación por ser uno de los más usados para este tipo de problemas, porque converge rápidamente y porque su programación no representaba gran dificultad, además de ser muy fácil de comprender.

4.2.4. RESOLVIENDO EL PROBLEMA

El objetivo de este trabajo fue el de poder reconocer una huella dactilar utilizando una red neuronal y para ello fueron necesarios los siguientes pasos:

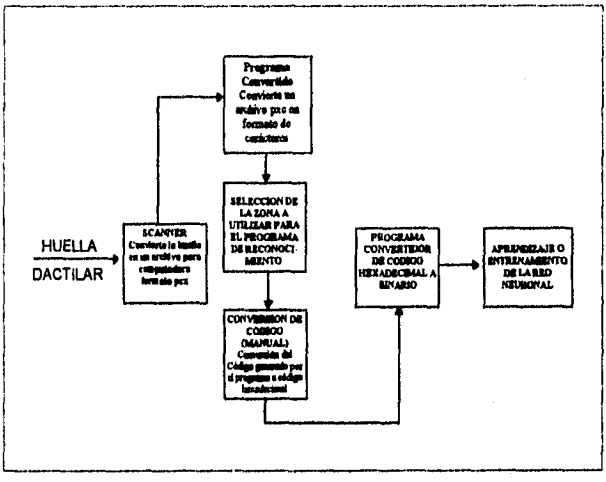


Fig. 4.5. Diagrama a bloques de la solución

Este diagrama muestra los pasos que se siguieron en el proceso para dar solución al problema planteado.

- Scanneo de una huella digital
- Convertir el código obtenido en código ascii, utilizando para ello un programa convertidor (de puntos a código hexadecimal), obteniéndose una matriz de datos de 174 x 185, es decir 139,800 datos.
- Obtención de puntos estratégicos de dicha huella para obtener un mínimo de datos.
- Elaboración y prueba del programa de entrenamiento, utilizando el lenguaje C.
-

Posteriormente se realizó el programa para el reconocimiento de las huellas, para lo cual se aplicaron diferentes entradas y se verificó que en realidad reconociera la huella para lo cual fue entrenada la red.

Aquí se muestra la huella dactilar scanueada.

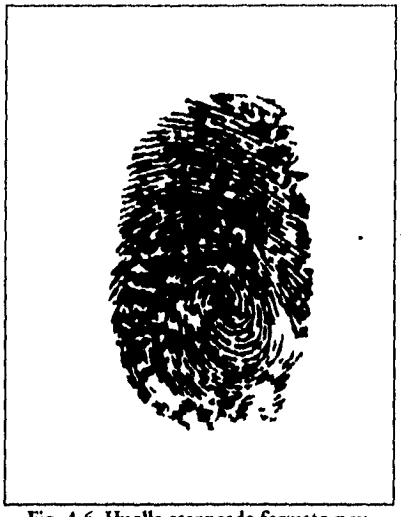


Fig. 4.6. Huella scanneada formato pcx.

El scanneo de la huella fue realizado en un scanner tamaño plano, lo que hizó que el archivo en formato pcx fuera muy grande, que no podía ser leída por el programa convertidor. Por lo anterior se tuvó que realizar una edición adicional utilizando el programa Painbrush.

La edición consistió en cargar el archivo pcx al programa Painbrush, posteriormente se copia la zona que comprende la huella en un archivo nuevo con los atributos de imagen : Unidades: pixeles y Color: blanco y negro.

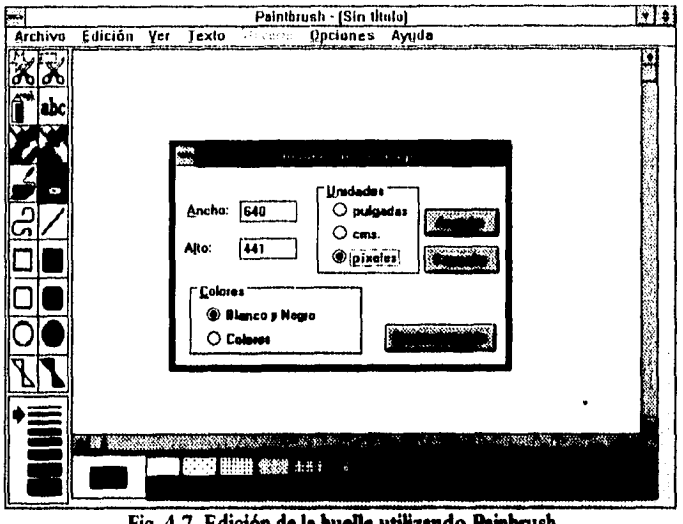


Fig. 4.7. **Edición de la huella utilizando Paintbrush**

Lo anterior nos permite crear un archivo lo suficientemente pequeño, que pueda ser procesado por el programa convertidor.

Este programa utiliza la instrucción [ESC] XD de Basic cuyas características son:

Función: Escribe caracteres y logos en la tarjeta de memoria.

El formatoes [ESC] xd; aa, b, ccc, ddd, eee, fff, ggg, h, iii iii[LF] [NULL]

Términos :	aa:	El conjunto de caracteres de la tabla 01 a 40
	b:	El código de carácter de la tabla 20h a FFh (hexadecimal)
	ccc:	El offset izquierdo 000 a 719 (en puntos)
	ddd:	El offset superior 000 a 719 (en puntos)
	eee:	El ancho del carácter 001 a 720 (en puntos)
	fff:	El ancho del carácter 001 a 720 (en puntos)
	ggg:	El espaciado horizontal/espacio proporcional 000 a 999 (en puntos)

h: El tipo de la tabla de información del carácter
0: El modo Nibble (4 bits / byte)
1: El modo Hexadecimal (8 bits / byte)

iii iii: La información del carácter de la tabla para estar almacenado.

La explicación de este programa no se realizó porque se estaría saliendo del orden, además unicamente se utilizó como una herramienta adicional. Solamente se describe brevemente su funcionamiento.

La forma en la cual trabaja el programa es la siguiente:

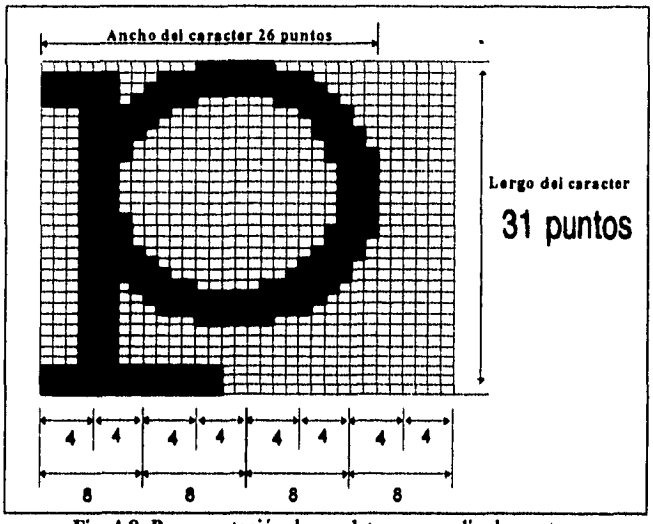


Fig. 4.8. Representación de una letra por medio de puntos.

Modo Nibble

1	30H	2	30H	3	30H	4	3FH	5	3CH	6	30H	7	30H	8	30H
9	3FH	10	3CH	11	37H										
														240	30H
241	3FH	242	3FH	243	3FH	244	3CH	245	30H	246	30H	247	30H	248	30H

Fig. 4.9. Tabla del código producida por el programa con el Modo Nibble.

MODO HEXADECIMAL							
1	00H	2	0FH	3	C0H	4	00H
5	FCH	6	7FH				
				.			
				.			
				.			
				.			
				.			
				.		120	00H
121	FFH	122	FCH	123	00H	124	00H

Fig. 4.10. Tabla del código producida por el programa con el Modo Hexadecimal.

Parte del programa convertidor es el siguiente:

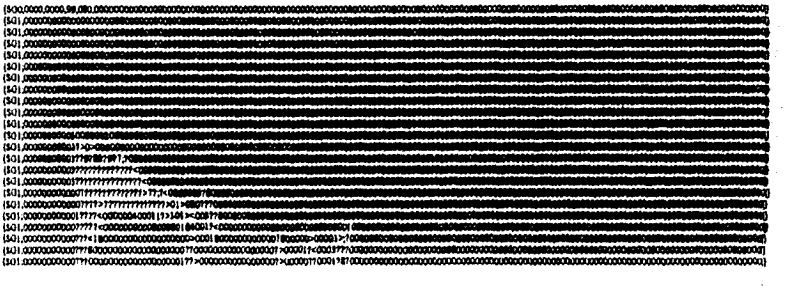
```
5 OPEN " H-1.PCX" FOR OUTPUT AS #1
10 PRINT #1, ESC$; "J1;C"; LF$; NUL$;
20 PRINT #1, ESC$; "XD;03,p,002,022,026,031,030,0,";
30 PRINT #1, "000?<000";
40 PRINT #1, "?<???800";
50 PRINT #1, "?<???<00";
60 PRINT #1, "?=?03>00";
70 PRINT #1, "1?<00?00";
80 PRINT #1, "1?800?00";
90 PRINT #1, "1?000?80";
100 PRINT #1, "1>000380";
110 PRINT #1, "1>0003<0";
120 PRINT #1, "1<0001<0";
130 PRINT #1, "1<0001<0";
140 PRINT #1, "1>0001<0";
150 PRINT #1, "1>0001<0";
160 PRINT #1, "1<0001<0";
170 PRINT #1, "1>0001<0";
180 PRINT #1, "1>0003<0";
190 PRINT #1, "1>000380";
200 PRINT #1, "1?000?80";
210 PRINT #1, "1?800?00";
220 PRINT #1, "1?<01?00";
230 PRINT #1, "1=?07>00";
240 PRINT #1, "1<???<00";
250 PRINT #1, "1<7??800";
260 PRINT #1, "1<0?<000"
270 PRINT #1, "1<000000";
280 PRINT #1, "1<000000";
```

