UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SIMULADOR DE REDES NEURONALES ARTIFICIALES

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A

GIOVANNI LEONETTI DEL NEGRO

DIRECTOR: DR. ISMAEL ESPINOSA ESPINOSA

MEXICO, D. F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INTRODUCCION

En la última década, el término "Redes Neuronales" ha sido utilizado ampliamente por investigadores y aficionados para promocionar el uso y desarrollo de ésta tecnología, sin embargo, a pesar de la información que existe al respecto, parece ser que en nuestro país se ha mitificado y, en lugar de ser tratada como una disciplina, se le ha visto, válgase la expresión, como una "ciencia oculta".

Desde tiempos memorables el sueño más añorado del hombre, sin considerar la conquista del Universo, ha sido el de crear un ser animado hecho a semejanza que pueda ser autónomo y pensante. El esmero del hombre se ha visto encaminado sobre todo en robótica, área en la que se han realizado grandes adelantos. No obstante, las más fuertes limitantes que a la fecha no han podido ser superadas son las de imitar el funcionamiento del sistema nervioso y, aún más difícil, el de producir pensamiento artificial.

Hace ya varios años que investigadores en áreas como neurobiología, fisiología, matemáticas y física, han desarrollado modelos electrónicos y computacionales que simulan tan sólo algunas de las funciones plenamente identificadas del sistema nervioso central. Estos modelos, neurológicamente inspirados, han sido llamados "Redes Neuronales Artificiales" o simplemente "Redes Neuronales". Esta tecnología utiliza unidades de procesamiento simples y conexiones semejantes a las neuronas y sinapsis del cerebro. La única comparación aceptable entre las redes neuronales artificiales y el sistema nervioso, es que las primeras se comportan de manera vagamente similar a las redes biológicas.

En los años cercanos a 1950, paralelamente al desarrollo de las redes neuronales, se observó por primera vez la posibilidad de modelar el proceso del pensamiento como una caja negra. Con el apoyo de las computadoras surgieron los sistemas expertos en el área que se denominó "Inteligencia Artificial". Esta disciplina, más tangible y concebible que las redes neuronales, tuvo tanto éxito que por más de veinte años opacó las investigaciones sobre redes neuronales, remitiéndolas prácticamente a la clandestinidad.

Los sistemas expertos utilizan conceptos de razonamiento de alto nivel a los que se han llamado "Reglas", éstas reglas que son desarrolladas por expertos permiten establecer una relación a priori entre la entrada y la salida, simulando el pensamiento. Las redes neuronales, en cambio, realizan el proceso de más bajo nivel en el cerebro; no se programan, no requieren información de como procesar la entrada, simplemente toman decisiones basadas en una serie de ejemplos aprendidos.

En la actualidad, las dos áreas han encontrado aplicaciones diversas en las que pueden interactuar, y se han generado sistemas híbridos que resuelven problemas muy complejos.

Dentro del área de inteligencia artificial se ha desarrollado en los últimos años una nueva tecnología, llamada lógica difusa, que utiliza el concepto de incertidumbre del pensamiento para emular la manera en que la gente toma realmente sus decisiones. La lógica difusa abre el campo de aplicación de los sistemas expertos convencionales y reduce de un 50% a un 80% el número de reglas utilizadas normalmente.

Lofti Zadeh, autor de la lógica difusa, propuso en los años sesentas que la lógica computacional no debía estar restringida a dos valores de verdad (0 y 1) exclusivamente. En lógica difusa se maneja un rango infinito de valores de verdad comprendidos entre 0 y 1, permitiendo que un gran número de funciones matemáticas continuas puedan ser consideradas para resolver un problema particular. Maneja conceptos tales como casi, moderado, tibio, tarde, etc. y permite la violación de la ley de no contradicción de la lógica tradicional.

Al igual que los sistemas expertos un sistema de lógica difusa está compuesto por una colección de reglas y su semejanza con las redes neuronales artificiales es prácticamente nula.

Existen también los métodos tradicionales de aprendizaje, tales como los clasificadores Bayesianos, los discriminadores lineales o el método del vecino más cercano, que tienen un fuerte fundamento matemático teórico y práctico. Estos métodos son empleados generalmente para el reconocimiento de patrones y aprenden o clasifican de acuerdo a eventos estadísticos presentes en los datos de entrada.

La base de muchos modelos de redes neuronales artificiales es precisamente la teoría que sustenta a los métodos tradicionales, no obstante, los métodos tradicionales utilizan modelos estadísticos mientras que las redes neuronales utilizan modelos conexionistas.

Las redes neuronales han sido utilizadas con buenos resultados en distintas áreas. Entre las aplicaciones en las que han destacado encontramos la medicina en el diagnóstico de enfermedades, la manufactura con el manejo de sistemas de control autónomo, el procesamiento de imágenes, el reconocimiento de voz, el procesamiento de señales haciendo las veces de filtro, compactadores o separadores de ondas, etc..

El objetivo de esta tesis es el de promover una formación básica en redes neuronales a estudiantes universitarios, contando con los elementos suficientes para plantear y resolver problemas prácticos con esta tecnología y proporcionar la bibliografía

especializada para su estudio y profundización. Para estos fines se ha desarrollado un simulador que contiene diversos algoritmos de aprendizaje y con el que se puede reproducir casi cualquier ejemplo que pudiera encontrarse en la bibliografía.

En el primer capítulo se establecen los conceptos básicos que se manejan en el ambiente de redes neuronales artificiales. Se contemplan los antecedentes históricos, la diferencia entre las redes neuronales biológicas y artificiales en el aspecto fisiológico y estructural, el aprendizaje en la redes artificiales y el elemento procesador o neurona de una red neuronal artificial.

En el segundo capítulo se integran los modelos de redes neuronales artificiales cuyo aprendizaje es del tipo supervisado. En el aprendizaje supervisado las redes neuronales aprenden a relacionar un conjunto de patrones de entrada con un conjunto de patrones de salida específico. Se han dividido para su estudio en redes con procesadores binarios y con procesadores análogos.

El tercer capítulo trata sobre los modelos que aprenden sin supervisión. Este tipo de aprendizaje deja a las redes neuronales la tarea de clasificar o agrupar el conjunto de patrones de entrada de acuerdo a su distribución estadística.

En el cuarto capítulo se estudia un modelo que integra los dos tipos distintos de aprendizaje. Estos modelos son conocidos como híbridos, rescatan los rasgos característicos del aprendizaje supervisado para asociar patrones de entrada con patrones de salida y emplean el aprendizaje no supervisado para clasificar los patrones de entrada.

El quinto capítulo, al que se le ha llamado tutorial, describe las funciones principales del simulador desarrollado para la implementación de redes neuronales artificiales. En el simulador están incluidos todos los modelos descritos en la tesis, la teoría puede ser entonces ratificada mediante aplicaciones prácticas.

Se incluyen además tres apéndices. El apéndice A describe el modelo de inhibición lateral empleado por tres de los algoritmos incluidos en el simulador. El apéndice B muestra un método para obtener vectores ortogonales, éstos son utilizados en uno de los algoritmos. Finalmente, en el apéndice C se muestran un diagrama de flujo de datos de la programación del simulador y los programas principales del mismo.

CAPITULO 1

REDES NEURONALES ARTIFICIALES

1.1 ANTECEDENTES HISTORICOS

La razón por la que los modelos de redes neuronales reciben este nombre, se debe principalmente a que el estudio y desarrollo de los mismos fue hecho por gente de las áreas biológicas, fisiológicas y sicológicas.

El primer dato que se tiene sobre publicaciones relacionadas con redes neuronales artificiales es alrededor del año 1943. Mc Culloch y Pitts, basados en lo que se conocía sobre estructuras biológicas en ese entonces, establecieron teoremas relacionados con modelos neuronales:

- La actividad de una neurona se define como un proceso "todo o nada".
- Cierto número de sinapsis se activan , durante un período de adición, con el fin de excitar una neurona. Este número de sinapsis es independiente de la actividad previa de dicha neurona.
- La estructura de la red es invariante con el tiempo.

Como resultado de sus estudios surgió el primer modelo matemático de neurona, al que llamaron "Neurona Formal". Una red conformada por éstas neuronas es capaz de realizar casi cualquier expresión lógica finita.

En el año de 1949 Donald O. Hebb definió por primera vez un método para la asignación de pesos en la sinapsis de acuerdo a la actividad neuronal. A este método se le conoce como "Sinapsis Hebbiana", y está asentado como un postulado neurofisiológico:

> *Cuando el axón de una neurona A está en contacto con las dendritas de una neurona B y la excita repetidamente, algún proceso metabólico toma lugar en ambas células. De manera tal que la eficiencia de la neurona A como una de las neuronas que excita a la neurona B, se incrementa.*

De aquí nace un concepto crucial en las redes neuronales, "La información en una red neuronal está almacenada en los pesos de las sinapsis (conexiones) y no en las neuronas (elementos procesadores)".

Posteriormente, en 1958, Frank Rosenblatt propuso una estructura de red neuronal de dos capas de neuronas formales, a la que denominó "Perceptrón". Siendo éste el primer modelo simulado en una computadora*. Los perceptrones realizan una tarea de

* Laboratorio de Aeronáutica de Cornell, computadora IBM 704.

aprendizaje basada en la sinapsis Hebbiana y son capaces de aprender a clasificar ciertos patrones como similares o distintos mediante la modificación de sus conexiones.

Para 1960, Bernard Widrow y Marcian Hoff introdujeron un dispositivo llamado "Adaline" (Adaptive Linear Element) que consiste en sólo una neurona con un número arbitrario de elementos de entrada que pueden tomar valores de ± 1 y un umbral de activación, comúnmente llamado bias, cuyo valor es siempre uno. Las entradas, incluyendo el bias, antes de ser sumadas por la neurona son modificadas por un peso al que los autores llamaron ganancia. La suma es pasada por una función de transferencia que define el nivel de activación de la neurona, si la suma es mayor o igual a cero la neurona se activa, de cualquier otra forma permanece inactiva. El trabajo de Widrow y Hoff fue el primero en ser implementado físicamente, además de ser simulado por computadora.

En el año de 1969 Marvin Minsky y Seymour Papert publicaron un libro llamado Perceptrones en el que realizaron un vasto análisis sobre el trabajo de Rosenblatt. Argumentaron que debido a que los perceptrones no tienen capacidad computacional booleana universal (no resuelven el Xor con el modelo propuesto por Rosenblatt) la mayor parte del trabajo escrito sobre perceptrones carecía de valor científico.

Hasta ese entonces no se había encontrado un método eficiente para la asignación de los pesos de las conexiones que evidenciara la capacidad cognoscitiva de las redes neuronales artificiales. Además, el poder de cómputo con el que se contaba era muy limitado. Por éstas razones, aunadas al trabajo de Minsky y Papert, los estudios que había sobre redes neuronales artificiales quedaron inactivos, dando lugar al desarrollo de sistemas expertos en el área de inteligencia artificial.

Después de 13 años de oscuridad, período durante el cual desaparecieron prácticamente las publicaciones sobre redes neuronales artificiales, el Dr. Teuvo Kohonen publicó una arquitectura de red neuronal muy distinta a la del perceptrón. Estas redes son actualmente conocidas con el nombre de redes auto-organizativas y son capaces de categorizar información sin saber el número de categorías que existen.

Sin embargo, no fue hasta el año de 1982, cuando el físico John J. Hopfield reunió las ideas más importantes que existían sobre redes neuronales, que se reactivaron las investigaciones sobre el tema. Hopfield identificó arquitecturas y algoritmos para redes neuronales que pueden ser generalizados y que tienen un alto grado de robustez. Se atribuye a Hopfield el hecho de reactivar el interés debido a que mostró sus redes de una manera muy sencilla, en especial para ingenieros, y la relación que éstas tienen con trabajos anteriormente realizados. Además propuso la implementación de sus redes en circuitos integrados.

En 1986, se conforma un grupo de 16 investigadores sobre Procesamiento Distribuido en Paralelo (PDP). Este grupo publicó dos libros que contenían la mayor parte de las investigaciones que había sobre redes neuronales hasta ese entonces. Los libros fueron editados por los doctores James L. McClelland y David E. Rumelhart, quienes redescubrieron el método de asignación de pesos más popular a la fecha, Retro propagación de errores.