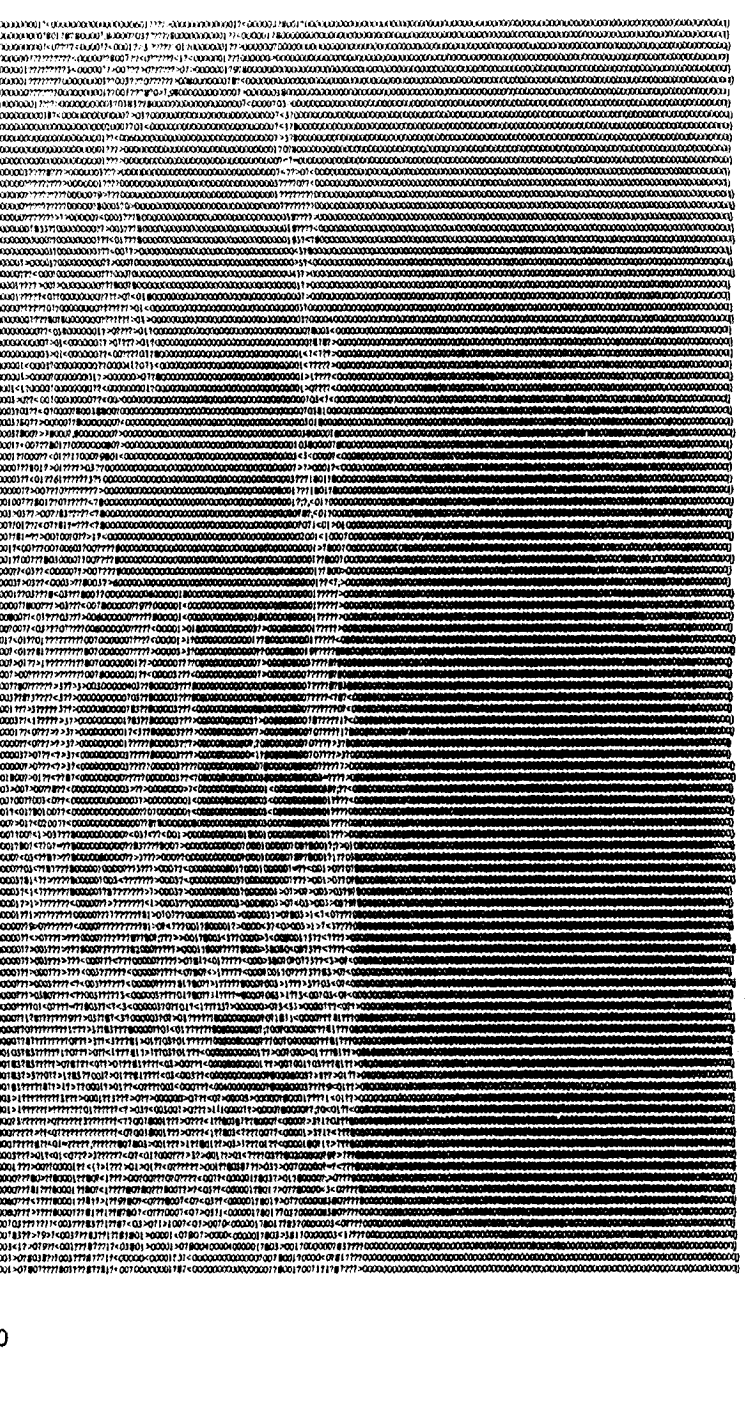
```
290 PRINT #1, "1<000000";
300 PRINT #1, "1<000000";
310 PRINT #1, "???<0000";
320 PRINT #1, "???<0000";
330 PRINT #1, "???<0000";
340 PRINT #1, LF$; NUL$;
```

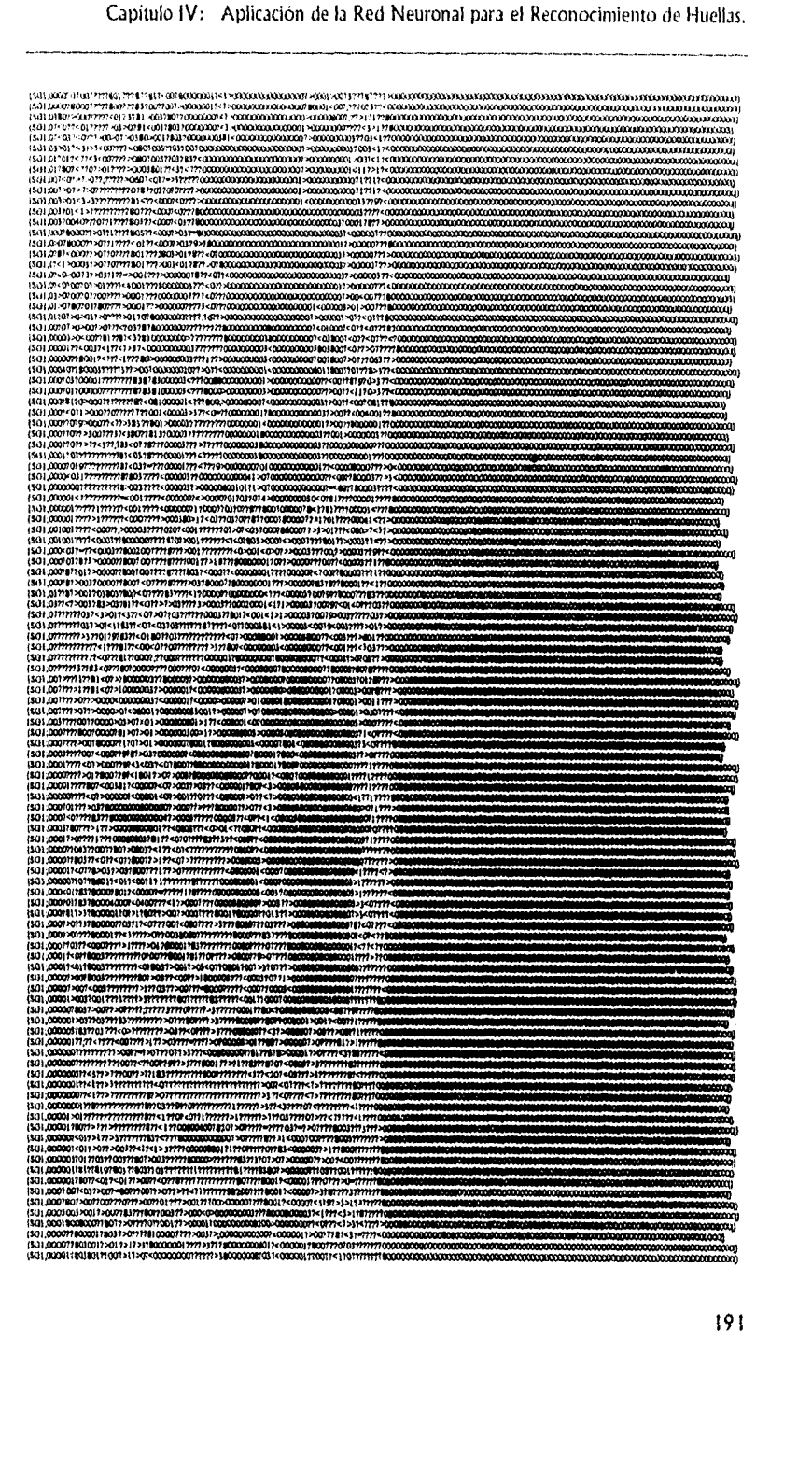
Cada dato representa un conjunto de 16 escalas de grises y utiliza una tabla de conversión la cual es:

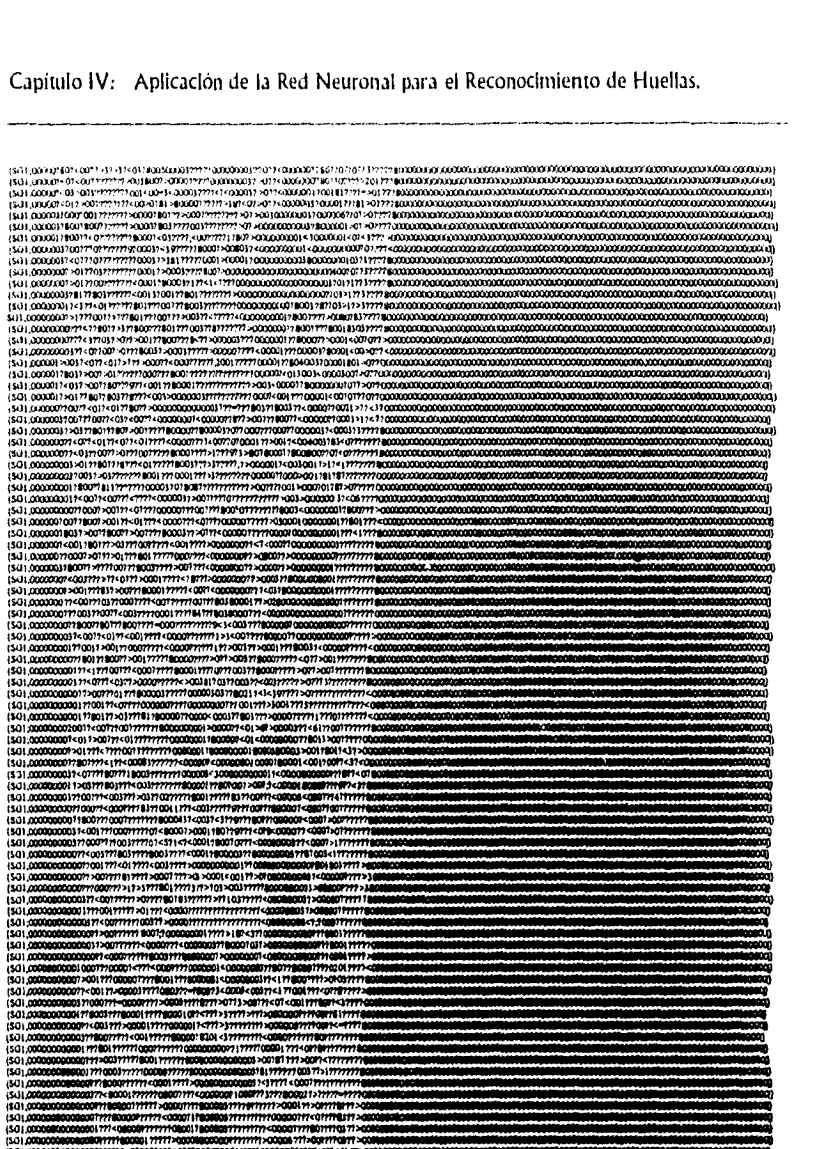
DATO	NORMAL		INVERTIDO	
	BINARIO	VALOR	BINARIO	VALOR
30H	0000	0	1111	?
31H	0001	1	1110	>
32H	0010	2	1101	=
33H	0011	3	1100	<
34H	0100	4	1011	;
35H	0101	5	1010	:
36H	0110	6	1001	9
37H	0111	7	1000	8
38H	1000	8	0111	7
39H	1001	9	0110	6
3AH	1010	:	0101	5
3BH	1011	;	0100	4
3CH	1100	<	0011	3
3DH	1101	=	0010	2
3EH	1110	>	0001	1
3FH	1111	?	0000	0

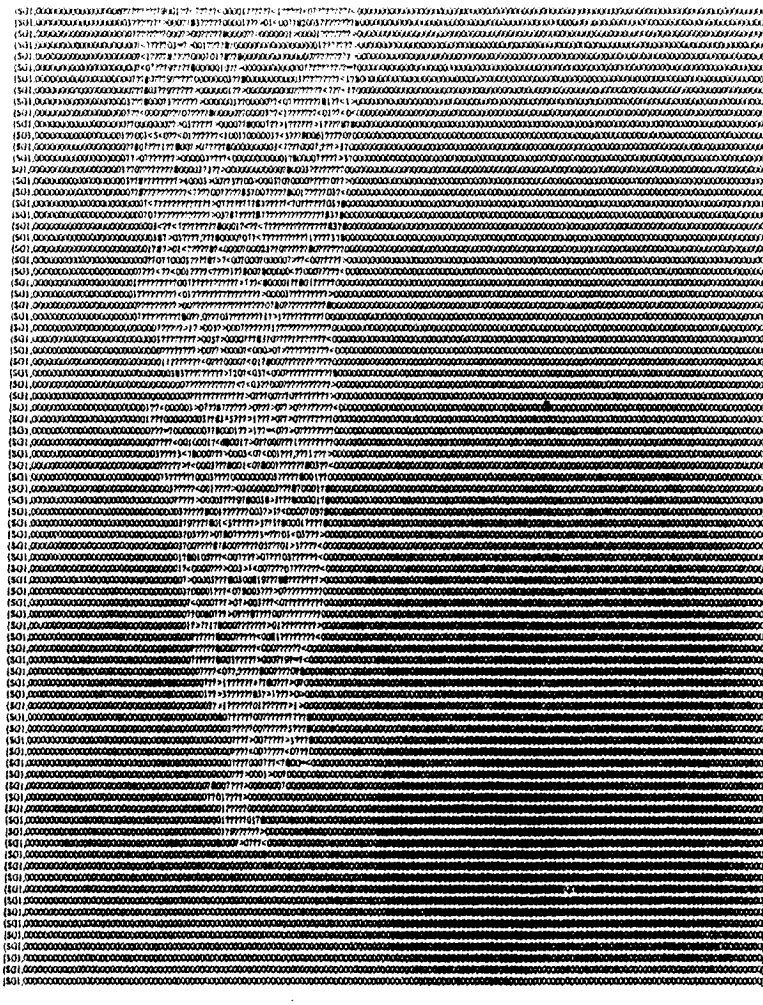
El código obtenido después de utilizar el programa convertidor para la huella fue el siguiente:











Como se puede observar, los datos obtenidos para esta huella son aproximadamente 71,162 datos (442 líneas x 161 datos por línea).

Sabemos que las huella dactilares son distintas en cada individuo, permanecen invariables durante toda la vida. Por lo tanto, por muy parecidas que sean, siempre serán diferentes.

A continuación se presentan cinco huellas dactilares digitalizadas y sus respectivos datos generados por el programa convertidor.

```
{SG0;0000,0000,99,080,000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000003000000200000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000003?>0>00600000000000000000000000000000000000000000000000000000000000000|}
{SG1;00000000003??9?83?9??;?00000000000200000000000000000000000000000000000000000000|}
{SG1;00000000003?????????????<00000000020000000000000000000000000000000000000000000000|}
{SG1;00000000003???????????????<000000020000000000000000000000000000000000000000000000|}
{SG1;0000000000007??????????????>??;?<0060003?800000000000000000000000000000000000000|}
{SG1;0000000000007???>7?????????????>01>0007??000000000000000000000000000000000000000|}
{SG1;0000000000001???<000000400011?>301><003??800000000000000000000000000000000000000|}
{SG1;0000000000007????<00000000000000000184001?<0000000000000000000000000001000000000|}
{SG1;00000000000???<18000000000000000000>00018000000000000078000000>00003>;?000000000|}
{SG1;00000000000???80000000000000000000007?00000000000000000?>00001?<0003???0000000000|}
{SG1;000000000007??000000000000000000001??>0000000000000000?>000007?0001?8?0000000000|}
{SG1;000000000001?<0000000000000000000601????>0000000000000001?<000003?8001?00000000000|}
```

Fig. 4.11. Huella No. 1

```
{SG0;0000,0000,99,080,0000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000000010000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000003<000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000003>000000000000000000000000000000000000000|}
{SG1;00000000000000000000000000000010003?>00007800000000000000000000000000000000|}
{SG1;0000000000000000000000000000003<003??0000?<000000000000000000000000000000000|}
{SG1;0000000000000000000000000000003>001??800??>00000000000000000000000000000000|}
{SG1;00000000000000000000000000000001?000??>00???00060000000000000000000000000000|}
{SG1;000000000000000000000000000000001?>003??<0???80070000000000000000000000000|}
{SG1;000000000000000000000000000000001?000??<0???800?<0000000000000000000000000|}
{SG1;000000000000000000000000000000078007?<03??>00?>000000000000000000000000000|}
{SG1;000000000000000000000000000000003?>07?<0007>007?8000000000000000000000000|}
{SG1;00000000000000000000000000000000801?>01??0003?001?<000000000000000000000000|}
{SG1;0000000000000000000000000000000<007?803?<003?<00?<01<00000000000000000000000|}
{SG1;000000000000000000000000003<00>001?<01?>003??<03>01>00000000000000000000000|}
{SG1;000000000000000000000000007>00?001??807?7<00?>03?01?00000000000000000000000|}
{SG1;000000000000000000000000007>00?<007?<018?>00??01?81?8000000000000000000000|}
{SG1;000000000000000000000000000??00?>0007?000??003?80?81?<00000000000000000000000|}
{SG1;00000000000000000000000007??801?<003?<007?<01?<03>0?<00000000000000000000000|}
```


Fig. 4.12 Huella No. 2

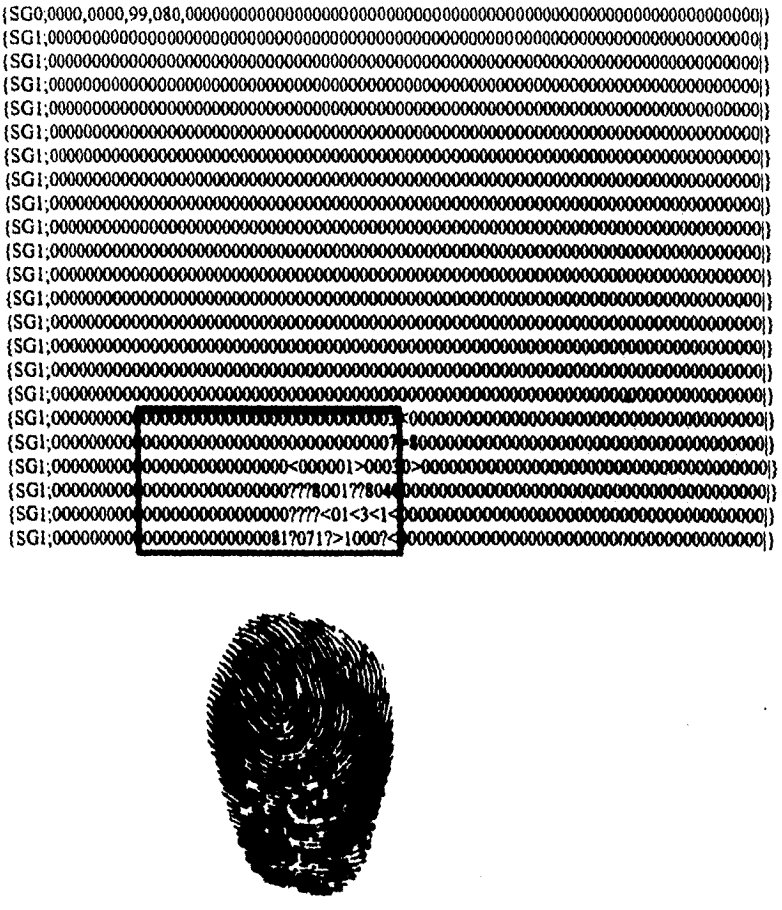


Fig. 4.13. Huella No. 3

{SG0;0000,0000,99,080,0000000000000000000000000000000000000000000000000000000000000000000000000000}
{SG1;0000000000000000000

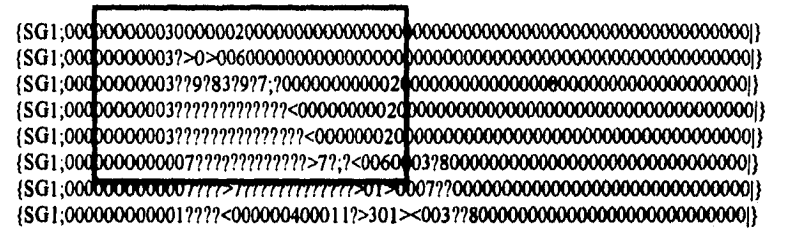
```
{SG0;0000,0000,99,080,00000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;000000000000000000000000000000000000000000000000000000000000000000000000000|}
{SG1;0000000000000000000000000007??0:00000000000000000000000000000000000000000|}
{SG1; 0000000000000000000000<0????????=??800000000000000000000000000000000000|}
{SG1; 0000000000000000000001??????????7???<0000000000000000000000000000000000|}
{SG1; 0000000000000000000001?????????81????0000000000000000000000000000000000|}
{SG1;0000000000000000000001?????????0007??000?0000000000000000000000000000000|}
{SG1; 0000000000000000000000??????????00031?>000?<000000000000000000000000000000|}
{SG1;0000000000000000000000003??1????>00000000007?<000000000000000000000000000000|}
{SG1;00000000000003>0000000000?>0????000000000000??>000000000000000000000000000000|}
{SG1;00000000000007?8000000000?>1?>???000000000000???000000000000000000000000000000|}
```

Fig. 4.15. Huella No. 5

Sabemos que las huella dactilares son distintas en cada individuo, permanecen invariables durante toda la vida. Por lo tanto por muy parecidas que sean siempre serán diferentes.

Bajo esta conceptualización, decidimos utilizar únicamente una pequeña porción de la información obtenida por el programa convertidor.

Para este caso únicamente usamos los datos correspondientes a las 7 primeras líneas con información, las cuales decidimos convertirlas en números enteros en lugar de caracteres. Para realizar esta conversión, utilizamos la tabla dad con anterioridad, quedando de la siguiente manera:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	3	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	15	14	0	14	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	15	15	9	15	8	3	15	9	15	7	11	15	0	0	0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	3	15	15	15	15	15	15	15	15	15	15	15	15	15	12	0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	3	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	12	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	7	15	15	15	15	15	15	15	15	15	15	15	15	15	14	7	15	11	15	12	0	0	6	0
0	0	0	0	0	0	0	7	15	15	15	14	7	15	15	15	15	15	15	15	15	15	15	15	15	15	14	0	1	14	0

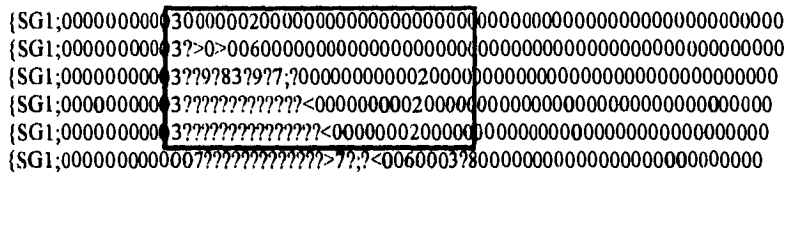
Fig. 4.16. Datos obtenidos para la huella No. 1

30x 6=210 datos

Este procedimiento se hizo de manera manual y se guardaron estos datos en un archivo de datos (datos.dat).

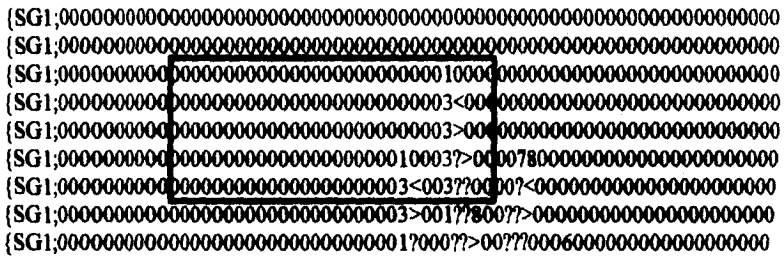
Posteriormente se realizó la conversión a binario, para poder utilizar el programa de aprendizaje, el cual se había implementado anteriormente para el reconocimiento de una letra V.