A partir de ese año aparecieron un gran número de publicaciones que mostraron, de manera significativa, el interés y la cantidad de investigadores que se han sumado a la investigación de redes neuronales.

1.2 REDES NEURONALES ARTIFICIALES Y REDES NEURONALES BIOLOGICAS

El cerebro humano contiene cerca de $10^{12\pm1}$ células nerviosas, cada una de éstas puede tener hasta 10,000 interconexiones con otras células nerviosas. El poder computacional del cerebro recae sustancialmente en la densa interconexión de sus neuronas.

El funcionamiento del cerebro se lleva al cabo probablemente en grupos de neuronas que actúan como unidades funcionales o módulos. Las neuronas de cada módulo se activan simultáneamente o generan patrones de actividad particulares en respuesta a estímulos distintos.

El cerebro está cubierto por un manto de sustancia gris, la corteza cerebral, cuyo tamaño y complejidad constituyen una diferencia crítica entre el ser humano y los animales inferiores. Es en la corteza donde se han identificado distintas regiones que se especializan en tareas tales como el habla, el entendimiento del lenguaje, análisis de información visual, organización de actividades motoras y otros aspectos importantes de la inteligencia.

Una red neuronal artificial es un conjunto de elementos procesadores o nodos (neuronas en los sistemas biológicos) conectados unos a otros de acuerdo a arquitecturas específicas y con pesos o intensidades modificables en la conexión (sinapsis) de un procesador a otro. En términos más generales, un red neuronal es un modelo inspirado en las neuronas biológicas que constituyen los cerebros de animales y humanos.

En semejanza al sistema nervioso, los modelos de redes neuronales artificiales realizan distintas tareas -- reconocer, completar y clasificar patrones -- eficientemente, mediante la densa interconexión de sus procesadores.

Sin embargo, existen grandes diferencias entre una estructura neuronal biológica y la implementación de una red neuronal artificial, entre las que destacan:

- En una red neuronal artificial típica, las conexiones de un procesador a sus vecinos pueden ser positivas y negativas que corresponde a las conexiones excitatorias o inhibitorias neuronales respectivamente. La diferencia estriba en que en las redes biológicas la sinapsis es siempre excitatoria o inhibitoria, en las redes artificiales una conexión puede cambiar de positiva a negativa, y viceversa, de acuerdo al entrenamiento.
- La información de los estados de activación en las redes artificiales, pasados de un procesador a otros mediante las conexiones, son valores que representan niveles de corriente directa. En redes neuronales biológicas, un tren de pulsos a través de la conexión (axón) contiene la información, correspondiendo a un nivel alto de activación una frecuencia elevada de pulsos, por lo que el nivel de activación se mide en frecuencia de corriente alterna.
- En un sistema biológico se pueden encontrar, en un radio pequeño, distintos tipos de neuronas; mientras que una red neuronal artificial es generalmente implementada con un solo tipo de procesador.
- Existe una considerable diferencia entre el número de procesadores en una red neuronal artificial típica y el número de neuronas involucradas en cualquier tarea en una red neuronal biológica. El número de procesadores de una red neuronal artificial va desde una docena hasta varios cientos, mientras que en una red biológica se habla de 500 millones de neuronas por módulo.

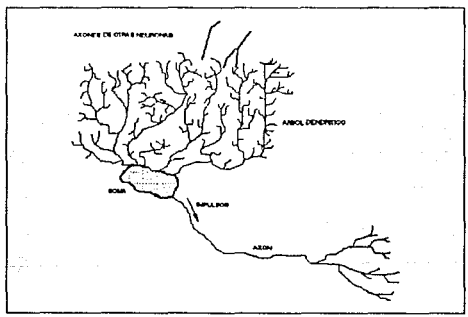
Las redes neuronales artificiales tienen generalmente topologías de interconexiones regulares, basadas en un organización por capas totalmente interconectadas. Las interconexiones biológicas en cambio, tienen una estructura definida por el sistema al que pertenecen. Considerando además el hecho de que el cerebro se organiza dinámicamente durante un período de desarrollo, y puede cambiar indefinidamente las interconexiones de sus neuronas basado en la experiencia durante un período crítico del desarrollo.

El procesamiento dinámico es también diferente. Las redes neuronales artificiales realizan cambios de forma simultánea o en intervalos regulares. Los sistemas biológicos ajustan las características de cada neurona continuamente, y llevan al cabo cambios asíncronos al iniciar o recibir estímulos.

1.3 LA NEURONA FORMAL

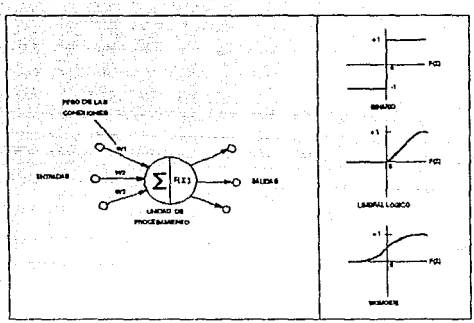
A pesar de que el actual conocimiento del cerebro es limitado, se cuenta con mucha información detallada sobre aspectos anatómicos y fisiológicos del mismo. La anatomía básica de una célula nerviosa individual (neurona) y la mayor parte de las reacciones bioquímicas que gobiernan su actividad están totalmente identificadas.

En la figura 1.1 se muestra una célula nerviosa típica del cerebro humano. El área de salida de la neurona es una fibra larga, con forma de raíz, llamada axón. La célula es capaz de generar una serie de impulsos y enviarlos a través del axón a otras células vecinas. El área de entrada de una célula nerviosa es un conjunto de fibras, con forma de ramas, llamadas dendritas. El punto de conexión entre el axón y las dendritas de otras neuronas es la sinapsis. Cuando una serie de impulsos es recibida en el área dendrítica de una neurona, el resultado es generalmente una probabilidad mayor de que la neurona receptora dispare un impulso a lo largo de su axón.



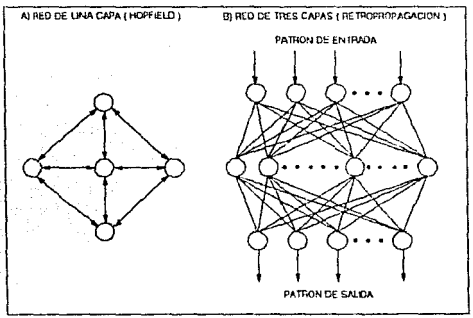
1.1 Célula nerviosa típica del cerebro humano

El procesamiento de señales de entrada en una neurona artificial es relativamente sencillo, normalmente involucra una sumatoria de entradas escalares pasada por una función de transferencia o activación. El proceso de la sinapsis en un sistema biológico es mucho más complejo, e integra una microestructura detallada con la participación de varios componentes bioquímicos.



1.2 Neurona Artificial, se muestran tres funciones de activación distintas

En la figura 1.2 se puede apreciar un ejemplo de una unidad de procesamiento típica de una red neuronal artificial. Del lado izquierdo se encuentran una serie de entradas, conectadas a la neurona mostrada en el centro, provenientes de otras neuronas. Cada interconexión tiene asociada un peso, dados como w_1, w_2,....,w_n. La neurona realiza una suma de los pesos a la entrada y utiliza una función de umbral no lineal, f, para obtener su salida. El resultado es enviado a través de las conexiones de salida a otras neuronas (lado derecho de la gráfica).



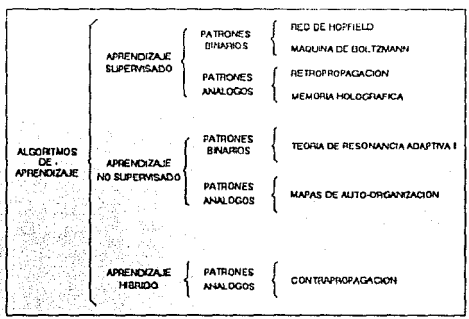
1.3 Distintos tipos de arquitectura

En una red neuronal artificial las neuronas están generalmente organizadas por capas totalmente interconectadas entre sí. La arquitectura se define de acuerdo al algoritmo de entrenamiento que se emplee, así pues, existen redes neuronales de una sola capa que son entrenadas con el algoritmo de Hopfield, figura 1.3.A. En la figura 1.3.B se muestra una red neuronal de tres capas de procesadores, organización típica para el entrenamiento por retro- propagación de errores.

1.4 APRENDIZAJE

Las redes neuronales no se programan, ellas aprenden por medio de ejemplos. Generalmente, un conjunto de patrones de entrenamiento es mostrado a la red neuronal para que ésta pueda aprender. Los ejemplos o patrones de entrenamiento, están representados como vectores, y pueden ser obtenidos, a partir de transformaciones, de fuentes tales como imágenes, señales de voz, sensores de datos, movimiento de brazos robóticos, datos financieros, etc.

Existen dos clases de entrenamiento. El entrenamiento es supervisado cuando a la red se le entrena con patrones de entrada asociados a patrones de salida. Los patrones de salida constituyen la respuesta correcta de la red o la clasificación correcta del patrón de entrada. En respuesta a los pares de ejemplos, la red neuronal ajusta los valores de las interconexiones. Si el entrenamiento tiene éxito, los parámetros internos son ajustados al punto en que la red puede producir la respuesta correcta para cada patrón de entrada.



1.4 Clasificación de siete distintas redes neuronales

El entrenamiento es no supervisado cuando la red ajusta sus pesos en respuesta a distintos patrones de entrada sin que estén asociados a patrones de salida. En el aprendizaje no supervisado la red clasifica los patrones de entrada en categorías similares.

El presente trabajo abarca distintas arquitecturas de redes neuronales que emplean ambos tipos de entrenamiento. En la figura 1.4 se puede apreciar la organización básica que sigue la investigación de este documento.

CAPITULO 2

REDES NEURONALES ARTIFICIALES CON APRENDIZAJE SUPERVISADO

2.1 CARACTERISTICAS GENERALES

Los escenarios de entrenamiento para redes neuronales artificiales más comúnmente utilizados, emplean el aprendizaje supervisado, durante el cual la red asocia una serie de patrones de entrada con un conjunto de patrones de salida.

Las redes entrenadas de esta manera sirven como clasificadores o memorias asociativas. El patrón de salida representa normalmente la respuesta correcta o la clasificación correcta para el patrón de entrada.

El aprendizaje supervisado se puede definir de la siguiente manera. Un patrón de entrada es mostrado a la red para que ésta genere una salida determinada, la salida es comparada entonces con el patrón de salida deseado y con base en la diferencia se adaptan los pesos de la red conforme a las reglas del entrenamiento.

El procedimiento se repite, generalmente, un gran número de veces durante el entrenamiento para que la red ajuste de forma gradual los pesos de las conexiones y llegue al punto en que puede producir la respuesta correcta para cada patrón de entrada.

2.2 MODELOS PARA PATRONES BINARIOS

2.2.1 REDES DE HOPFIELD
JOHN J. HOPFIELD

John Hopfield publicó en el año de 1982 un artículo en el que introdujo una arquitectura de red neuronal artificial capaz de solucionar problemas de optimización y de funcionar como memoria asociativa. Gracias a dicha publicación resurgió el interés en las investigaciones, prácticamente olvidadas, sobre redes neuronales.

La red neuronal de Hopfield cuenta solamente con una capa de unidades binarias. La entrada a la red se representa por un vector binario [0,1] en el que cada elemento corresponde solamente a una unidad en la red.

Se considera que la red tiene un estado global en cada momento, dicho estado se representa también por un vector de ceros y unos:

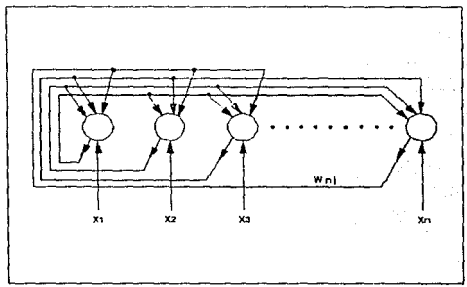
$$U = (u_1, u_2, ..., u_n)$$

Este vector refleja una red de n unidades, y el estado de la i-ésima unidad es u_i.

En la red de Hopfield cada unidad está conectada con todas las restantes, pero no consigo misma, y los pesos son simétricos:

$$W_{ij} = W_{ji}$$

Donde W_{ij} es el valor escalar de la interconexión entre la i-ésima y la j-ésima unidad. En la figura 2.1 se muestra la arquitectura de la red de Hopfield.