Se realizaron estas conversiones para cada una de las huella quedando de la siguiente manera:



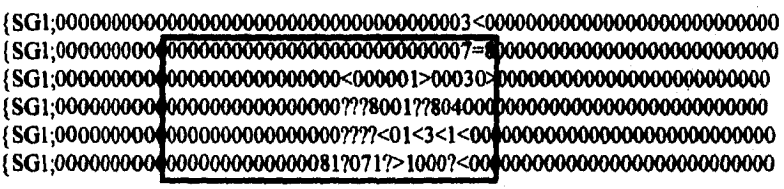
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
3	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	15	14	0	14	0	0	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	15	15	9	15	8	3	15	9	15	7	11	15	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
3	15	15	15	15	15	15	15	15	15	15	15	15	15	15	12	0	0	0	0	0	0	0	0	0	2	0	0	0	0
3	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	12	0	0	0	0	0	0	0	0	0	0	0	0
0	0	7	15	13	15	15	15	15	15	15	15	15	15	15	15	14	7	15	11	8	0	0	0	0	0	0	0	0	0

Fig. 4.17. Datos obtenidos para la huella No. 2



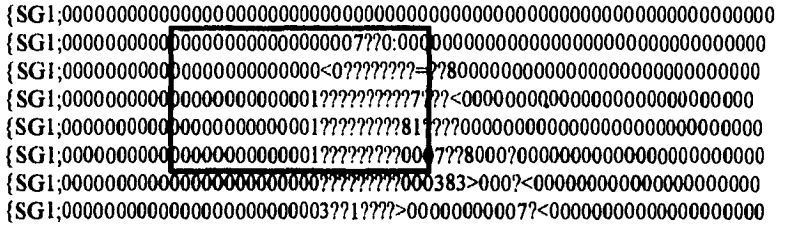
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	12	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	14	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	15	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	12	0	0	3	15	15	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	14	0	0	1	15	15	8	0	0

Fig. 4.18. Datos obtenidos para la huella No. 3



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	13	8
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	1	14	0	0	0	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	15	8	0	0	1	15	15	8	0	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	15	15	12	0	1	12	3	12	1	12	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	1	15	0	7	1	15	14	1	0	0	0	15	14	15

Fig. 4.19. Datos obtenidos para la huella No. 4

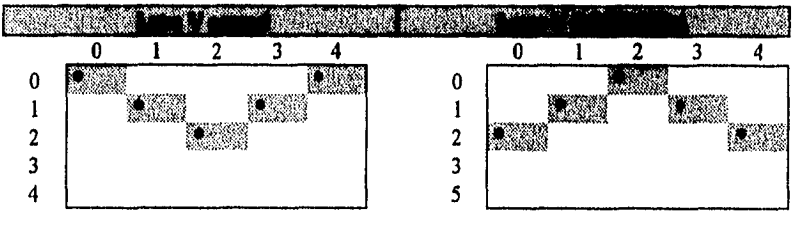


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	15	15	0	10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	15	15	15	15	15	15	15	15	10	15	15	8
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	15	15	15	15	15	15	15	15	15	15	7	15	15	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	15	15	15	15	15	15	15	15	15	8	1	15	15	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	15	15	15	15	15	15	15	15	15	0	0	0	7	15	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	15	15	15	15	15	15	15	0	0	0	3	8	

Fig. 4.20. Datos obtenidos para la huella No. 5

4.2.5. PROGRAMAS PARA LA RED NEURONAL

Antes de aplicar el programa para el reconocimiento de huellas dactilares, se capturaron dos programas para la identificación de una letra V normal e invertida, los cuales consideradan una letra representada por una matriz de 5 x 5.



Estos programas son presentados a continuación con una breve descripción de los módulos que los componen.

4.2.5.1. Programa de Aprendizaje para reconocimiento de una letra V

Este programa de aprendizaje esta diseñado baja las siguientes suposiciones:

- Dato escaneado, esta representado por una matriz de 5x5.
- La letra v puede estar normal o invertida v ó ∧.
- La salida debe ser 1 para ambos casos.
- Un error máximo de 0.005, es decir del 0.5%
- Un archivo diferente para guardar los datos de la matriz de pesos para c/u de los patrones.

A continuación se explica cada uno de los módulos que constituyen este programa.

```
/*   PROGRAMA DE APRENDIZAJE PARA RECONOCER UNA LETRA V UTILIZANDO EL
ALGORITMO DE BACKPROPAGACION */
```

Módulo de definición de librerías a utilizar, variables globales, y tamaño de los arreglos

```
#include <stdlib.h>
#include <stdio.h>  /* Incluir funciones de entrada-salida */
#include <math.h>   /* Incluir funciones matematicas  */
#include <conio.h>

/* Definición de variables globales */

#define NUMRENS 5
#define NUMCOLS 5
#define NUMPATS 2
#define DATOS 25
#define ERROR 0.005
#define COAPRE 0.25

/*  Definición del tamaño de los arreglos  */

double mapesos[NUMRENS][NUMCOLS];
double entra1[DATOS];
double entra2[DATOS];
double respues[NUMPATS];
double matriz[DATOS];
double matriz2[DATOS];
FILE *fopen(),*outfile,*salida,*infile,*infile2;
char *path="MAT1";
char *path2="MAT2";
```

```
int maxren, maxcol;
```

Módulo que visualiza en pantalla el contenido de la matriz de pesos

```
/* Impresión de los valores actuales de la matriz de pesos */

imprint_pesos()
{
int i,j,k;
clrscr();
printf("\n Matriz de pesos ");
for(i=0;i<NUMRENS;i++)
  {
  printf("\n");
  for(j=0;j<NUMCOLS;j++)
    {
    printf("%3.16f\n",mapesos[i][j]);
    }
  }
  printf("\n");
  return;
} /* fin de imprint_pesos */
```

Módulo que sirve para salvar los datos de la matriz de pesos, para cada uno de los patrones de entrada

```
salvar_pesos(pattnum)
int pattnum;
{
int i,j,k;
clrscr();
k=0;

switch(pattnum)
      {
      case 0: outfile=fopen(path,"wb"); break;
      case 1: salida=fopen(path2,"wb"); break;
      }

printf("\n Matriz de pesos ");
for(i=0;i<NUMRENS;i++)
  {
  printf("\n");
  for(j=0;j<NUMCOLS;j++)
    {
    printf("%3.16f\n",mapesos[i][j]);
    switch(pattnum)
     {
     case 0:
     matriz[k]=mapesos[i][j];
     fwrite(&matriz[k], sizeof(double), 1, outfile);
     k++;
     break;
```

```
        case 1:
        matriz2[k]=mapesos[i][j];
        fwrite(&matriz2[k], sizeof(double), 1, salida);
        k++;
        break;
        }
      }
    }
    printf("\n");
switch(pattnum)
        {
        case 0: fclose(outfile); break;
        case 1: fclose(salida); break;
        }
return;
} /* fin de salvar_pesos */
```

Módulo que lee los valores guardados en los archivos que contienen los valores de la matriz de pesos, esto como una manera de verificar de los valores se hayan guardado bien.

```
lectura_pesos(pattnum)
int pattnum;
{
int i,j,k;
clrscr();
k=0;
infile=fopen(path,"rb");
infile2=fopen(path2,"rb");
printf("\n Matriz de pesos ");
for(i=0;i<NUMRENS;i++)
  {
  printf("\n");
  for(j=0;j<NUMCOLS;j++)
    {
    printf("%3.16f\n",mapesos[i][j]);
    switch(pattnum)
     {
     case 0:
     matriz[k]=mapesos[i][j];
     fread(&matriz[k], sizeof(double), 1, infile);
     printf("dato %d = %3.15f \n",k,matriz[k]);
     mapesos[i][j]=matriz[k];
     k++;
     break;
     case 1:
     matriz2[k]=mapesos[i][j];
     fread(&matriz2[k], sizeof(double), 1, infile);
     printf("dato %d = %3.15f \n",k,matriz2[k]);
     mapesos[i][j]=matriz[k];
     k++;
     break;
     }
```

```
    }
  }
  printf("\n");
  fclose(outfile);
  fclose(salida);
  return;
} /* fin de lectura_pesos */
```

Módulo que inicializa los patrones de entrada con los valores deseados, dichos valores son visualizados en pantalla.

```
/* Inicializa los patrones de entrada con las entradas deseadas,     */
/* así como calcula las magnitudes de las entradas y las almacena    */
/* para ser usadas posteriormente por el procedimiento adjust_pes    */

init_entras()
{
int row,col;
int i,j,k;
char ent;
clrscr();

/* Inicialización del tamaño de maxren, maxcol */
maxren=NUMRENS;
maxcol=NUMCOLS;
/* primer conjunto de señales de salida a -1 */
for(row=0;row<DATOS;row++)
  {
    entra1[row]=0;
    entra2[row]=0;
  }

/* Elaboración del patrón V con los valores de entrada1 */
    entra1[0]=1.0;   /*  x  x  */
    entra1[4]=1.0;   /*  x x  */
    entra1[6]=1.0;   /*   x   */
    entra1[8]=1.0;   /* _____ */
    entra1[12]=1.0;  /* _____ */

/* Elaboración del patrón V invertida con los valores de entrada2 */
    entra2[2]=1.0;   /*   x   */
    entra2[6]=1.0;   /*  x x  */
    entra2[8]=1.0;   /*  x  x */
    entra2[10]=1.0;  /* _____ */
    entra2[14]=1.0;  /* _____ */

    printf("\n Muestra la salida del sistema \n");
    printf("\n Patrón de entrada1:");
    for(i=0;i<DATOS;i++)
    {
    printf("%8.3lf\n", entra1[i]);
    }
    printf("\n Patrón de entrada 2:");
```

```
    for (i=0;i<DATOS;i++)
     {
      printf("%8.31f\n", entra2[i]);
     }
    printf("\n Oprime ENTER para continuar.....");
    scanf("%c",&ent);
   return;
} /* fin de init_entras */
```

Módulo que inicializa la matriz de pesos con valores aleatorios

```
/* Inicialización de los pesos aleatoriamente entre -1 y +1. */
/* Obsérvese que los valores de los pesos no son */
/* necesariamente normalizados. */

init_pes()
{
int ranvalue;
unsigned int seed;
double weigth;
int row,col;
printf("\n Introduce un valor entero aleatorio:");
scanf("%d",&seed);
srand(seed);
for(row=0;row<NUMRENS;row++)
  {
  for(col=0;col<NUMCOLS;col++)
    {
    ranvalue=rand();   /* selección del no. aleatoriamente entero*/
    weigth=(16384.-(double)ranvalue)/16384.; /* para lograr que los pesos sean "+"
o "-" */
    mapesos[row][col]=weigth;
    printf("peso %d , %d = %3.31f\n",row, col, mapesos[row][col]);
    } /* fin para cada columna */
  } /* fin para cada renglón */
  return;
} /* fin de init_pes */
```

Módulo que inicializa los valores de salida deseados

```
/* Inicialización del arreglo de respuestas para */
/* proporcionar las salidas correctas */

init_respues()
{
/* colocación de respuestas utilizando un m,todo primitivo */
respues[0]=1;
respues[1]=1;
return;
} /* fin de init_respues */
```

Módulo donde se hacen los cálculos de la señal de salida, es decir :

$$net = \sum_{i=0}^{N-1} w_i x_i \qquad \text{y} \quad O = f(net), \qquad \text{donde} \quad f(net) = \frac{1}{1+e^{-net}}$$

```
/* Cálculo de la señal de salida del sistema para el patrón de entrada
"pattnum". El valor regresado tiene un valor de doble punto flotante */

double out_senal(pattnum)
int pattnum;
{
Int i,row,col;
double result;
result=0.0;
i=0;

for(row=0;row<NUMRENS;row++)
  {
  for(col=0;col<NUMCOLS;col++)
    {
    switch(pattnum)
      {
      case 0:
      result += entra1[i]*mapesos[row][col];
      i++;
      break;
      case 1:
      result += entra2[i]*mapesos[row][col];
      i++;
      break;
      } /* fin del switch */
    } /* fin para cada columna */
  } /* fin para cada renglón */
  printf("\n res = %3.16f",result);
  result=1/(1+exp(-result)); /* Cálculo de la señal de salida aplicando */
  printf("\nResultado a la salida %3.16f", result);
return(result);              /* una función de tipo sigmoidal          */
} /* fin de out_senal */
```

Módulo donde se ajustan los valores de la matriz de pesos, de acuerdo a la siguiente fórmula: peso_actual=peso_anterior+(η*error*entrada), donde n es el coeficiente de aprendizaje.

```
/* Ajuste de la matriz de pesos
w_act=w_ant+(n*Error*Input) */

double adjust_pes(patt,res)
Int patt;  /* número de patrones de entrada */
double res; /* salida actual */
{
```

```
double error,w_ant,w_act;
double in,insize;
int row,col,dat;
double coef;
coef=COAPRE;
dat=0;
printf("\n res = %3.16f",res);
printf("\no. de patrón %3.16f",respues[patt]);
error=(respues[patt]-res)*res;   /* Error=(salida ideal-salida actual)*salida actual */
printf("\n ERROR = %3.16f", error);
for(row=0;row<NUMRENS;row++)
    {
    for(col=0;col<NUMCOLS;col++)
    {
    switch(patt)
      {
      case 0:
        in=entra1[dat];
        dat++;
        break;
      case 1:
        in=entra2[dat];
        dat++;
        break;
      }
    w_ant=mapesos[row][col];
    w_act=w_ant+(coef*error*in);
    //printf("\n peso actual %3.17f",w_act);
    mapesos[row][col]=w_act;
    } /* fin para cada renglón */
    } /*fin para cada columna */
return(error);
} /* fin de adjust_pes  */
```

Programa principal

```
/* ******************* */
/*                     */
/* PROGRAMA PRINCIPAL  */
/*                     */
/* ******************* */


main ()
{
int patt, prev_patt;
int ok;
double ans, res, err, eta;
double out_senal();
int in;
int i,j,k,row,col;
clrscr();
```

Instrucciones que ejecutan las inicializaciones de pesos,entradas y salidas.

```
/* Inicialización del sistema artificial */
init_pes();
init_entras();
init_respues();
maxren=NUMRENS;
maxcol=NUMCOLS;
eta=ERROR;
```

Visualización vía pantalla de la matriz de pesos

```
/* Visualización de los pesos iniciales */
printf("\n\n PESOS INICIALES: ");
imprint_pesos();
printf("\n");
```

Instrucciones que ejecutan varios procedimientos para encontrar los valores de la matriz de pesos, para la estabilización de la red neuronal.

```
/* Respuesta de BNP para cada patrón de entrada */
for(patt=0;patt<NUMPATS;patt++)
  {
  printf("\n COMIENZA LA ADAPTACION DEL PATRÓN %d",patt);
  res=out_senal(patt);
  printf("\n LA RESPUESTA ES %8.31f y debe ser %3.15f.",res,respues[patt]);
  err=(respues[patt]-res)*res;
  printf("\n El error es de %2.7f\n",err);    /* Error=(salida ideal-salida actual)*salida
actual */
  while(err>=eta)
    {
    err=adjust_pes(patt,res);
    printf("\n\n\n AJUSTANDO LOS PESOS DEL PATRÓN %d", patt);
    res=out_senal(patt);
    printf("\n LA RESPUESTA ACTUAL ES %8.17f Y DEBE SER %3.5f.",res,
respues[patt]);
    printf("\n ERROR DE = %3.15f",err);
    } /* fin del while */
```

Al encontrar los valores de la matriz, dichos datos son almacenados y leídos nuevamente como una verificación.

```
    printf("\n EL SISTEMA RECONOCIO EL PATRÓN %d", patt);
    printf("\n PATRÓN IDENTIFICADO CORRECTAMENTE");
    printf("\n Presione cualquier tecla para continuar");
    getch();
    salvar_pesos(patt);

  } /* fin del proceso de reconocimiento de cada patrón */
```

Visualización en pantalla de datos de la matriz.

```
    /* Proceso para checar que todos los patrones fueron reconocidos */
     printf("\n\n EL SISTEMA ESTA CAPACITADO PARA RECONOCER TODOS
LOS PATRONES ");
   printf("\n Los valores finales de los pesos son: ");
     for(patt=0;patt<NUMPATS;patt++)
      {
      res=out_senal(patt);
      printf("\n\n Patrón#  Respuesta    Salida \n");
      printf("%d %3.31f  %3.15f", patt, res, respues[patt]);
      }

clrscr();
printf("\n\n Todos los patrones han sido aprendidos ....");
printf("\n\n El programa ha terminado .....");
  for(patt=0;patt<NUMPATS;patt++)
     {
     lectura_pesos(NUMPATS);
     }

retum(0);
} /* fin del programa principal */
```

4.2.5.2. Programa de reconocimiento para una letra V

El programa de reconocimiento utiliza algunos módulos que se utilizaron en el programa de aprendizaje, pero se eliminan los módulos de inicialización (de entradas, salidas y matriz de pesos) y el módulo de ajuste de pesos, pero ahora se incluye el módulo de lectura de pesos. Recordemos que en el programa de aprendizaje se utilizaron dos archivos para guardar las matrices de pesos correspondientes a los patrones de $\vee$ y $\wedge$, dichas matrices serán utilizadas en el módulo de lectura de pesos. Además ahora se han definido los valores de los patrones de entrada en arreglos de 1x25.

Es conveniente hacer mención que el programa de aprendizaje al hacer el cálculo de la segunda matriz de pesos, es decir al calcular los valores para $\wedge$, no inicializó la matriz de los peso, sino que sobre la misma matriz que se cálculo para $\vee$ se fué ajustando hasta encontrar los pesos correspondientes a este patrón.

Por lo anterior, al ocupar la primera matriz MAT1, con los patrones de entrada, solamente reconoce el patrón 3, ya que es únicamente la que contiene los datos correctos de una $\vee$. Pero si utilizamos la segunda matriz MAT2, no solo reconoce el patrón 3 como una letra V, también reconoce al primer y segundo patrón, los cuales son una V invertida y una V con un datos de más.

A continuación se mencionan los módulos que integran el programa de reconocimiento.

```
/*   RECONOCIMIENTO DE UNA LETRA V UTILIZANDO EL ALGORITMO DE
BACKPROPAGACION */

/* ESTE PROGRAMA RECONOCE UNA LETRA V NORMAL, INVERTIDA O CON
UN PEQUEÑO ERROR */
```

Módulo de definición de librerías a utilizar, variables globales, y tamaño de los arreglos

```
#include <stdlib.h>
#include <stdio.h>  /* incluir funciones de entrada-salida */
#include <math.h>   /* incluir funciones matematicas  */
#include <conio.h>

/* Definición de variables globales  */

#define NUMRENS 5
#define NUMCOLS 5
#define NUMPATS 4
#define DATOS 25
#define ERROR 0.009
#define COAPRE 0.25

/*  Definición del tamaño de los arreglos  */

double mapesos[NUMRENS][NUMCOLS];
```

Definición de los valores de los patrones de entrada

```
/* ENTRADA1=V INV.,  ENTRADA2= V CON UN ERROR, ENTRADA3=V  ,
ENTRADA4=A */

double
entra1[DATOS]={0.,0.,1.,0.,0.,0.,1.,0.,1.,0.,1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.};
double
entra2[DATOS]={0.,0.,1.,0.,0.,0.,1.,0.,1.,0.,1.,0.,0.,0.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.};
double
entra3[DATOS]={1.,0.,0.,0.,1.,0.,1.,0.,1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.};
double
entra4[DATOS]={0.,0.,1.,0.,0.,0.,1.,0.,1.,0.,1.,1.,1.,1.,1.,1.,0.,0.,0.,1.,1.,0.,0.,0.,1.};

double respues[NUMPATS];
double matriz[DATOS];
double matriz2[DATOS];
FILE *fopen(),*outfile,*salida,*infile;
```

```
char *path="MAT1";
char *path2="MAT2";
int maxren, maxcol;
```

Módulo donde se hacen los cálculos de la señal de salida, es decir :

$$net = \sum_{i=0}^{N-1} w_i x_i \qquad \text{y} \quad O = f(net), \qquad \text{donde} \quad f(net) = \frac{1}{1+e^{-net}}$$

```
/* Cálculo de la señal de salida del sistema para los patrones de entrada
El valor regresado tiene un valor de doble punto flotante */

double out_senal(pattnum)
int pattnum;
{
int i,row,col;
double result;
result=0;
i=0;

for(row=0;row<NUMRENS;row++)
  {
  for(col=0;col<NUMCOLS;col++)
    {
    switch(pattnum)
      {
      case 0:
      result += entra1[i]*mapesos[row][col];
      i++;
      break;
      case 1:
      result += entra2[i]*mapesos[row][col];
      i++;
      break;
      case 2:
      result += entra3[i]*mapesos[row][col];
      i++;
      break;
      } /* fin del switch */
    } /* fin para cada columna */
  } /* fin para cada renglón */
  result=1/(1+exp(-result));   /* Cálculo de la señal de salida aplicando */
  printf("\nResultado a la salida %3.15f", result);
return(result);              /* una función de tipo sigmoidal          */
} /* fin de out_senal */
```

Módulo que se encarga de leer los datos que se tienen almacenados en el archivo MAT2.

```
carga_pesos()
```

```
{
int i,j,k;
clrscr();
k=0;
infile=fopen(path2,"rb");
printf("Matriz de pesos\n");
for(i=0;i<NUMRENS;i++)
  {
  for(j=0;j<NUMCOLS;j++)
    {
    fread(&matriz[k],sizeof(double),1,infile);
    mapesos[i][j]=matriz[k];
    printf("peso %d,%d = %3.17f\n",i,j,matriz[k]);
    k++;
    }
  }
  printf("\n");
  fclose(infile);
  return;
} /*fin de carga_pesos */
```

Programa principal

```
/* ******************* */
/*                     */
/* PROGRAMA PRINCIPAL  */
/*                     */
/* ******************* */


main ()
{
int patt, prev_patt;
int ok;
double ans, res, err, eta;
double out_serial();
clrscr();

carga_pesos();

maxren=NUMRENS;
maxcol=NUMCOLS;
eta=ERROR;

/* Respuesta de BNP para cada patrón de entrada */
for(patt=0;patt<NUMPATS;patt++)
  {
  printf("\n COMIENZA EL RECONOCIMIENTO DEL PATRÓN %d",patt);
  res=out_serial(patt);
  err=(1-res)*res;
  printf("\n EXISTE UN ERROR DE %2.5f",err);
  if(err<=eta)
    {
   printf("\n PATRÓN %d FUE IDENTIFICADO CORRECTAMENTE COMO UNA
V\n",patt);
```

```
    }
   else
    {
   printf("\n El PATRÓN %d NO FUE RECONOCIDO\n",patt);
    }
  } /* fin del for */
  printf("\n EL SISTEMA A TERMINADO");
  return(0);
 } /* fin del programa principal */
```

Después de verificar el correcto funcionamiento de estos programas, se procedió a crear los programa que reconoce una huella digital, para ello fué necesario hacer unas modificaciones a los programas anteriores.

El programa principal esta integrado por 3 programas:

- Programa de Menú
- Programa de Aprendizaje
- Programa de Identificación

A continuación se presentan los programas anteriormente mencionados.