2.1 Arquitectura de una red de Hopfield

Esta configuración hace a la red recursiva, debido a que las salidas de cada unidad van direccionadas a las entradas de otras unidades en la misma capa cuyas salidas a su vez están conectadas con todas las demás. Esta propiedad permite a la red encontrar estados estables en ausencia de entrada externa.

La red de Hopfield almacena el conjunto de patrones en un solo paso de aprendizaje, en el que se definen los valores de los pesos de las interconexiones. Esto quiere decir que se debe de conocer de antemano al conjunto de patrones que debe aprender la red; esta restricción hace a la red no adaptiva, es decir, no puede aprender en tiempo real. Además, para obtener un buen funcionamiento de la red, los patrones deben ser tan ortogonales como sea posible.

Los pesos de las interconexiones dependen de los patrones por ser almacenados. Debido a que los pesos en la red son simétricos, el total de las conexiones puede ser representada por una matriz simétrica. La matriz de conexiones está compuesta básicamente por la suma de los patrones (representados por vectores), cada uno multiplicado por su transpuesto.

En forma general se puede definir el peso W_{ij} de la siguiente manera:

$$W_{ij} = \sum_{p=1}^{m} (2x_{pi} - 1)(2x_{pj} - 1) \quad - - - - - - - - - - \quad \{1\}$$
$$i \neq j$$

Donde x_{pi} es el i-ésimo elemento del patrón de entrada p, y m es el número total de patrones.

Este método de asignación de pesos presume ser un tipo de aprendizaje Hebbiano entre la sinapsis de dos neuronas. Si se cumple que $x_{pi}=x_{pj}$ entonces la conexión entre la i-ésima y la j-ésima neuronas es mutuamente excitatoria, es decir, una refuerza a la otra:

Si $x_{pi}=1$ y $x_{pj}=1$ entonces, sustituyendo en {1}: (2-1)(2-1)=1

Si $x_{pi}=0$ y $x_{pj}=0$ entonces, sustituyendo en {1}: (0-1)(0-1)=1

Si por el contrario, $x_{pi} \neq x_{pj}$ entonces la conexión entre las neuronas es mutuamente inhibitoria:

Si $x_{pi}=1$ y $x_{pj}=0$ entonces, sustituyendo en {1}: (2-1)(0-1)=-1

MODO DE OPERACION

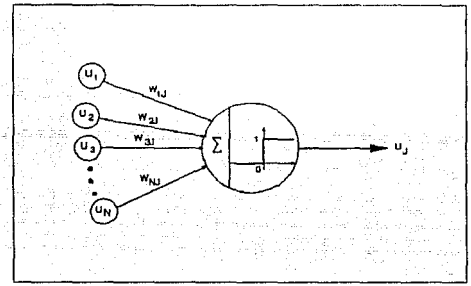
Inicialmente se asigna a la red un estado para cada unidad, dado por un patrón de entrada ruidoso. Cada unidad realiza la suma de sus entradas de la siguiente manera:

$$s_i = \sum_{i \neq j}^{n} u_j(t)\ w_{ij}$$

Una vez hecha la suma, se hace pasar por la función de activación para las unidades. Si la suma es mayor o igual que cero entonces el siguiente estado de la unidad es uno, de cualquier otra manera es cero:

$$u_i(t+1) = \begin{cases} 1 \text{ si } s_i \geq 0 \\ 0 \text{ si } s_i < 0 \end{cases}$$

En la figura 2.2 se puede apreciar la función de activación utilizada por las neuronas de la red de Hopfield.



2.2 Neurona utilizada por Hopfield

El nivel de activación previo de la unidad no se toma en cuenta para el cambio de estado, la unidad cambia o no de un valor a otro de acuerdo al estado de las demás unidades.

En el mecanismo de actualización de estados propuesto por Hopfield la unidad elegida para efectuar el cambio de estado es escogida aleatoriamente, lo que permite a todas las unidades tener la misma probabilidad de ser actualizadas. Conforme evoluciona la red se obtiene un estado estable, es decir, todas las unidades mantienen su estado aún después de la actualización.

El procedimiento de actualización debe repetirse, generalmente, un gran número de veces antes de que la red alcance un estado estable. El hecho de elegir aleatoriamente una unidad para actualizarla, hace al algoritmo de Hopfield semejante al sistema neurológico, pues las neuronas biológicas envían pulsos a través de su axón debido a eventos que, en la mayoría de los casos, no tienen sincronía con las neuronas vecinas.

Cada estado en la red de Hopfield tiene asociada una energía, definida por:

$$E = -\frac{1}{2} \sum_i \sum_j W_{ij}\, u_i\, u_j$$

Esta ecuación de energía, así como en todos los modelos de redes neuronales artificiales, no representa la energía real de algún sistema físico; es una función que debe ser minimizada por la red. En los pasos sucesivos de actualización en la red de Hopfield la energía se reduce progresivamente. Cuando la red alcanza un estado estable la energía

de la red es mínima, si el mínimo de energía es global entonces la red encontró una solución al patrón de entrada, si es local la solución que entrega es espuria.

Cada vez que es actualizada una unidad, la energía de la red permanece igual o decrece, por lo que el algoritmo de actualización hace que la energía sea siempre convergente en un mínimo. La contribución de energía de la j-ésima unidad a la energía global de la red está dada por:

$$E_j = -\tfrac{1}{2}\, u_j \sum_{\substack{i \\ i \neq j}} W_{ij}\, u_i$$

Cuando la j-ésima unidad es actualizada y no cambia de estado, la diferencia de energías es cero:

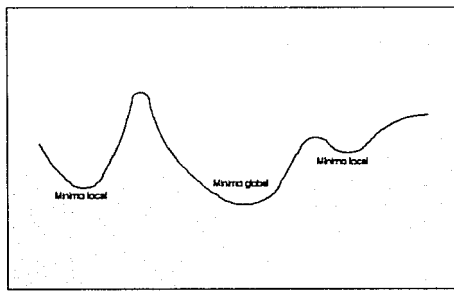
$$E_j(t+1) = E_j(t) - E_j(t-1) = -\tfrac{1}{2}\, u_j \sum W_{ij}\, u_i$$

Donde $u_j=0$.

Si la j-ésima unidad cambia del estado 0 al 1, entonces $u_j=1$, para que este cambio ocurra $\Sigma\, W_{ij}\, u_i$ debe ser mayor o igual que cero por lo que $E_j \leq 0$.

Si cambia del estado 1 al 0 entonces $u_j=-1$, cuando ésto ocurre $\Sigma\, W_{ij}\, u_i$ debe ser forzosamente menor que cero, por lo que $E_j < 0$.

Los cambios de energía en la red durante el procedimiento de actualización producen una superficie semejante a un terreno montañoso, figura 2.3.



2.3 Función de energía en dos dimensiones

Si la red es atrapada en un mínimo local no existe ninguna manera de alcanzar un mínimo global a partir de ese estado, sin embargo, la red puede ser reinicializada con el mismo patrón de entrada y de acuerdo con la selección aleatoria de unidades por ser actualizadas, la red puede partir de otro punto en la función de energía y alcanzar el mínimo global esperado.

El algoritmo de Hopfield tiene tres grandes deficiencias. La primera es que no todos los patrones aprendidos se recuerdan con el mismo énfasis. En algunos casos los patrones con menor energía son recordados inevitablemente.

La segunda es que en ocasiones la red converge en soluciones espurias debido a que en el procedimiento se alcanzó un mínimo local, la causa de éste problema es la correlación que existe entre los vectores que aprendió la red. Hopfield propuso un proceso de desaprendizaje para dar una solución parcial a estos dos problemas, en este proceso los pesos cambian de acuerdo a la siguiente ecuación:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(2x_{pi} - 1)(2x_{pj}-1)$$

Donde α tiene un valor muy pequeño, $0 < \alpha << 1$, y p es el patrón que no es recuperado debidamente. Este proceso equilibra la accesibilidad de la memoria y reduce los estados espurios.

La tercera desventaja es que existe un límite en la cantidad de patrones que puede ser almacenada por la red y que es proporcional al número de unidades. Este límite fue determinado experimentalmente por Hopfield en simulaciones por computadora. El número total de patrones que pueden ser almacenados y recordados correctamente es de aproximadamente 0.15N, donde N es el número de unidades en la red. Si se excede éste límite entonces puede ocurrir un número significativo de errores cuando la red es utilizada para recordar patrones. No obstante, si el conjunto de patrones es ortogonal el número de patrones que pueden ser almacenados se incrementa.

Si la red es atrapada en un mínimo local no existe ninguna manera de alcanzar un mínimo global a partir de ese estado, sin embargo, la red puede ser reinicializada con el mismo patrón de entrada y de acuerdo con la selección aleatoria de unidades por ser actualizadas, la red puede partir de otro punto en la función de energía y alcanzar el mínimo global esperado.

El algoritmo de Hopfield tiene tres grandes deficiencias. La primera es que no todos los patrones aprendidos se recuerdan con el mismo énfasis. En algunos casos los patrones con menor energía son recordados inevitablemente.

La segunda es que en ocasiones la red converge en soluciones espurias debido a que en el procedimiento se alcanzó un mínimo local, la causa de éste problema es la correlación que existe entre los vectores que aprendió la red. Hopfield propuso un proceso de desaprendizaje para dar una solución parcial a estos dos problemas, en este proceso los pesos cambian de acuerdo a la siguiente ecuación:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(2x_{pi} - 1)(2x_{pj}-1)$$

Donde α tiene un valor muy pequeño, $0 < \alpha << 1$, y p es el patrón que no es recuperado debidamente. Este proceso equilibra la accesibilidad de la memoria y reduce los estados espurios.

La tercera desventaja es que existe un límite en la cantidad de patrones que puede ser almacenada por la red y que es proporcional al número de unidades. Este límite fue determinado experimentalmente por Hopfield en simulaciones por computadora. El número total de patrones que pueden ser almacenados y recordados correctamente es de aproximadamente 0.15N, donde N es el número de unidades en la red. Si se excede éste límite entonces puede ocurrir un número significativo de errores cuando la red es utilizada para recordar patrones. No obstante, si el conjunto de patrones es ortogonal el número de patrones que pueden ser almacenados se incrementa.

ALGORITMO DE APRENDIZAJE

1.- Definir la matriz de pesos:

$$W_{ij} = \begin{cases} 0 & \text{si } i=j \\ \sum_{p=1}^{m} (2x_{pi}-1)(2x_{pj}-1), & \text{para } i \neq j \end{cases}$$

Donde x_{pi} es el i-ésimo elemento del patrón de entrada p, y m es el número total de patrones.

2.- Inicializar los estados de cada neurona con un patrón ruidoso.

$$u_i(0) = x_i$$

Donde u_i es el estado de la neurona (0 ó 1) y x_i es un patrón de entrada.

3.- Para cada neurona de la red, elegidas aleatoriamente, realizar la siguiente operación:

$$S_i = \sum_{i \neq j}^{n} u_j(t)\, w_{ij}$$

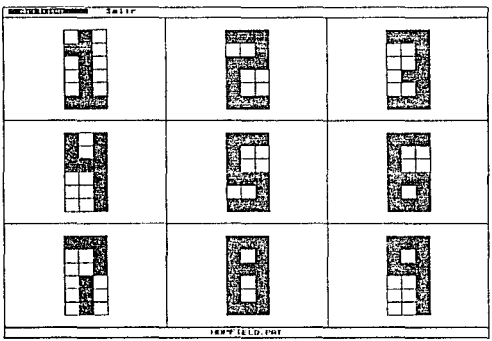
El estado siguiente de la i-ésima neurona queda definido por:

$$u_i(t+1) = \begin{cases} 1 \text{ si } S_i \geq 0 \\ 0 \text{ si } S_i < 0 \end{cases}$$

Este paso se realiza indefinidamente hasta que todas las neuronas de la red permanezcan en un estado estable.

EJEMPLO

Para ilustrar el algoritmo de Hopfield, se ha incluido en el simulador una red de 18 neuronas que es entrenada para aprender un conjunto de nueve patrones reticulares. Los patrones, mostrados en la figura 2.4, son los dígitos del 1 al 9 formados en retículas de 3 cuadros de base por 6 de altura.



2.4 Archivo de patrones reticulares HOPFIELD.PAT

Los pesos que unen a las neuronas de una red de Hopfield son determinados matemáticamente, de forma directa, en una matriz simétrica. Debido a ésto, las redes entrenadas bajo el algoritmo de Hopfield no tienen un proceso de aprendizaje que pueda ser apreciado visualmente.

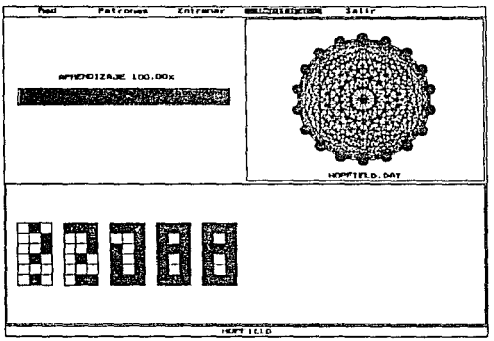
Durante el entrenamiento de una red de Hopfield se despliega del lado superior izquierdo de la pantalla una barra que indica el porcentaje de conexiones que se han establecido y representa, de alguna manera, el desarrollo del aprendizaje de la red.

La arquitectura, mostrada en la parte superior derecha de la pantalla, es un arreglo circular de 18 neuronas en una sola capa. Cuando las neuronas que conforman la red exceden de 25 es difícil poder apreciar las conexiones.

El único parámetro que se debe establecer para el entrenamiento de una red de Hopfield es la constante de desaprendizaje. Esta constante se utiliza para disminuir la profundidad de los pozos, en la superficie de energía generada por la red, producidos por

patrones de muy baja energía. Para el ejemplo se utilizó un valor de 0.15 para la constante de desaprendizaje.

Al verificar el aprendizaje, aparece en la pantalla una secuencia de retículas, iguales a las de entrada, que representan el número de pasos que tomó a la red alcanzar un estado estable. Como se puede apreciar en la figura 2.5 el patrón de prueba, el más a la izquierda de la secuencia de retículas, es una versión ruidosa del dígito 7. La red alcanza, después de cinco pasos, un estado estable no deseado pero que si representa un patrón aprendido por la red.



2.5 Verificación del aprendizaje en una red de Hopfield

Para reproducir el ejemplo en el simulador es necesario leer el capítulo 5.

BIBLIOGRAFIA

1.-J. J. Hopfield.

Neural networks and physical systems with emergent collective computational abilities.

Proc. Natl Acad. Sci. USA, Vol. 79, pp 2554-2558, Abril 1982.

2.-J. J. Hopfield, D. I. Feinstein, R. G. Palmer.

Unlearning has a stabilizing effect in collective memories.

Nature, Vol. 304, pp. 148-149, Julio 1983.

3.-J. J. Hopfield.

Neurons with graded response have collective computational properties like those of two-state neurons.

Proc. Natl. Acad. Sci. USA, Vol. 81, pp. 3088-3092, Mayo 1984.

2.2.2 MAQUINAS DE BOLTZMANN

HINTON, SEJNOWSKI Y ACKLEY

La máquina de Boltzmann diseñada por Geoffrey Hinton, Terrence Sejnowski y David Ackley es una red neuronal de tres capas que encuentra sus principios en el trabajo de Kirkpatrick, Gellat y Vechi llamado "temple simulado".

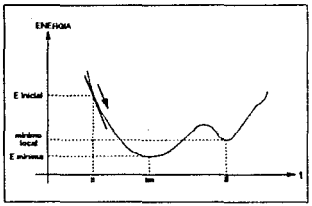
La máquina de Boltzmann es utilizada para almacenar patrones binarios y su característica esencial es que las unidades binarias que la conforman adoptan cualquiera de sus dos estados, prendido o apagado, como una función probabilística de los estados de las unidades vecinas y de los pesos conectados a ellas.

La manera en que funciona la máquina de Boltzmann tiene sus bases en la red neuronal de Hopfield, en la cual se genera un espacio de energía definido por una función de Liapunov. Las soluciones o estados estables encontrados por una red de Hopfield son precisamente los mínimos en el espacio de energía producidos por la distribución de probabilidad sobre los dos posibles estados de cada neurona. En la máquina de Boltzmann la distribución de probabilidad está repartida en las neuronas de la capa intermedia.

El algoritmo de aprendizaje, conocido como "Temple simulado", permite encontrar los valores óptimos en los estados de las neuronas intermedias que generan la energía mínima para cada patrón de entrada. El método de temple simulado es también conocido como estocástico o estadístico pues, para su operación es necesario generar eventos aleatorios para después evaluar sus efectos en términos de distribuciones de probabilidad y en la obtención de resultados deseados.

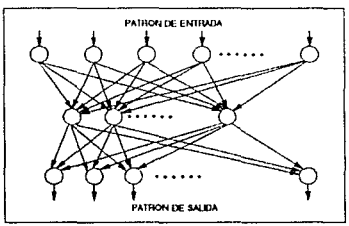
Para comprender el temple simulado se debe hacer cierta analogía entre el estado global de energía de la red neuronal y el estado de energía de un sólido que es templado lentamente. Suponemos que las partículas del sólido pueden tomar sólo dos posibles estados, uno de alta energía correspondiente a una unidad prendida y uno de baja energía que corresponde a una unidad apagada.

La energía global del sólido depende de dos factores, la combinación de energía de cada partícula individual y la entropía negativa (la forma en que están ordenadas las partículas) multiplicada por la temperatura del sólido. La analogía parte del hecho que el sólido encuentra el equilibrio térmico en el punto más bajo de la curva de energía generada. En la figura 2.6 se puede observar el ejemplo de una curva de energía generada al templar un sólido.



2.6 Gráfica de la energía

Si la temperatura es disminuida lentamente, el sólido tendrá la oportunidad de encontrar el punto más bajo de energía, aún si existen varios puntos de energía mínima. A bajas temperaturas existe una fuerte tendencia en favor de los estados con baja energía, pero el tiempo requerido para alcanzar el equilibrio es largo. A altas temperaturas los estados de baja energía no son encontrados fácilmente pero el equilibrio es encontrado rápidamente. Debido a que el número de partículas en un estado energético es función de la probabilidad, la proporción entre los diferentes estados puede expresarse como una distribución de probabilidad. En la máquina de Boltzmann se pretende realizar una tarea semejante; se crea una variable de temperatura artificial, con un valor inicial elevado, de forma tal que al ir disminuyendo gradualmente esta temperatura los pesos de las conexiones toman valores que generan mínimos globales en la curva de energía, evitando el ser atrapado en mínimos locales.



2.7 Arquitectura de la red de Boltzmann

La máquina está compuesta por tres capas de unidades con función de activación binaria, distribuidas en cascada como se muestra en la figura 2.7. Las unidades deben

estar siempre en uno de sus dos posibles estados y conectadas entre sí por lazos cuyos pesos son valores reales positivos o negativos.

Las unidades en la red de Boltzmann se dividen en dos grupos funcionales, un conjunto de unidades visibles, que son la interface con el ambiente, y un conjunto de unidades intermedias o escondidas. Durante el entrenamiento todas las unidades visibles son forzadas a tomar estados específicos debido a la presencia de patrones. Se asume, para su funcionamiento, que los patrones de entrada persistirán por el tiempo suficiente para que el sistema encuentre la configuración de energía mínima compatible con la entrada.

A cada estado global de la red se le asigna una energía, que con las correctas consideraciones, puede ser minimizada con la participación individual de las unidades. La energía de una configuración global se define como:

$$E = - \sum_{i<j} w_{ij} u_i u_j + \sum \theta_i u_i$$

Donde w_{ij} es el peso de las conexiones entre las unidades, u_i es uno si la i-ésima unidad está prendida y cero si está apagada, θ_i es el umbral correspondiente a la i-ésima unidad.

Debido a que las conexiones son simétricas, la diferencia entre la energía de todo el sistema con el estado de la i-ésima unidad prendida y su energía con la i-ésima unidad apagada puede ser determinada localmente por la misma unidad, el pozo de energía generado es calculado como:

$$\Delta E_i = \sum_j w_{ij} u_j - \theta_i$$

El término del umbral puede ser eliminado considerando la siguiente observación: El efecto de θ_i en la energía global correspondiente a una unidad tiene el mismo efecto que un lazo con peso $-\theta_i$ entre la i-ésima unidad y una unidad especial que por definición se encuentra siempre prendida. Esta unidad no tiene ningún sentido físico pero simplifica los cálculos. El término $-\theta_i$ es comúnmente llamado el bias de una unidad. Tomando ésta consideración, las ecuaciones anteriores pueden ser escritas como:

$$E = - \sum_{i<j} w_{ij} u_i u_j$$

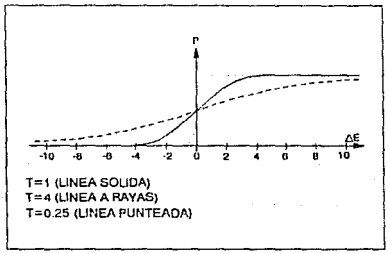
$$\Delta E_i = \sum_j w_{ij} u_j$$

Para evitar caer en mínimos de energía locales existe un mecanismo que permite hacer saltos ocasionales a configuraciones de mayor energía, una forma del algoritmo Metrópolis válido para el paralelismo: Si el pozo de energía entre los estados prendido y

apagado de la i-ésima unidad es $\triangle E_i$, entonces, sin importar el estado previo, el nuevo estado será prendido, $s_i = 1$, con probabilidad:

$$P_k = \frac{1}{(1 + e^{-\triangle Ei/T})}$$

Donde T es el parámetro que actúa como la temperatura. Cuando T es muy grande, $\triangle E_i/T$ es un número relativamente pequeño y $e^{-\triangle Ei/T}$ es aproximadamente 1, lo que significa que P(0) es cerca de 0.5, o que la mitad de las neuronas se encuentran en un estado alto de energía. Si T decrece hasta un número relativamente pequeño comparado con $\triangle E_i$, entonces $e^{-\triangle Ei/T}$ es muy cercano al cero, lo que significa que a bajas temperaturas la P(0) es casi 1, o que la mayor parte de las neuronas se encuentran en un estado de baja energía. En la figura 2.8 se puede apreciar la función de probabilidad que define el estado de las neuronas en las máquinas de Boltzmann.



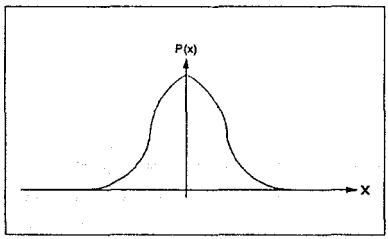
2.8 Probabilidad para el cambio de estado de las neuronas

La probabilidad relativa de dos estados de energía globales, en un sistema que realiza temple simulado para encontrar eventualmente el equilibrio térmico, sigue una distribución Boltzmann [figura 2.9]:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T}$$

Donde P_α es la probabilidad de encontrarse en el α-ésimo estado global y E_α es la energía correspondiente a ese estado.

Se dice que la red tiene un modelo perfecto del ambiente si ésta proporciona exactamente la misma distribución de probabilidad sobre todos los estados de las unidades visibles cuando opera en equilibrio térmico sin patrones presentes.



2.9 Distribución Boltzmann

Debido a que la energía es una función lineal de los pesos, es posible establecer un relación entre las probabilidades logarítmicas de un estado global y el peso de las conexiones individuales (Ackley, Hinton y Sejnowski, 1985) :

$$\frac{\delta \ln P\alpha}{\delta w_{ij}} = \frac{1}{T} - [u^{\alpha}_{i} u^{\alpha}_{j} - p'_{ij}]$$

Donde u^{α}_{i} es el estado de la i-ésima unidad en el α-ésimo estado global y p'_{ij} es la probabilidad de encontrar las unidades i y j prendidas al mismo tiempo cuando el sistema está en equilibrio.

Una medida de discrepancia entre el modelo interno de la red y el ambiente es:

$$G = \Sigma\ P(V_{\alpha}) \ln \frac{P(V_{\alpha})}{P'(V_{\alpha})}$$

Donde $P(V_{\alpha})$ es la probabilidad del α-ésimo estado de las unidades visibles cuando sus estados son determinados por el ambiente y $P'(V_{\alpha})$ es la probabilidad correspondiente cuando la red corre libremente sin ningún patrón presente. G, también conocida como divergencia asimétrica, es una medida de la distancia de la distribución dada por $P'(V_{\alpha})$ a la distribución dada por $P(V_{\alpha})$. G es cero si las distribuciones son idénticas, de cualquier otra forma es positiva.