4.2.5.3. Menú del Programa del reconocimiento de Huellas Dactilares

/* Menú del programa que reconoce una huella dactilar digitalizada */

```
# include <stdio.h>
# include <stdlib.h>
# include <conio.h>
# include <ctype.h>
# include <math.h>  /* incluir funciones matematicas */
# include <graphics.h>
# include <alloc.h>
# include "tesis.c"
# include "tesis2.c"
# include "grafico2.c"


int main(void)
{
int i;
int op[10];
```

```
char cont[10];
char label1[60]="UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO";
char label2[60]="Campus  A R A G O N";
char label3[60]="PROGRAMA QUE RECONOCE UNA HUELLA DACTILAR
DIGITALIZADA";


int gdriver=DETECT,modograf,coderror;
initgraph(&gdriver,&modograf,"C:\\TC\\BGI");


setcolor(WHITE);
settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
outtextxy(10,15,label1);
setfillstyle(1,3);
outtextxy(160,31,label2);
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
setfillstyle(1,5);
outtextxy(125,81,label3);

setfillstyle(1,3);

bar(153,200,415,225);
outtextxy(173,211,"1-PROGRAMA DE ENTRENAMIENTO");
bar(153,239,415,264);
outtextxy(173,250,"2-PROGRAMA DE RECONOCIMIENTO");
bar(153,278,415,303);
outtextxy(173,289,"3-SALIR");
setcolor(YELLOW);
settextstyle(1,2,1);
outtextxy(10,60," AUTORES:  MARTINEZ CRUZ ESTELA     ");
outtextxy(35,85,"   JIMENEZ BARBOSA MERCEDES");
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

gotoxy(22,22);
printf("QUE OPCION DESEAS  =>  ");
   do {
    scanf("%d",op);
    switch(op[0])  {
     case 1:
         closegraph();
         printf("Programa tesis");
         aprend();
         break;
```

```
        case 2:
          closegraph();
          printf(" Programa tesis2");
          identi();
          break;
        case 3:
          break;
        }
 }while ((op[0]!=1) && (op[0]!=2) &&(op[0]!=3));
return 0;
}
```

4.2.5.4. Programa de Aprendizaje

```
/*  RECONOCIMIENTO DE UNA HUELLA DIGITALIZADA UTILIZANDO EL
ALGORITMO DE
BACKPROPAGACION */

/* PROGRAMA DE APRENDIZAJE */

/* AUTORES: MARTINEZ CRUZ ESTELA          LENGUAJE: C */
/*          MERCEDES JIMENEZ BARBOSA      SEP. 1996.  */


#include <stdlib.h>
#include <stdio.h>  /* incluir funciones de entrada-salida */
#include <math.h>   /* incluir funciones matematicas */
#include <conio.h>


#define NUMRENS 6
#define NUMCOLS 120
#define NUMPATS 1
#define DATOS 720
#define ERROR 0.000001
#define COAPRE 0.25

/*  Almacenamiento global de variables y arreglos */

double mapesos[NUMRENS][NUMCOLS];
double entra1[DATOS];
double entra2[DATOS];
```

```
double respues[NUMPATS];
double matriz[DATOS];
double matriz2[DATOS];
FILE *fopen(),*outfile,*salida,*infile,*infile2,*fp2;
char path[20];
char archi[20];
int maxren, maxcol;



/* Impresión de los valores actuales de la matriz de pesos */

imprint_pesos()
{
int i,j,k;
clrscr();
printf("\n Matriz de pesos ");
for(i=0;i<NUMRENS;i++)
  {
  printf("\n");
  for(j=0;j<NUMCOLS;j++)
    {
    printf("%3.16f\n",mapesos[i][j]);
    }
  }
  printf("\n");
  return;
} /* fin de imprint_pesos */

salvar_pesos()
{
int i,j,k;
char file_name[20];
clrscr();
k=0;

 printf("\n Nombre del Archivo para guardar la matriz de pesos ");
  scanf("%s",&path);

  strcpy(file_name,path);
  strcat(file_name,".dat");

  if((outfile=fopen(file_name,"wb"))==NULL)
  {
```

```
    perror("\n El archivo no se puede abrir ");
    exit(0);
    }

printf("\n Matriz de pesos ");
for(i=0;i<NUMRENS;i++)
  {
  printf("\n");
  for(j=0;j<NUMCOLS;j++)
    {
    printf("%3.16f\n",mapesos[i][j]);
          matriz[k]=mapesos[i][j];
          fwrite(&matriz[k], sizeof(double), 1, outfile);
          k++;
    }
  }
  printf("\n");
  fclose(outfile);
  return 0;
} /* fin de salvar_pesos */

/* Inicializa los patrones de entrada con las entradas deseadas,     */
/* así como calcula las magnitudes de las entradas y las almacena    */
/* para ser usadas posteriormente por el procedimiento adjust_pes    */

init_entras()
{
int row,col;
int i,j,k,c;
char ent, file_name[20];
float n;
clrscr();

/* Inicialización del tamaño de maxren, maxcol */
maxren=NUMRENS;
maxcol=NUMCOLS;
/* primer conjunto de señales de salida a -1 */
for(row=0;row<DATOS;row++)
  {
    entra1[row]=0;
  }

/* lectura del archivo de datos del patrón */
```

```
            i++;
            break;
            case 1:
            result += entra2[i]*mapesos[row][col];
            i++;
            break;
            } /* fin del switch */
    } /* fin para cada columna */
  } /* fin para cada renglón */
  printf("\n res = %3.16f",result);
  result=1/(1+exp(-result)); /* Cálculo de la señal de salida aplicando */
  printf("\nResultado a la salida %3.16f", result);
return(result);              /* una función de tipo sigmoidal          */
} /* fin de out_senal */


/* Ajuste de la matriz de pesos
w_act=w_ant+(n*Error*Input) */

double adjust_pes(patt,res)
int patt;  /* número de patrones de entrada */
double res; /* salida actual */
{
double error,w_ant,w_act;
double in,insize;
int row,col,dat;
double coef;
coef=COAPRE;
dat=0;
printf("\n res = %3.16f",res);
printf("\no. de patrón %3.16f",respues[patt]);
error=(respues[patt]-res)*res;   /* Error=(salida ideal-salida actual)*salida actual */
printf("\n ERROR = %3.16f", error);
for(row=0;row<NUMRENS;row++)
    {
    for(col=0;col<NUMCOLS;col++)
    {
    switch(patt)
          {
          case 0:
            in=entra1[dat];
            dat++;
            break;
          case 1:
```

```
  for(col=0;col<NUMCOLS;col++)
    {
    ranvalue=rand();   /* selección del no. aleatoriamente entero*/
    weigth=(16384.-(double)ranvalue)/16384.; /* para lograr que los pesos sean "+" o "-"
*/
    mapesos[row][col]=weigth;
    printf("peso %d , %d = %3.31f\n",row, col, mapesos[row][col]);
    } /* fin para cada columna */
  } /* fin para cada renglón */
  return;
} /* fin de init_pes */


/* Inicialización del arreglo de respuestas para */
/* proporcionar las salidas correctas */

init_respues()
{
/* colocación de respuestas utilizando un m,todo primitivo */
respues[0]=1;
respues[1]=-1;
return;
} /* fin de init_respues */


/* Cálculo de la señal de salida del sistema para el patrón de entrada
"pattnum". El valor regresado tiene un valor de doble punto flotante */

double out_senal(pattnum)
int pattnum;
{
int i,row,col;
double result;
result=0.0;
i=0;

for(row=0;row<NUMRENS;row++)
  {
  for(col=0;col<NUMCOLS;col++)
    {
    switch(pattnum)
          {
          case 0:
          result += entra1[i]*mapesos[row][col];
```

```
  printf("\n Nombre del Archivo de Datos [ patrón 1] ");
  scanf("%s",&archi);

  strcpy(file_name,archi);
  strcat(file_name,".bin");

  if((fp2=fopen(file_name,"r"))==NULL)
  {
   perror("\n El archivo no se puede abrir ");
   exit(0);
   }

   printf("\n Patrón de entrada 1:");
   for (i=0;i<DATOS; i++)
    {
        fscanf (fp2,"%f", &n);
      entra1[i]=n;
        printf("\n Dato %d %3.5f\n ",i,entra1[i]);
    }
 if ((c=fclose(fp2)) !=0)
  printf("\n El archivo no puede ser cerrado %d ", c);
  printf("\n Oprime ENTER para continuar.....");
   scanf("%c",&ent);
   return;
} /* fin de init_entras */


/* Inicialización de los pesos aleatoriamente entre -1 y +1. */
/* Obsérvese que los valores de los pesos no son */
/* necesariamente normalizados. */

init_pes()
{
int ranvalue;
unsigned int seed;
double weigth;
int row,col;

//printf("\n Introduce un valor entero aleatorio:");
//scanf("%d",&seed);
seed=156;
srand(seed);
for(row=0;row<NUMRENS;row++)
  {
```

```
            in=entra2[dat];
            dat++;
            break;
          }
    w_ant=mapesos[row][col];
    w_act=w_ant+(coef*error*in);
    //printf("\n peso actual %3.17f",w_act);
    mapesos[row][col]=w_act;
    } /* fin para cada renglón */
    } /*fin para cada columna */
return(error);
} /* fin de adjust_pes */


/* ****************** */
/*                    */
/* PROGRAMA PRINCIPAL */
/*                    */
/* ****************** */


aprend()
{
int patt, prev_patt;
int ok;
double ans, res, err, eta;
double out_senal();
int in;
int i,j,k,row,col;
clrscr();
```

```
/* Inicialización del sistema artificial */
init_pes();
init_entras();
init_respues();
maxren=NUMRENS;
maxcol=NUMCOLS;
eta=ERROR;


/* Visualización de los pesos iniciales*/
printf("\n\n PESOS INICIALES: ");
imprint_pesos();
printf("\n");

/* Respuesta de BNP para cada patrón de entrada */
for(patt=0;patt<NUMPATS;patt++)
  {
  if (patt==1) init_pes();
  printf("\n COMIENZA LA ADAPTACION DEL PATRÓN %d",patt);
  res=out_senal(patt);
  printf("\n LA RESPUESTA ES %8.31f y debe ser %3.15f.",res,respues[patt]);
  err=(respues[patt]-res)*res;
   printf("\n El error es de %2.7f\n",err);     /* Error=(salida ideal-salida actual)*salida
actual */
  while(err>=eta)
        {
        err=adjust_pes(patt,res);
        printf("\n\n\n AJUSTANDO LOS PESOS DEL PATRÓN %d", patt);
        res=out_senal(patt);
        printf("\n LA RESPUESTA ACTUAL ES %8.17f Y DEBE SER %3.5f.",res,
respues[patt]);
        printf("\n ERROR DE = %3.15f",err);
        } /* fin del while */

   printf("\n EL SISTEMA RECONOCIO EL PATRÓN %d", patt);
   printf("\n PATRÓN IDENTIFICADO CORRECTAMENTE");
   printf("\n Presione cualquier tecla para continuar");
   getch();
   salvar_pesos();

 } /* fin del proceso de reconocimiento de cada patrón */

  /* Proceso para checar que todos los patrones fueron reconocidos */
```

```
  printf("\n\n EL SISTEMA ESTA CAPACITADO PARA RECONOCER TODOS LOS
PATRONES ");
  printf("\n Los valores finales de los pesos son: ");
    for(patt=0;patt<NUMPATS;patt++)
     {
     res=out_senal(patt);
     printf("\n\n Patrón#  Respuesta    Salida \n");
     printf("%d %3.31f  %3.15f", patt, res, respues[patt]);
     }

clrscr();
printf("\n\n Todos los patrones han sido aprendidos ....");
printf("\n\n El programa ha terminado .....");
return(0);
} /* fin del programa principal */
```