Como $P'(V_{\alpha})$ depende de los pesos, G puede ser alterada si éstos se cambian. Para lograr un gradiente descendente en G, es necesario conocer la derivada parcial de G con respecto a cada uno de los pesos.

Las probabilidades de los estados globales son determinadas por las energías y las energías se determinan a partir de los pesos. Con estas consideraciones la derivada parcial de G es:

$$G = -\frac{1}{T}(p_{ij} - p'_{ij})$$

Donde p_{ij} es el promedio de probabilidad de dos unidades prendidas cuando el ambiente fuerza a las unidades visibles a tomar estados específicos.

Para minimizar G basta con observar p_{ij} y p'_{ij} cuando la red se encuentra en equilibrio térmico, y cambiar cada peso con una cantidad proporcional a la diferencia entre éstas dos probabilidades:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(p_{ij} - p'_{ij})$$

ENTRENAMIENTO

En el entrenamiento de la red se reconocen dos fases, a las que los autores han llamado positiva y negativa.

En la fase positiva los patrones de entrada y salida definen el estado de las unidades de las capas respectivas. Para cada par de patrones se permite a la red encontrar el estado de las unidades intermedias que hacen mínima la energía para distintas temperaturas. Cuando la temperatura es mínima el estado global de la red es óptimo, y la actividad de todas las neuronas es almacenada.

En la fase negativa solo el patrón de entrada define el estado de las unidades de entrada, el resto de las unidades toman aleatoriamente valores binarios. Nuevamente, disminuyendo la temperatura, se permite a la red encontrar el estado global de energía mínima para cada patrón, cambiando de ser necesario el valor de las unidades de las capas intermedia y salida. Cuando la temperatura es mínima se almacena la actividad de todas las neuronas.

Cuando se terminan ambas fases se hace un recuento, para cada peso, de las veces en que dos unidades estuvieron activas simultáneamente y se modifica el valor de los pesos. El proceso se repite hasta que la salida para cada patrón de entrada es la deseada.

ALGORITMO DE APRENDIZAJE

FASE POSITIVA	FASE NEGATIVA
1. Asignar temperatura inicial T	1.Asignar temperatura inicial T
2.Acutalizar las neuronas de entrada y salida con los patrones respectivos.	2.Actualizar las neuronas de entrada con el patrón respectivo.
3.Asignar a las neuronas de la la capa intermedia estados aleatorios.	3.Asignar a las neuronas de la la capa intermedia y salida estados aleatorios.
4.Escoger aleatoriamente una neurona de la capa intermedia.	4.Escoger aleatoriamente una neurona de la capa intermedia o de salida.
5.Calcular la contribución de energía de esa neurona: $\Delta E_i = \Sigma\, u_j w_{ij}$	5.Calcular la contribución de energía de esa neurona: $\Delta E_i = \Sigma\, u_j w_{ij}$
6.Si $\Delta E_i > 0$ entonces $u_i=1$, sino si se cumple que: $\frac{1}{1 + e^{-\Delta Ei/T}} >$ num. aleatorio entonces $u_i=1$ sino $u_i=0$.	6.Si $\Delta E_i > 0$ entonces $u_i=1$, sino si se cumple que: $\frac{1}{1 + e^{-\Delta Ei/T}} >$ num. aleatorio entonces $u_i=1$ sino $u_i=0$.
7.Repetir desde el paso 4 cuantas veces se establezca.	7.Repetir desde el paso 4 cuantas veces se establezca.
8.Reducir la temperatura.	8.Reducir la temperatura.
9.Repetir desde el paso 4 hasta que la temperatura llegue al mínimo.	9.Repetir desde el paso 4 hasta que la temperatura llegue al mínimo.
10.Almacenar el estado de las neuronas de la capa intermedia.	10.Almacenar el estado de las neuronas de la capa intermedia y salida.
11.Repetir desde el paso 1 para cada patrón.	11.Repetir desde el paso 1 para cada patrón.

ALGORITMO

1.- Inicializar los pesos de la red con valores aleatorios entre [-1,1].

2.- Realizar la fase positiva.

3.- Realizar la fase negativa.

4.- Cambiar los pesos de la red de acuerdo a:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(p_{ij} - p'_{ij})$$

$$p_{ij} = \sum_k u^k_i\, u^k_j \quad k=1,2,\ldots,\text{número de patrones.}$$

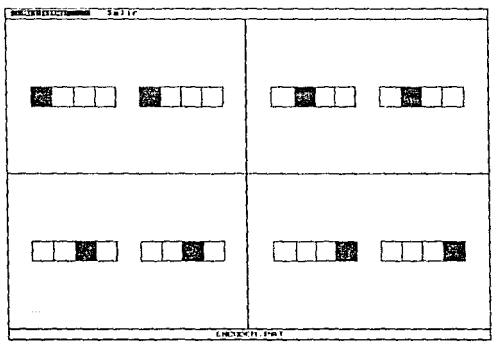
Y u_i son los estados de las neuronas almacenados en la fase positiva. Para p'_{ij} se utilizan los estados almacenados en la fase negativa.

5.- Repetir desde el paso 1 hasta que la red entregue a su salida los valores deseados.

EJEMPLO

Para ilustrar el algoritmo de máquina de Boltzmann se ha elegido el ejemplo propuesto por los autores en su artículo "A learning Algorithm for Boltzmann Machine".

El ejemplo es llamado Codificador 4-2-4. La red empleada tiene 4 neuronas de entrada, 2 neuronas intermedias y 4 neuronas de salida [figura 2.11]. El conjunto de patrones esta conformado por cuatro patrones reticulares. Como se puede observar en la figura 2.10, para cada par de patrones sólo esta prendido un elemento tanto en el patrón de entrada como en el de salida.



2.10 Conjunto de patrones ENCODER.PAT

La tarea de la red, es la de generar un código binario en las neuronas de la capa intermedia para cada par de patrones. De forma tal que al mostrar un patrón de entrada, la red lo codifique en las neuronas de la capa intermedia y, después, lo decodifique en la capa de salida. Obteniendo así, el mismo patrón de entrada.

El sistema de temple que debe seguir la red para el entrenamiento, es necesario establecerlo en los parámetros de la red. Los parámetros necesarios son:

a) Intervalo de los pesos: 0.

Este parámetro se utiliza exclusivamente cuando la red ha sido apenas creada y sirve para establecer en que intervalo de inicialización deben de estar los pesos para que comience el entrenamiento.

b) Alfa: 0.1

Esta constante representa el factor de aprendizaje que deberá de regular el entrenamiento de la red. Cada incremento o decremento en los pesos es multiplicado por esta constante.

c) Temperatura inicial: 20.

El sistema de temple involucra tres parámetros. El primero, la temperatura inicial, representa el punto de partida para que la red encuentre el estado de energía mínima.

d) Iteraciones: 5.

El segundo parámetro involucrado en el sistema de temple es el número de iteraciones que se deben realizar para que la red decremente la temperatura hasta la temperatura final.

e) Temperatura final: 10.

El tercer parámetro para el sistema de temple, representa la temperatura en la que serán tomadas las estadísticas para el cambio de los pesos.

f) Temples: 2.

Esta variable indica el número de intentos que tendrán las neuronas para establecer la configuración de mínima energía en cada iteración realizada antes de llegar a la temperatura mínima.

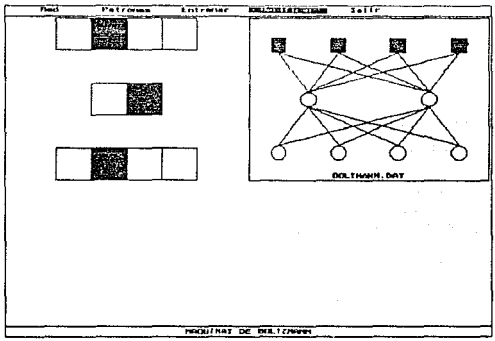
g) Estadísticas: 10.

Una vez que la temperatura ha sido disminuida al mínimo, la red debe de encontrar la configuración de energía mínima de acuerdo a la actividad de las neuronas y los pesos de las conexiones. El parámetro de estadísticas representa el número de veces que deberá ser almacenada la configuración de actividad de la red en distintos casos de energía mínima para el futuro cambio de pesos.

Las máquinas de Boltzmann no siempre son eficientes y, por consecuencia, el entrenamiento suele ser muy prolongado. En este ejemplo, al ser pocas las neuronas involucradas y pocos los patrones, el aprendizaje es generalmente rápido. No obstante, si se incrementa el número de patrones y el número de neuronas, la red puede tardar una eternidad en aprenderse el conjunto de patrones (¡ Si es que lo logra !).

En la figura 2.11 se muestra la pantalla generada al verificar el aprendizaje. En el caso de las Máquinas de Boltzmann, la pantalla de entrenamiento y la de verificación son prácticamente las mismas.

La arquitectura de la red, formada por tres capas, se muestra en la parte superior derecha de la pantalla. La capa de entrada, cuyas unidades son representadas con rectángulos, es la capa más superior; la capa de salida es la más inferior.



2.11 Verificación del aprendizaje

En la parte izquierda de la pantalla se muestra el patrón de entrada seleccionado para el ejemplo. Las neuronas de la capa intermedia, representadas como una hilera de rectángulos, están por debajo del patrón de entrada. Debido a que la actividad de las neuronas en una red entrenada por máquina de Boltzmann es del tipo binario, las neuronas de la capa intermedia son tratadas igual que los patrones binarios de entrada. La salida generada por la red se encuentra abajo de las neuronas de la capa intermedia.

Durante el entrenamiento aparece por arriba del patrón de entrada una P junto con un signo de adición "+" o un signo de sustracción "-", que indican la fase que se esta llevando al cabo.

P+ es la fase positiva, en la que las neuronas de entrada y de salida son forzadas a tomar, como nivel de activación, su valor correspondiente en el patrón de entrada y en el de salida respectivamente. Sólo las neuronas de la capa intermedia tratan de encontrar la configuración de energía mínima para el par de patrones.

P- indica la fase negativa, en la que solamente las neuronas de la capa de entrada son forzadas a tomar, como nivel de activación, su valor correspondiente en el patrón de entrada. Las neuronas de la capa intermedia y de la capa de salida tratan de establecer la configuración de energía mínima para el patrón de entrada.

BIBLIOGRAFIA

1.- Kirkpatrick S., Gellat C. D. Jr, Vechi M. P.

Optimization by simulated annealing.

Science Vol.220, pp 671-680, 1983.

2.- D. H. Ackley, J. E. Hinton, T. J. Sejnowsky.

A Learning Algorithm for Boltzmann Machines.

Cognitive Science, Vol 19, pp 147-169, 1985.

3.- J. E. Hinton, T. J. Sejnowsky.

Learning and relearning in Boltzmann Machines.

Parallel Distributed Processing.

Volumen 1: Foundations, Cambridge: MIT press, 1986.

4.- Fure-Ching Jeng, John W. Woods.

Simulated Annealing in Compound Gaussian Random Fields.

IEEE Transactions on Information Theory, Vol 35, No.1, pp 94-107, Enero de 1990.

2.3 MODELOS PARA PATRONES CONTINUOS

2.3.1 RETROPROPAGACION

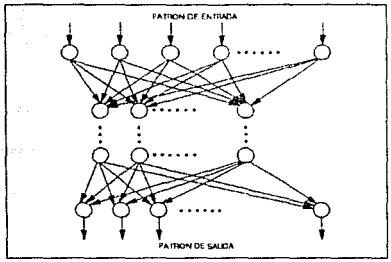
RUMELHART Y MCCLELLAND

Retropropagación es el método de aprendizaje para redes neuronales más ampliamente utilizado y sus posibles aplicaciones abarcan un gran número de áreas. Retropropagación es uno de los métodos más fáciles de entender, sus procedimientos de aprendizaje y actualización son intuitivamente atractivos porque están basados en un concepto simple: Si la red entrega una respuesta equivocada, entonces los pesos se corrigen de tal manera que el error disminuye y las respuestas posteriores de la red son más acertadas.

Retropropagación puede atacar cualquier problema que requiera del mapeo de patrones. Ha sido utilizado en el diagnóstico médico para asociar un cuadro clínico o una serie de síntomas a enfermedades particulares, y también en la detección de arritmias, taquicardias y otras anomalías cardiovasculares en la lectura de electrocardiogramas. En robótica se ha empleado sobre todo en áreas de control, para manejar los movimientos de brazos mecánicos o dar seguimiento a trayectorias espacio-temporales. También ha sido utilizado en ambientes difíciles de imaginar como en las predicciones en los mercados bursátiles, en la elaboración de apuestas en football americano profesional, en el reconocimiento de códigos postales, etc.

La base del concepto de retropropagación fue presentada en un principio por Paul Werbos en 1974, después fue reinventada independientemente por David Parker en 1982, y publicada en 1986 por Rumelhart y McClelland. Retropropagación encuentra su ancestro en el "perceptrón", sin embargo, la diferencia entre los modelos es enorme: El perceptrón estaba limitado solamente a dos capas de procesadores y sólo una capa de pesos adaptivos, además, el perceptrón sólo puede aprender patrones que son linealmente separables.

Normalmente, retropropagación utiliza redes neuronales de tres o más capas de unidades. La arquitectura de estas redes es muy común, sólo la capa de entrada recibe entradas externas, no hay limitantes en el número de capas intermedias y sólo hay una capa de salida. Cada unidad de las capas intermedias está conectada con todas las unidades de las capas inferior y superior a ella. En la figura 2.12 se puede apreciar la arquitectura clásica de este tipo de redes, las unidades no se pueden conectar con otras en la misma capa, pero las conexiones si pueden brincar capas intermedias.



2.12 Arquitectura de una red de retropropagación

Las redes de retropropagación no deben de estar necesariamente interconectadas en su totalidad, sin embargo, muchas aplicaciones han tenido buenos resultados con las capas totalmente interconectadas.

Los pesos de la unidad i a la unidad j serán denotados como w_{ij}. Durante el entrenamiento de la red, los pesos asociados a cada interconexión son ajustados. Al concluir el entrenamiento, el valor final de los pesos hace que el error total disminuya, por lo que la red recuerda mejor las asociaciones entrada-salida. Pero no sólo recuerda dichas asociaciones, sino que existe generalización.

El algoritmo de entrenamiento por retropropagación tiene básicamente dos pasos:

PROPAGACION HACIA ADELANTE.

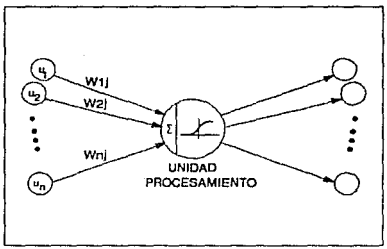
Comienza con la presentación de un patrón de entrada a la red. En las redes entrenadas por retropropagación las unidades de entrada no son ciertamente procesadores, éstas toman el valor del vector de entrada y lo hacen pasar a la siguiente capa. Una vez presentado el patrón, los niveles de activación en la capa de entrada son propagados hacia las capas intermedias hasta la capa de salida, definiendo así los niveles de activación de las unidades. Cada unidad realiza la suma de todas sus entradas y después la hace pasar por una función no-lineal para generar el nivel de activación. La entrada a una unidad queda definida entonces como:

$$s_j = \sum_i u_i w_{ij}$$

Donde u_i es el nivel de activación de la unidad i, y w_{ij} es el peso correspondiente de la unidad i a la unidad j. La función empleada para definir el nivel de activación utilizada con mayor frecuencia es la función sigmoidal, expresada como:

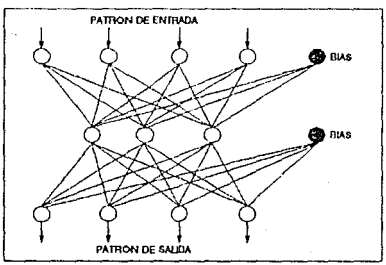
$$u_j = \frac{1}{1 + e^{-sj}}$$

En la figura 2.13 se muestra una neurona con la función antes mencionada.



2.13 Neurona utilizada por retropropagación

Cabe aclarar que se puede emplear cualquier tipo de función no-lineal, que cumpla con ser derivable, continua y acotada, para generar el nivel de activación. Este valor es pasado entonces a través de las interconexiones de salida de la unidad.



2.14 Red 4-3-4 con unidades de Bias

Para proporcionar una convergencia más rápida a la red se incluye un umbral, generalmente conocido como "bias", a cada neurona. Como se puede observar en la figura 2.14, se implementa con una neurona más en cada capa, excepto en la capa de salida, que está conectada a todas las neuronas de la capa inmediata posterior y cuyo nivel de activación es por definición uno.

PROPAGACION HACIA ATRAS.

La propagación hacia atrás, comienza con la comparación de la salida de la red con el patrón deseado, generando así un error para cada una de las unidades de salida. Este error se retropropaga sobre las capas intermedias hasta la capa de entrada y es calculado para todas las unidades. Los pesos son modificados en cantidades proporcionales al error generado con la finalidad de disminuirlo. Por esta razón, retropropagación es un método de gradiente descendente. Cada unidad tiene entonces una cantidad de error asociado generalmente denotado como δ.

La variable δ para las unidades de salida se calcula de la siguiente manera:

$$\delta j = (y_j - u_j) f'(S_j)$$

Donde y_j es el valor del patrón deseado para la unidad j, u_j es el nivel de activación de la unidad j y f'(x) es la derivada de la función no lineal.

La cantidad de error está dada por $(y_j - u_j)$. La derivada de la función sigmoidal escala el error para obligar a una corrección más fuerte cuando la suma S_j está cerca del nivel del umbral. Para cualquier unidad de las capas intermedias la δ se calcula como:

$$\delta_j = \left[\sum_k \delta_k w_{kj} \right] f'(S_j)$$

En este caso, se realiza la suma de las δ para todas las unidades que reciben salida de la unidad j. La derivada de la función sigmoidal tiene la misma función que en el caso anterior. Para simplificar los cálculos, puede demostrarse que la derivada de la función sigmoidal cumple con:

$$f'(S_j) = (1 - f(S_j)) f(S_j)$$

El ajuste de los pesos se realiza mediante los valores de las δ para cada unidad. Cada peso se ajusta considerando el valor de la δ de la unidad que recibe entrada de esa interconexión. El ajuste se lleva al cabo de la siguiente manera:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j u_i$$

Esta ecuación de ajuste es conocida como la regla Delta generalizada. La cantidad de ajuste en los pesos es proporcional a δ_j, el valor del error para la unidad objetivo, y a u_i, el valor de activación de la unidad que genera dicho error. Esto implica que un error muy grande para la unidad j se refleja en un fuerte ajuste para sus pesos de entrada.

La variable α es conocida como el factor de aprendizaje, sus valores fluctúan, comúnmente, entre 0.25 y 0.75. El valor de α es escogido a priori y refleja la rapidez de aprendizaje de la red. Si α toma valores muy elevados puede provocar inestabilidad en la red y por consiguiente un aprendizaje deficiente.

El método no converge tan rápidamente como los métodos que utilizan segundas derivadas, sin embargo, es más sencillo y puede ser fácilmente implementado en hardware. El método puede ser significativamente mejorado, sin sacrificar su simplicidad, utilizando un método de aceleración en el cual el gradiente es utilizado para aumentar la velocidad del aprendizaje, el término que se agrega a la función de ajuste de pesos es conocido como momento. El ajuste de pesos queda definido entonces como:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_j u_i + \beta(w_{ij}(t) - w_{ij}(t-1))$$

Donde β es un factor de caida exponencial que determina la contribución relativa del gradiente actual y los siguientes al cambio de los pesos. Los valores de β están entre cero y uno.

Para verificar el grado de aprendizaje de la red se calcula el mínimo error cuadrático, que resulta ser la función de costo a minimizar por el algoritmo:

$$E = \frac{1}{2} \sum_j \sum_i (y_{ji} - u_{ji})^2$$

Donde u_{ji} es el nivel de activación de la i-ésima unidad para el j-ésimo patrón presentado, y y_{ji} es la i-ésima salida deseada correspondiente al j-ésimo patrón.

Para que la red tenga un buen aprendizaje y sea capaz de generalizar es necesario mostrarle repetidamente una gran cantidad de ejemplos con distintas versiones de cada tipo o clase. El entrenamiento de la red se puede hacer de dos formas distintas. La primera es conocida como "entrenamiento por épocas", en la cual se presenta consecutivamente el conjunto de patrones y el ajuste de pesos se realiza después de haber presentado todos los patrones. La segunda manera es conocida como "entrenamiento continuo", en la cual la correción proporcional de los pesos se efectúa cada vez que es mostrado un patrón.

Cuando el mínimo error cuadrático es próximo al cero se dice que la red ha convergido para el conjunto de patrones presentados. La convergencia no es una

garantía en retropropagación, el proceso de aprendizaje toma por lo general bastante tiempo y la red puede caer en mínimos locales y el aprendizaje se detiene.

Existen una serie de técnicas para evitar que la red quede atrapada en un mínimo local, cambiar la escala de aprendizaje o cambiar el número de unidades en las capas intermedias es suficiente, sin embargo, la técnica más efectiva es hacer que la red haga un salto de una configuración, con mínimo local, a otra agregando valores aleatorios a los pesos.

ALGORITMO DE APRENDIZAJE.

1.- Inicializar los pesos de la red con valores aleatorios.

2.- Mostrar un patrón de entrada.

3.- Obtener la salida de la red. La actividad de las neuronas de la capa de entrada se define como el patrón de entrada. La actividad de las neuronas de las capas intermedias y de salida se define como:

$$u_i = \frac{1}{1 + e^{-s_j}}$$

Donde $s_j = \sum_k u_k w_{kj}$

4.- Actualizar los pesos de la red de acuerdo a:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j u_i + \beta(w_{ij}(t) - w_{ij}(t-1))$$

Si la j-ésima neurona pertenece a la capa de salida entonces:

$$\delta_j = u_j(1-u_j)(y_j-u_j)$$

Donde y_j representa al patrón de salida deseado.

Si la j-ésima neurona pertenece a una capa intermedia entonces:

$$\delta_j = \left[\sum_k \delta_k w_{kj} \right] f'(s_j)$$

El índice k se refiere a las neuronas inmediatas posteriores a la capa que refiere el índice j.

5.- Repetir desde el paso 2 para cada uno de los patrones.

6.- Calcular el error cuadrático:

$$E=\tfrac{1}{2} \sum_j \sum_i (y_{ji} - u_{ji})^2$$

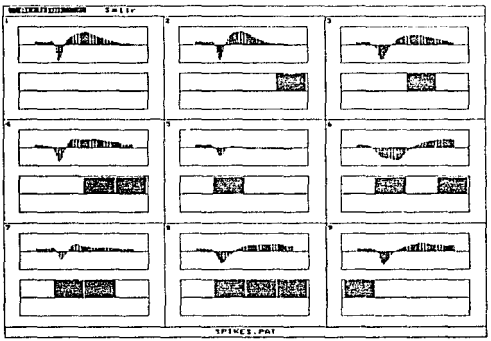
El índice j representa a cada uno de los patrones de salida.

Si $E<<0.5$ el entrenamiento termina, en caso contrario repetir desde el paso 2.

EJEMPLO

Para ilustrar el algoritmo de retropropagación se ha elegido un conjunto de 9 señales bioeléctricas, potenciales de acción, obtenidas en un experimento de laboratorio y transformadas a números en código ASCCI mediante un convertidor.

La intención del ejemplo es que la red asocie a cada patrón de entrada un número binario de 4 posiciones. En la figura 2.15 se muestra el archivo de patrones empleado.



2.15 Archivo de patrones SPIKES.DAT

La red utilizada tiene 64 neuronas de entrada, 12 neuronas intermedias y 4 neuronas de salida. En el simulador, durante el entrenamiento de una red con el algoritmo de retropropagación, se muestra en la parte superior derecha de la pantalla la arquitectura de la red (véase figura 2.16).

Las redes que pueden entrenarse por retropropagación tienen topologías de dos o más capas de neuronas conectadas en cascada, en el simulador se pueden definir arquitecturas hasta de cinco capas. En la representación gráfica la capa de entrada, cuyos procesadores se muestran en forma de cuadros, se encuentra en la parte superior. La capa de salida es la más inferior.