4.2.5.5. Programa de Identificación

```
/*      RECONOCIMIENTO DE UNA HUELLA DIGITAL UTILIZANDO EL
ALGORITMO DE
BACKPROPAGACION */

/* ESTE PROGRAMA DETECTA UNA LETRA HUELLA */


/* AUTORES: MARTINEZ CRUZ ESTELA          LENGUAJE: C */
/*       JIMENEZ BARBOSA MERCEDES       SEP. 1996. */


#include <stdlib.h>
#include <stdio.h>  /* incluir funciones de entrada-salida */
#include <math.h>   /* incluir funciones matematicas */
#include <conio.h>


#define NUMRENS 6
#define NUMCOLS 120
#define NUMPATS 1
#define DATOS 720
#define ERROR 0.000001
#define COAPRE 0.25

/*  Almacenamiento global de variables y arreglos */
```

```
double mapesos[NUMRENS][NUMCOLS];

float entra[DATOS];
double respues[NUMPATS];
double matriz[DATOS];
FILE *fopen(),*outfile,*salida,*infile,*fp2;
char path3[20];
char archi[20];
int maxren, maxcol;



/* Cálculo de la señal de salida del sistema para los patrones de entrada
El valor regresado tiene un valor de doble punto flotante */

double senal(pattnum)
int pattnum;
{
int i,row,col;
double result;
result=0;
i=0;

for(row=0;row<NUMRENS;row++)
  {
  for(col=0;col<NUMCOLS;col++)
    {
    switch(pattnum)
          {
          case 0:
          result += entra[i]*mapesos[row][col];
          i++;
          break;
          case 1:
          result += entra2[i]*mapesos[row][col];
          i++;
          break;
          } /* fin del switch */
    } /* fin para cada columna */
  } /* fin para cada renglón */
  result=1/(1+exp(-result));  /* Cálculo de la señal de salida aplicando */
  printf("\nResultado a la salida %3.15f", result);
return(result);              /* una función de tipo sigmoidal          */
```

```
} /* fin de out_senal */


carga_pesos()
{
int i,j,k,c;
char file_name[20];
clrscr();
k=0;
  printf("\n Nombre del Archivo en donde esta la matriz de pesos ");
  scanf("%s",&path3);

  strcpy(file_name,path3);
  strcat(file_name,".dat");
infile=fopen(file_name,"rb");
printf("\n  Matriz de pesos cargada en Memoria \n");
for(i=0;i<NUMRENS;i++)
  {
  for(j=0;j<NUMCOLS;j++)
    {
    fread(&matriz[k],sizeof(double),1,infile);
    mapesos[i][j]=matriz[k];
    printf("%3.5f",mapesos[i][j]);
    k++;
    }
  }
  fclose(infile);
  return;
} /*fin de carga_pesos */


binario()
{
 int i,j,c;
 char file_name[20];
 float n;

 printf("\n Nombre del Archivo en donde están los Datos ");
 printf("\n del Patrón que va ser identificado =>  ");
 scanf("%s",&archi);

 strcpy(file_name,archi);
 strcat(file_name,".bin");
```

```
  if((fp2=fopen(file_name,"r"))==NULL)
  {
   perror("\n El archivo no se puede abrir ");
   exit(0);
   }

   for (i=0;i<DATOS;i++)
    {
        fscanf (fp2,"%f", &entra[i]);
    }
 printf("\n Datos del Patrón cargados en memoria\n ");
 if ((c=fclose(fp2)) !=0)
  printf("\n El archivo no puede ser cerrado %d ", c);
return 0;
}


/* ******************* */
/*                     */
/* PROGRAMA PRINCIPAL  */
/*                     */
/* ******************* */


identi ()
{
int patt, prev_patt;
int ok;
double ans, res, err, eta;
double senal();
clrscr();

carga_pesos();
binario();
//carga_datos();
maxren=NUMRENS;
maxcol=NUMCOLS;
eta=ERROR;

/* Respuesta de BNP para cada patrón de entrada */
for(patt=0;patt<NUMPATS;patt++)
  {
  printf("\n COMIENZA EL RECONOCIMIENTO DEL PATRÓN %d",patt);
  res=senal(patt);
```

```
  err=(1-res)*res;
  printf("\n EXISTE UN ERROR DE %2.5f",err);
  if(err<=eta)
        {
  printf("\n PATRÓN %d FUE IDENTIFICADO CORRECTAMENTE ", patt);
  printf("\n Y CORRESPONDE A LA SIGUIENTE PERSONA ");
  gra();
        }
  else
        {
  printf("\n El PATRÓN %d NO FUE RECONOCIDO\n\n\n",patt);
        }
 } /* fin del for */
 printf("\n EL SISTEMA A TERMINADO\n");
 printf("\n\n\n Presione cualquier tecla para continuar");
 getche();
 return(0);
} /* fin del programa principal */
```

Además de estos programas se tuvieron que realizar dos programas más, uno para convertir los datos hexadecimales a binarios y otro para la aplicación cuando reconoce la huella digital.

```
/* PROGRAMA CONVERTIDOR DE DATOS ENTEROS A BINARIOS */


/* AUTORES: MARTINEZ CRUZ ESTELA                LENGUAJE C */
/*        MERCEDES JIMENEZ BARBOSA             SEP. 1996. */


#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <string.h>



#define DATOS 8              /* max no. datos */


char archi[20];
float d[DATOS];
FILE *fp1, *fp2, *fp3, *fopen();
```

```
entrada()
{
 int i, con;
 con=DATOS;
 for(i=0;i<con;i++)
 {
 printf("\n Dato %d   ",i);
 scanf("%f",&d[i]);
 }
 return 0;
 }


lectura()
{
  int i,j,c;
  char file_name[20];

  printf("\n Nombre del Archivo de No. Enteros ");
  scanf("%s",&archi);

  strcpy(file_name,archi);
  strcat(file_name,".dat");

  if((fp2=fopen(file_name,"r"))==NULL)
  {
    perror("\n El archivo no se puede abrir ");
    exit(0);
    }

   for (i=0;i<DATOS; i++) {
         fscanf (fp2,"%f", &d[i]);
         printf("\n Dato %d  %3.5f ",i,d[i]);
    }
 if ((c=fclose(fp2)) !=0)
   printf("\n El archivo no puede ser cerrado %d ", c);
return 0;
}

binario()
{
```

```
  int i,j,c;
  char file_name[20];
  float n;

  printf("\n Nombre del Archivo de No. Binarios ");
  scanf("%s",&archi);

  strcpy(file_name,archi);
  strcat(file_name,".bin");

  if((fp2=fopen(file_name,"r"))==NULL)
  {
   perror("\n El archivo no se puede abrir ");
   exit(0);
   }

   for (i=0;i<DATOS; i++)
    {
        printf("\n Dato %d  ",i);
        for(j=0;j<4;j++)
        {
        fscanf(fp2,"%f", &n);
        printf(" %3.5f\n ",n);
        }
    }
 if ((c=fclose(fp2)) !=0)
  printf("\n El archivo no puede ser cerrado %d ", c);
return 0;
}



 guardar()
 {

 int  i, j, c, k;
 char file_name[20];

 printf("\n Nombre del archivo para guardar ");
 scanf("%s",&archi);

 strcpy(file_name,archi);
 strcat(file_name, ".dat");
```

```
 if (( fp2 =fopen(file_name,"w+")) ==NULL)
 {
   perror("El archivo no puede abrirse");
   exit(0);
 }

k=0;
for (i=0; i<DATOS; i++)
   {
       if(k==30) {
         k=0;
         fprintf(fp2,"\n");
       }
       fprintf(fp2, "%3.5f", d[i]);
       fprintf(fp2,"\n");
       k++;
       }
       if ((c=fclose(fp2)) !=0)
          printf("\nEl archivo no puede cerrarse %d ", c);
       return 0;
       }

escribe()
{

int  i, j, c;
float k,n;
char file_name[20];

printf("\n Nombre del archivo para guardar ");
scanf("%s",&archi);

strcpy(file_name,archi);
strcat(file_name, ".bin");

if (( fp2 =fopen(file_name,"w+")) ==NULL)
{
  perror("El archivo no puede abrirse");
  exit(0);
}

k=0.0;
n=1.0;
```

```
 for (i=0; i<DATOS; i++)
    {
       if (d[i]==15.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,n,n,n);
       if (d[i]==14.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,n,n,k);
       if (d[i]==13.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,n,k,n);
       if (d[i]==12.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,n,k,k);
       if (d[i]==11.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,k,n,n);
       if (d[i]==10.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,k,n,k);
       if (d[i]==9.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,k,k,n);
       if (d[i]==8.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", n,k,k,k);
       if (d[i]==7.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,n,n,n);
       if (d[i]==6.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,n,n,k);
       if (d[i]==5.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,n,k,n);
       if (d[i]==4.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,n,k,k);
       if (d[i]==3.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,k,n,n);
       if (d[i]==2.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,k,n,k);
       if (d[i]==1.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,k,k,n);
       if (d[i]==0.0) fprintf(fp2, "%3.5f\n%3.5f\n%3.5f\n%3.5f\n", k,k,k,k);
       }
       if ((c=fclose(fp2)) !=0)
          printf("\nEl archivo no puede cerrarse %d ", c);
       return 0;
       }


/***************** MAIN ********************/

main()
{
  char select[20], cont[10];
  strcpy(archi, "**********");

  do {
   printf("\n      Escoger OPCION ");
   printf("\n  G .... Guardar archivo en binario");
   printf("\n  B .... Lectura de archivo binario ");
   printf("\n  W .... Guardar archivo de No. Enteros");
   printf("\n  L .... Lectura de archivo de No. Enteros");
   printf("\n  E .... Entrada de datos de No. Enteros\n");
   do {
    scanf("%s",select);

    switch(select[0]) {
       case 'w':
```

```
        case 'W':
          guardar();
          break;
        case 'l':
        case 'L':
          lectura();
          break;
        case 'e':
        case 'E':
          entrada();
          break;
        case 'b':
        case 'B':
          binario();
          break;
        case 'g':
        case 'Q':
          escribe();
        break;
        default:
          break;
        }
 }while ((select[0]!='g') && (select[0]!='G') &&(select[0]!='l') && (select[0]!='L') &&
(select[0]!='b') && (select[0]!='B')  && (select[0]!='w') && (select[0]!='W') &&
(select[0]!='w') && (select[0]!='W'));
 printf("\n Desea continuar  [S/N]  ? ");
 scanf("%s",cont);
} while((cont[0]=='s') || (cont[0]=='S'));
 printf("\nEste es el final. ");
 printf("\n Adiós ");
return 0;
}
```

CONCLUSIONES

De acuerdo con los objetivos del presente trabajo se puede observar la importancia que tienen hoy en día el estudio de los sistemas artificiales, especialmente las redes neuronales. Al mismo tiempo se plantearon las bases para el diseño e implementación de nuevos sistemas con redes neuronales aplicando diferentes procedimientos y algoritmos.

En este trabajo nos enfocamos a una aplicación, que fue el reconocimiento de huellas, pero existen cientos o miles de aplicaciones. La selección de la aplicación dependerá únicamente de personas con una gran inquietud de estudiar y diseñar sistemas novedosos, basados en sistemas inteligentes.