Con la finalidad de medir el aprendizaje, se dispone de una gráfica del error cuadrático producido por la red. El eje vertical es la magnitud del error y en el eje horizontal se manejan las épocas que se han realizado. Una época se refiere a una

presentación del conjunto de patrones a la red. La gráfica del error se muestra en la parte superior izquierda de la pantalla (véase figura 2.16).

Los parámetros que se deben establecer para el entrenamiento por retropropagación son:

a) Intervalo de los pesos: ± 0.7

   Es el valor que pondera la asignación aleatoria de los pesos antes de comenzar el entrenamiento.

b) Alfa: 0.5

   Es el factor de aprendizaje, y sirve para regular el cambio de los pesos cuando se retropropaga el error de salida.

c) Momento: 0.3

   Esta constante, cuyos valores van del 0 al 1, es utilizada para aumentar la rapidez del aprendizaje una vez que se ha encontrado el gradiente descendente.

d) Número de iteraciones: 500

   Se refiere al número de veces que deberá ser mostrado el conjunto de patrones a la red. El entrenamiento se detiene cuando las iteraciones realizadas superan el número definido o cuando el error cuadrático de la red es menor que el establecido.

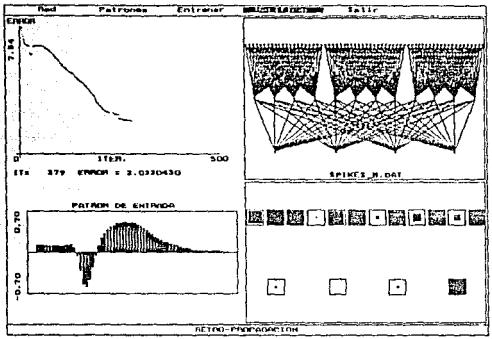
e) Error por patrón: 0.0001

   Esta cantidad representa al error cuadrático máximo que debe de entregar la red para un patrón de entrada. Si el error producido es mayor al que se establece por patrón, se retropropaga el error, en caso contrario se puede afirmar que la red responde correctamente al patrón de entrada.

f) Error total: 0.01

   Se refiere al error cuadrático máximo que debe de tener la red en respuesta al conjunto de patrones para considerar que ya han sido aprendidos.

Cuando se hace una verificación del aprendizaje, se muestra el patrón de entrada temporal en la parte inferior izquierda de la pantalla y un diagrama de Hinton de la actividad neuronal, en las capas inferiores a la capa de entrada, en la parte inferior derecha de la pantalla [Figura 2.16].



2.16 Verificación del aprendizaje en retropropagación

En el diagrama de Hinton, las neuronas están representadas por rectángulos, mismos que están rellenos en forma proporcional a la actividad de las neuronas, es decir, si la neurona tiene actividad nula el cuadro estará vacío y si tiene actividad máxima el cuadro estará completamente lleno. La solución o clasificación del patrón presentado a la red es la capa de rectángulos más inferior del diagrama de Hinton.

BIBLIOGRAFIA

1.- William P. Jones, Josiah Hoskins.
Back-Propagation. A generalized delta learning rule.
BYTE, pp.155-162, Octubre 1987.

2.- David E. Rumelhart, Geoffrey Hinton, Ronald Williams.
Learning representation by back-propagation errors.
Nature, Vol. 323, pp. 533-536, 1986.

3.- Yoh-Han Pao
Adaptive Pattern Recognition and Neural Networks.
Addison Wesley, 1989.

2.3.2 MEMORIA ASOCIATIVA HOLOGRAFICA

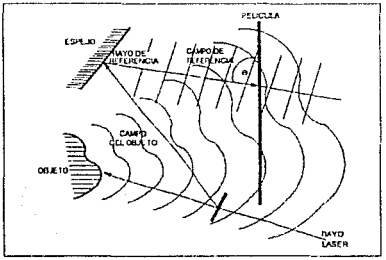
YOH-HAN PAO

En la tecnología de información, las memorias holográficas han sido desarrolladas para el almacenamiento de grandes masas de información contenidas en medios tales como fotografías, imágenes de video, etc. En algunas aplicaciones especiales ofrecen la gran ventaja de ser tolerantes a daños parciales del medio de almacenamiento.

Se ha propuesto, pero jamás demostrado por alguna simulación, que la memoria distribuida en el sistema neuronal puede estar basado en la holografía. Sin embargo, existe una gran cantidad de inconvenientes para la aplicación de este principio en la neurofisiología, por ejemplo, se requiere de un medio óptico, frentes de onda coherentes y ondas de referencia por medio de las cuales la información almacenada pueda ser decodificada.

Se puede decir que la hipótesis acerca de la holografía como el principio biológico de memoria fue propuesto cuando aún no se conocían otros principios que también permiten la distribución espacial de información, como son los modelos adaptivos de redes neuronales artificiales.

La explicación holográfica de la memoria biológica tiene como fundamento el hecho de que la información puede ser recuperada aún cuando el patrón presentado estuviese trasladado de la posición en que fue aprendido. Se sabe que las respuestas y percepciones del ser humano y de los animales tienen una invariancia limitada con respecto al tamaño, rotación y forma de los patrones de estímulo.



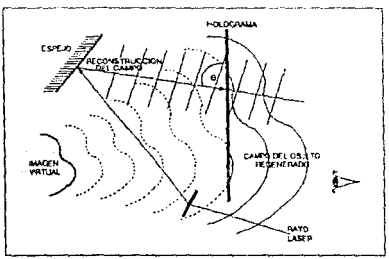
2.17 Formación de un holograma convencional

La teoría que sustenta a la memoria asociativa holográfica se basa en la formación de hologramas tradicionales. En un holograma convencional, un objeto es almacenado en una película fotográfica mediante la interferencia de un campo de referencia y un campo generado al hacer incidir un rayo en el objeto [Figura 2.17].

Para almacenar varios objetos en la misma película es necesario que cada uno de éstos tenga su propio campo de referencia. Todos los campos de referencia son ondas planas de la misma longitud de onda , la diferencia que existe entre éstos es que el ángulo de incidencia con respecto al plano de la película es distinto.

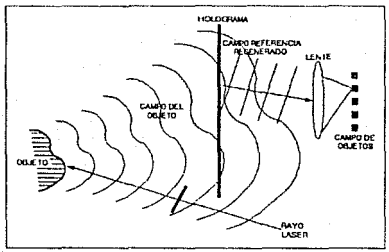
Los campos en todos los casos son espacialmente coherentes, es decir, sus fases relativas son constantes. Los objetos son almacenados, junto con sus campos de referencia individualmente, y la película fotográfica es revelada hasta que han sido almacenados todos.

Una vez generado el holograma basta con iluminarlo con un rayo de referencia que reconstruya el campo de referencia asociado a un objeto para que el holograma genere una imagen virtual del objeto [Figura 2.18].



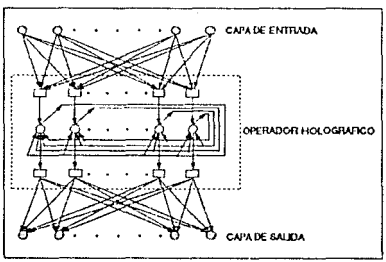
2.18 Formación virtual de un objeto

El proceso inverso de este principio es utilizado en las memorias asociativas holográficas. Si el holograma es iluminado con un campo formado a partir de un objeto, entonces se crea el campo de referencia asociado y mediante el uso de una lente se puede identificar cual de los campos almacenados en el holograma ha sido generado a través de un campo de objetos incidente [Figura 2.19].



2.19 Recuperación de un objeto almacenado en un holograma

La memoria asociativa holográfica no es considerada una red neuronal propiamente debido a las operaciones matemáticas que involucra el operador holográfico. En la figura 2.20 se muestra la arquitectura de una red neuronal que aproxima a dicho operador. Está compuesta por una capa de entrada cuya única función es la de presentar el patrón de entrada al operador holográfico mediante los pesos conectados a éste. En la capa superior del operador se realiza la proyección de cada uno de los patrones almacenados. En la capa intermedia del operador se lleva al cabo inhibición lateral (Apéndice A) para obtener la mejor proyección, es decir, el patrón almacenado que mayor parecido tiene con el de entrada. En la última capa del operador se obtiene lo que correspondería al campo de referencia asociado, y finalmente, en la capa de salida de la red se obtiene el patrón completo asociado al patrón de entrada.



2.20 Arquitectura de una red tipo memoria holográfica

El operador holográfico debe cumplir con las dos características centrales de los hologramas con respecto a la distribución espacial de trazas de memoria. Estas son:

a) Los elementos en el conjunto de datos están distribuidos en el área de memoria mediante una transformación.

b) Un gran conjunto de datos son sobrepuestos en el mismo medio de forma distribuida.

Si se define un conjunto de patrones de entrada como los vectores:

$$X_i = (x_1, x_2, x_3, \ldots\ldots, x_n)$$

y un conjunto de patrones de salida como los vectores:

$$Y_i = (y_1, y_2, y_3, \ldots\ldots, y_n)$$

entonces el operador holográfico, que cumple con las características centrales de un holograma, está conformado por dos matrices de correlación (matrices de pesos), M_1 y M_2 generadas de la siguiente manera:

$$M_1 = \sum_{i=1}^{m} X_i^t Z_i$$

$$M_2 = \sum_{i=1}^{m} Z_i^t Y_i$$

Donde m es el número total de patrones y Z_i es un conjunto de vectores ortonormales que definen un espacio euclidiano L-dimensional (véase Apéndice B).

MODO DE OPERACION.

Una vez que el operador holográfico contiene la información de todo el conjunto de patrones entrada-salida y se desea reconocer un patrón de entrada determinado, se debe realizar el producto:

$$X_k M_1 = [X_k X_k^t] \cdot Z_k + \sum_{i \neq k}^{m} [X_k X_i] \cdot Z_i$$

Basados en el principio de recuperación de un holograma, se necesita formar una operación de filtro o proyección que descomponga $X_k M_1$ en sus componentes a lo largo de los ejes Z_i y que obtenga el índice del vector que tenga mayor magnitud:

$$\max_{l} \{ [Z_l Z_k][X_k X_k] + \sum_{i \neq k}^{m} [Z_l Z_i][X_k X_i] \}$$

Donde l es el índice del vector con el cual se obtiene la mayor magnitud.

Si se asume que $[X'_k X'_k] > [X_k X_i]$ para toda $i \neq k$, entonces se recupera el valor correcto del índice k y se genera el vector Z_k. Después se presenta solamente el vector Z_k a la matriz M_2 y se recupera completamente el vector de salida asociado:

$$Z_k M_2 = Z_k \sum_{i=1}^{m} Z_i Y_i = [Z_k Z_k] Y_k = Y_k$$

ALGORITMO DE APRENDIZAJE.

1.- Definir las matrices de pesos de la siguiente manera:

$$M_1 = \sum_{i=1}^{m} X_i^t Z_i$$

$$M_2 = \sum_{i=1}^{m} Z_i^t Y_i$$

Donde m es el número total de patrones y Z_i es un conjunto de vectores ortonormales que definen un espacio euclidiano L-dimensional (véase Apéndice B).

2.- Mostrar un patrón de entrada X.

$$U = X M_1$$

3.- Obtener el índice máximo m de:

$$m = \max_{l} \{ [Z_l U] \}$$

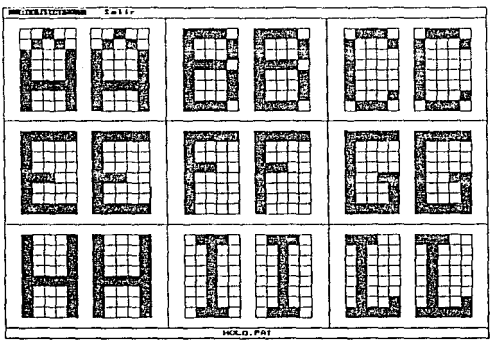
Donde l=0,1,.....,L.

4.- La salida se define entonces como:

$$Y = Z_m M_2$$

EJEMPLO

Para ilustrar el método de memoria asociativa holográfica se escogió un ejemplo con el cual se pueda apreciar la tarea de reconstrucción que se realiza generalmente en la holografía. Para éstos fines se generó un archivo de patrones, mostrado en la figura 2.21, que contiene algunas de las letras del abecedario. Las letras, formadas en retículas, aparecen por pares de entrada/salida.