Es importante recalcar que el uso de estos sistemas tienen actualmente gran auge en los países desarrollados, por lo que no podemos quedarnos atrás en cuanto a su estudio e investigación, ya que presenta grandes expectativas a lo futuro, debido a la simplificación de procedimientos engorrosos y reducción de tiempo y costos.

Las aplicaciones que tienen las redes neuronales son muy amplias, como ya se presento, además realizan actividades complejas en forma rápida. Lo anterior no quiere decir que vayan a sustituir al hombre, sino que podrán hacer su vida más cómoda y sin riesgos (procesos de control), además debido a que se basan en cálculos matemáticos para su operación, resultan de gran confiabilidad.

Los resultados de este trabajo fueron satisfactorios aunque se pueden mejorar algunos aspectos de presentación, pero este trabajo estuvo enfocado al uso de los principios del algoritmo de backpropagación y establecer las bases para aplicaciones más sofisticadas en el reconocimiento de patrones.

BIBLIOGRAFIA

A. E. Rosenberg, *"Automatic Speaker Verification: A Review,"* Proc. *IEEE*, Vol. 64, No. 4, pp. 475-487, April 1976.

B.S. Atal, *"Automatic Recognition of Speakers from Their Voices,"* Proc. IEEE, Vol. 64, No. 4, pp. 460-475, April 1976.

Branko Soucek and the Iris Group, *"Neural Intelligent Systems Integration"*, De. Wiñey-Interscience Publication, USA 1988.

D. R. Reddy, *"Speech Recognition by Machine: A Review,"* Proc. IEEE, Vol. 64, No. 4, pp. 501-531, April 1976.

Freeman, James A. and Skapura, David M. *"Neural Networks: Algorithms, Applications, and Programming Techniques"*, Addison-wesley Publishing Company, pp. 89-126, New York, 1991.

K. Yamamoto, H. Yamada, T. Saito, and R.I. Oka, *Recognition of handprinted Chinese Characters and Japanese cursive syllabary*, Proc. Intl. Conf. Pattern Recognition, . July-August 1984, p.p. 385-388 (1984).

Martin,Thomas B. *"Automatic Speech & Speaker Recognition", Edited by Dixon N. Rex, The Institute of Electrical and Electronics Engineers, Inc. New York IEEE-Press , pp. 87-91, USA 1979.*

Patrick K. Simpson *"Foundations of Neural Networks"*

Rabiner, L.R. and Schafer R.W. *"Digital Processing of Speech Signals"*, Prentice-Hall, pp. 5-9 y 462-502, USA 1978.

Schwab, Eileen. *Pattern Recognition by human and machines-Visual Perception, Speech Perception.* v1.

Wilson, G.V. and G.S. Pawley, 1988. On the stability of the travelling salesman problem algorithm of Hopfield and Tank, *Biological Cybernetics*, Vol. 58, pp. 63-70

Winston, P.H., 1984. *Artificial Intelligence*, 2nd Ed., Addison-Wesley, Reading, MA.

Young Tzay Y.,*Handbook of Pattern Recognition and Image Processing*. De Academic Press, Inc.,San Diego.1986.

10th. International Conference on Pattern Recognition. Ed. IEEE, IAPR.

REFERENCIAS BIBLIOGRAFICAS

A. Chapanis, *"Interactive Human Communication,"* Scientific American, Vol. 232 (3), pp. 36-42, 1975.

A. E. Rosenberg and M. R. Sambur, *"New Techniques for Automatic Speaker Verification," IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-23, pp. 169-176, 1975.

Anderson, J.J. Silverstein, S. Ritz, and R. Jones. *"Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model".* Physch. Rev., Vol. 84, 1977.

B. Gold, *"Hopfield Model Applied to Vowel and Consonant discrimination",* MIT Lincoln Laboratory Technical Report, TRT-747, ADA169742, June 1986.

B. Widrow and M.E. Hoff. Jr. *"Adaptive Switching Circuits".* IRE Western Electric Show and Convention Record, Part Iv, 1960.

B.Widrow and M.E. Hoff. Jr. *"Adaptive Switching Circuits".* IRE Western Electric Show and Convention Record, Part IV, 1960.

B.Widrow. *"Generalization and Information Storage in Networks of ADALINE Neurons".* Eds. Washington, D.C.: Spartan Books, 1962.

C. H. Coker, *"A Model of Articulatory Dynamics and Control,"* Proc. IEEE, Vol. 64, No. 4, pp. 452-460, April 1976.

Y. Suen , *Computer Recognition of Kanji characters*, Proc., Intl. Conf. On Text Processing with a Large Character Set, Oct. 1983, p.p. 429-435 (1983)

D. E. Rumelhart, and J. L. McClelland, *"Parallel Distributed Processing: Explorations in the Microstructure of Cognition",* MIT Press, 1986.

D. E. Rumelhart, G. E. Hinton and R. J. Williams, *"Learning Iniernal Representation by Error Propagation"* in D.E. Rumelhart & J. L. McClelland (Eds.) Parallel

Distributed Processing: Explorations in the Microstructure of Recognition, Vol. 1: Foundations. MIT Press, 1986.

D.B. Parker. *"Learning Logic Technical Report TR-47, Center for Computational Research in Economics and Management Science"*, MIT, Cambridge, MA, April 1985.

D.E. Rumelhart and J.L. McClelland. *"Learning Internal Representations by Error propagation"*. In Parallel Distributed Processing. vol. 1. ch. VIII. Eds. Cambridge, Ma:M.I.T. Press, 1986.

D.J. Wallace, *"Memory and Learning in a Class of Neural Models"*, in B. Bunk and K. H. Mutter (Eds.) Proceeding of the Workshop on Lattice Gauge Theory, Wuppertal, 1985, Plenum 1986.

D.R. Reddy, *"Speech Recognition by Machine: A Review,"* to be published in the Proc. IEEE, April 1976.

DARPA, *"Neural Networks Study,"* (Fairfax, VA:AFCEA International Press, 1988), p. XXVIII:

David A. Brown, *"Neural Research May Benfit AI,"* IEEE Software, vol. 5, no. 6 (Noviembre 1988):88.

David S. Touretzky. *"Aduanced in Neural Information Process ing System"* Vol. II.

Drew Van Camp. *"The amateur Scientist"*. Scientific Americanan pag. 125.

F. L. Lewis, *"Optimal Estimation"*, John Wiley & Sons, New York 1986.

F.H. Glanz *"Statiscal Extrapolation in Certain Adaptive Pattern-Recognition Systems"*. Ph. D. Thesis, Tech. Rep., Stanford Electron, Labs., Stanford, CA., May, 1965.

F.Rosenblatt.*"Principles of Neurodynamics: Perceptrons and the Theory of brain Mechanism"*. Washington, D.C.: Spartan Books, 1962.

Fukushima, K. *"Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition"*. Neural Networks, Vol. 1, 1988.

Grossberg, S. *"Studies of Mind and Brain"*. Boston: Reidel, 1982.

H. Dudley, *"Remaking Speech,"* J. Acoust. Soc. Am., Vol. 11, pp. 169-177, 1939.

H. Levitt, *"Speech Processing Aids for the Deaf: An Overview,"* IEEE Trans. on Audio and Electroacoustics, Vol. AU-21, pp. 269-273, June 1973.

H. Nyquist, *"Certain Topics in Telegraph Transmission Theory,"* Trans. AIEE, Vol. 47, pp. 617-644, February 1928.

Hopfield J. and Tank. *"Neural Computation of Decision in Optimization Problems".* Biol. Cybernet., Vol. 52. 1985.

J. L. Flanagan, *"Computers That Talk and Listen: Man-Machine Communication by Voice,"* Proc. IEEE, Vol. 64, No. 4, pp. 416-432, April 1976.

J. Makhoul, *"Spectral Analysis of Speech by Linear Prediction,"* IEEE Trans. Audio Electroacoustics, Vol. Au-21 (3), pp. 140-148, 1973.

J. S. Denker, AIP Conference Proceedings 151, *Neural Networks for Computing*, Snowbird Utah, AIP, 1986.

J.J. Hopfield and D.W. Tank, *"Computing with Neural Circuits: A Model"*, Science, Vol. 233, 625-633, August 1986.

J.J. Hopfield, *"Neural Networks and Phisycal Systems with Emergent Collective Computational Abilities"*, Proc. Natl. Acad. Sci. USA Vol. 79, 2554-2558, April 1982.

J.J. Hopfield, *"Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons"*, Proc. Natl. Acad. Sci. USA, Vol. 91, 3088-3092, May 1984.

J.Schwartz(Presidente de Schwartz Asociados, Vista Montaña, CA) *"Neural Networks: Capabilities and Applications,"* IEEE Videoconferencias, Septiembre 27, 1989.

Jacek M. Zurada. *"Introduction to Artificial Neural System"*. West Publishing Company. USA, 1992.

James McClelland and David Rumelhart. *"Parallel Distributed Processing"*, volumes 1 and 2. MIT Press, Cambridge. Ma, 1986.

Kohonen, T. *"Learning Vector Quantization for Pattern Recognition"*. Helsinki University of Tecnology, 1986.

L. H. Rosenthal, L. R. Rabiner, R. W. Schafer, P. Cummiskey, and J.L. Flanagan, *"A multiline Computer Voice Response System Utilizing ADPCM Coded Speech"*, *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-22, No. 5, pp. 339-352, October 1974.

L.R. Rabiner and R. W. Schafer, *"Digital Techniques for ComputerVoice Response: Implementations and Applications"*, *Proc. IEEE*, Vol. 64, No. 4, pp. 416-433, April 1976.

M.E. Hoff, Jr. *"Learning Phenomena in Networks of Adaptive Switching Circuits"*. Ph D. Thesis, Techs. Rep., Stanford Electron Labs., Stanford. C.A. May., 1965.

M.R. Schroeder, *"Models of Hearing,"* Proc. IEEE, Vol. 63 (9), pp. 1322-1352, September 1976.

P. Werbos. *"Beyord Regression: New Tools for Prediction and Analysis in the Behavioral Sciences"*. PhD Thesis, Harvard, Cambridge, MA, August 1974.

R.O. Duda and P.E. Hart, *"Pattern Classification and Scene Analysis"*, John Wiley & Sons, Ney York 1973.

R.P Lippmann, B.Gold, and M.L. Malpass, *"A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification"*, MIT Lincoln Laboratory Technical Report, TR-769, to be published.

Rosemblatt, F. *"Principles of Neurodynamics"*. Washington, D.C.: Spartan Books, 1962.

T. Kohonen, *"Self-Organization and Associative Memory"*, Springer-Verlag, Berlin 1984.

T. Parsons, *"Voice and Speech Processing"*, Mc Graw Hill, New York 1986.

T. Sejnowski and C. R. Rosenberg, *"NETtalk: a Parallel Network That Learns to Read Aloud"*, John Hopkins Univ. Technical Report JHU/EECS-8601, 1986.

T.B. Martin, *"Applications of Limited Vocabulary Recognition Systems,"* in Speech Recognition: Invited Papers of the IEEE Symposium (D.R. Reddy, ed.), Academic Press, New York, 1976.

W.C. Ridgway III, *"An Adaptive Logic System with Generalizing properties"* Ph. D. Thesis Tech. Rep., Stanford Electron Labs., Stanford, C.A. April, 1962.

Widrow, and M.E. Hoff *"Adaptive Switching Circuits"*, 1960 IRE WESCON Conv. Record, Part 4, 96-104 August 1960.

Widrow, B. and. M. Hoff. *"Adaptive Switching Circuits"*. in WESCON Convention Records, Vol. 4, 1960.