2.21 Archivo de patrones de entrenamiento HOLO.PAT

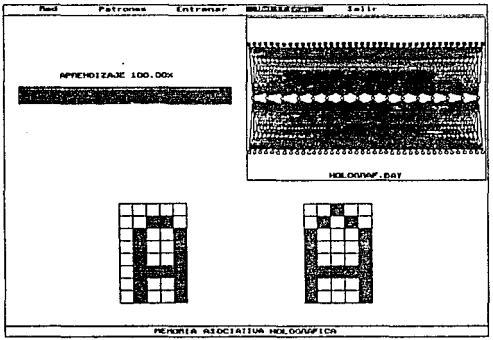
Con este ejemplo se pretende que la red, una vez que ha aprendido el conjunto de patrones, sea capaz de reconstruir cualquier letra a partir de versiones ruidosas de las mismas.

Al realizar un entrenamiento con una red tipo memoria holográfica en el simulador, la arquitectura mostrada en la esquina superior derecha de la pantalla incluye una capa de entrada, una intermedia y una capa de salida. Por razones de espacio no fue posible representar el operador holográfico completamente; aparece en su lugar una capa cuyo número de nodos depende de las dimensiones del espacio Euclidiano definido generado.

El algoritmo para las memorias holográficas no es adaptivo, la fase de entrenamiento se realiza en un sólo paso, es decir, los pesos de las conexiones se establecen al generar las matrices de correlación (matrices de pesos) para los pares de patrones. Cuando se entrena una red de esta clase en el simulador aparece una barra en la parte superior izquierda de la pantalla que indica el porcentaje de conexiones que se han establecido.

El algoritmo de aprendizaje no requiere de ningún parámetro para establecer los pesos de la red. El operador holográfico que emplea para el reconocimiento de patrones no utiliza ninguna variable que regule su operación.

Al realizar una verificación del aprendizaje aparece en la parte inferior de la pantalla el patrón de entrada temporal del lado izquierdo, y el patrón de salida que entrega la red del lado derecho.



2.22 Verificación del aprendizaje con memoria holográfica

Como se puede apreciar en la figura 2.22, la gráfica de la izquierda muestra una versión ruidosa de la letra A, tiene distinta escala y está trasladada con respecto a la A que aprendió la red. La gráfica del lado derecho muestra la reconstrucción completa de la letra.

BIBLIOGRAFIA

1.- Yoh Han Pao.

Adaptive Pattern Recognition and Neural Networks.

Addison Wesley, 1989.

2.- D. Gabor.

Associative Holographic Memories.

IBM J. Res. Develop, Vol. 13, pp. 156-159, 1969.

3.- J. Jesús González F.

Desarrollo de un simulador digital de redes neuronales artificiales tipo MAB.

Tésis de licenciatura (Físico), U.N.A.M., 1991

CAPITULO 3

REDES NEURONALES ARTIFICIALES CON APRENDIZAJE NO SUPERVISADO

3.1 CARACTERISTICAS GENERALES

Cuando una red neuronal solo recibe patrones de entrada, sin que estén asociados con patrones de salida, e intenta organizarlos en grupos o clases de manera autónoma, el aprendizaje es no supervisado.

Normalmente el aprendizaje ocurre cuando de antemano se conocen las características relevantes del conjunto de entrada. Sin embargo, frecuentemente no se cuenta con dicha información, y lograr que un sistema de aprendizaje pueda manejar un gran conjunto de patrones de entrada es computacionalmente costoso. El aprendizaje no supervisado puede ser utilizado como un módulo descubridor de características que precede al aprendizaje supervisado.

Generalmente en un esquema de aprendizaje no supervisado, las neuronas de la capa de salida compiten entre sí para obtener la respuesta correcta a cada patrón de entrada. La red adapta sus parámetros de acuerdo a las asociaciones estadísticas de los patrones de entrada y los clasifica en categorías similares.

3.2 TEORIA DE LA RESONANCIA ADAPTIVA

GROSSBERG Y CARPENTER

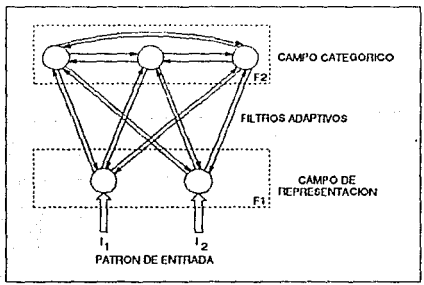
Una de las redes neuronales más complejas que se haya inventado es la basada en la teoría de resonancia adaptiva (ART), desarrollada por Stephen Grossberg y Gail Carpenter. Existen dos modelos de redes basados en ART que son entrenados sin supervisión; el primero, ART-1, que auto-organiza el reconocimiento de categorías para una secuencia arbitraria de patrones de entrada binarios en tiempo real; el segundo, ART-2, que realiza la misma tarea tanto para patrones de entrada binarios como análogos.

Las redes ART-1 son un buen intento de hacer un sistema de aprendizaje autónomo, puede categorizar automáticamente patrones de entrada y recordar patrones previamente aprendidos de forma inmediata. El entrenamiento consiste básicamente en presentar un patrón desconocido a la red, si es encontrado informa a que clase pertenece en caso contrario lo aprenderá como una nueva categoría. Mediante el ajuste de solo un parámetro, la red puede reorganizar su categorización de patrones en distinciones más finas según como sea apropiado. En muchos sentidos, la red satisface los requerimientos de un sistema de aprendizaje autónomo. Una gran desventaja de las redes ART-1 consiste en que sólo pueden aceptar patrones de entrada binarios.

La construcción básica de una red ART-1, consiste en dos capas de nodos totalmente interconectadas entre sí. Esta estructura es llamada "dipolo" por Grossberg, y a cada una de las capas "campo de compuerta bipolar". Estos nodos tienen una serie de propiedades interesantes, pero la de mayor atracción es que cuando los nodos activos reciben una señal de entrada particular llamada "señal global de vigilancia", se apagan y permanecen en ese estado por un lapso de tiempo no específico. Los nodos no activos no se ven afectados por esta señal.

Se construyen dos capas de nodos y se interconectan constituyendo filtros adaptivos, de tal manera que cada nodo en la capa de entrada se conecta a cada nodo de la capa de salida y viceversa. A las interconexiones se les llama filtros adaptivos porque hacen las veces de filtro cuando la actividad de los nodos de una capa pasa a la otra por medio de éstas y también porque van cambiando sus valores de acuerdo a las clasificaciones que haga la red.

Las dos capas, ligadas ambas por filtros adaptivos, constituyen el subsistema de atención. Las conexiones tienen pesos que son modificados lentamente en el transcurso del aprendizaje, como su tiempo de vida es alto, se les llama Términos de Memoria Prolongada. Todas las demás conexiones en la red tienen pesos fijos. En la figura 3.1 se puede apreciar la arquitectura utilizada por el modelo, la capa inferior o campo de representación se utiliza como entrada y es reconocida por F1, la capa superior o campo categórico es utilizada como salida y se le reconoce por F2.



3.1 Arquitectura de una red ART-1

La capa de salida, también llamada capa de almacenamiento, tiene arquitectura de inhibición lateral, lo que significa que en ésta se realiza una competencia autoregulada

entre sus nodos (véase Apéndice A). Cada nodo trata, simultáneamente, de activarse y evitar que los demás nodos en competencia se activen. Sólo el nodo con la respuesta más fuerte al patrón de entrada de la capa inferior será capaz de generar una salida.

Al aplicar un patrón binario a los nodos de entrada, la actividad producida en la capa de entrada es una copia exacta del patrón externo. Esta actividad es transmitida mediante las interconexiones de pesos a la capa de almacenamiento, generando a su vez cierta actividad en dicha capa. El patrón transmitido a la capa de almacenamiento es conocido como patrón "esperado" o "de abajo hacia arriba".

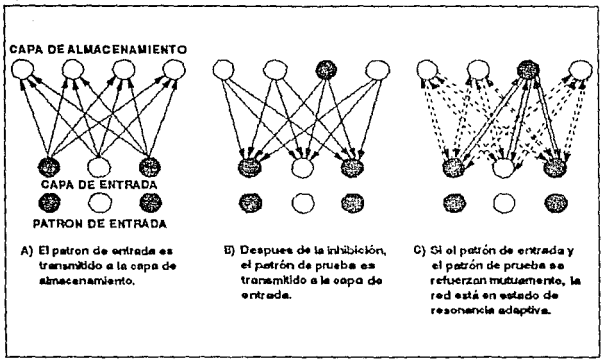
Debido a que el patrón de entrada es alterado por los pesos de las interconexiones, la actividad generada en la capa de almacenamiento es distinta de la generada en la capa de entrada, de hecho, debido a la inhibición lateral y a otro sistema contenido en la arquitectura de la red, la actividad generada por el patrón consiste exactamente de un nodo activo en la capa de almacenamiento.

El nodo activo se dice que ha codificado, clasificado, ortogonalizado o particionado al patrón de entrada.

El patrón de la capa de almacenamiento es retransmitido a la capa de entrada con las conexiones que van de ésta a la capa de entrada. Como antes, este patrón de "prueba" o "de arriba hacia abajo" es alterado por su paso a través de los pesos de las conexiones, y genera así cierta actividad en la capa de entrada.

Para la operación de la red, la capa de entrada debe de estar recibiendo aún el patrón de entrada cuando el patrón de prueba llega. Si la actividad generada en la capa de entrada por el patrón de prueba coincide con la actividad generada por el patrón de entrada, se dice que las capas están en estado de resonancia adaptiva [Figura 3.2]. El nodo que se activó en la capa de almacenamiento corresponde a la categoría asignada al patrón de entrada.

Cuando el patrón de entrada no es categorizado correctamente por la capa de almacenamiento, la actividad generada por el patrón de prueba no está en resonancia. Por esta razón se añade a la red un subsistema conocido como "vigilancia", con el fin de implementar una búsqueda de los patrones conocidos por la red para ver si alguno coincide con el patrón de entrada nuevo. Este mecanismo tiene tres funciones distintas, corregir la elección de una categoría errónea, reforzar el aprendizaje y responder a patrones de entrada distintos.

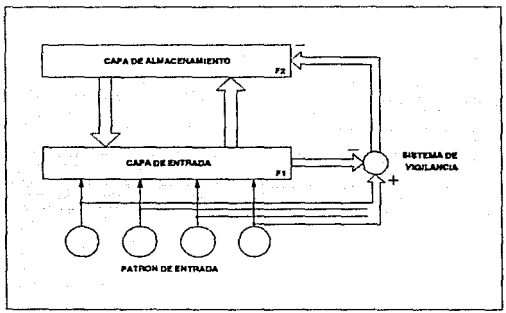


3.2 Teoría de la resonancia adaptiva

La implementación más sencilla del sistema de vigilancia consiste de una neurona que recibe entradas de dos lugares, de la capa de entrada y del patrón externo de entrada [Figura 3.3]. Los nodos de la capa de entrada están conectados a la neurona de vigilancia con un peso inhibitorio fijo, mientras que el patrón externo de entrada esta conectado con un peso excitatorio fijo a la neurona de vigilancia. La salida de la neurona de vigilancia va a cada uno de los nodos en la capa de almacenamiento con conexiones inhibitorias fijas. Esta entrada a la capa de almacenamiento, actúa como una señal de reset global a los nodos activos en la capa, desactivándolos y manteniéndolos en ese estado, mientras que los nodos inactivos no se ven afectados. Este proceso permite a los nodos, antes inactivos, tratar de ganar en la inhibición lateral. El nuevo ganador envía nuevamente un patrón de prueba a la capa de entrada para intentar la resonancia.

La neurona de vigilancia tiene un umbral variable que determina que tanto parecido debe de existir entre el patrón de prueba y el patrón de entrada. Si el umbral tiene un valor cercano a cero los dos patrones no deben ser muy parecidos, si por el contrario, tiene un valor cercano a uno los dos patrones deben ser muy similares. Si el umbral se fija en el valor uno, significa que los patrones deben coincidir exactamente, cualquier diferencia provocaría que el sistema de vigilancia se active. Este umbral es comúnmente llamado parámetro de vigilancia; "alta vigilancia" implica que la red está muy alerta a

pequeños cambios entre los patrones y corresponde a un valor alto de umbral, "baja vigilancia" significa que la red es relativamente insensible a pequeñas diferencias y corresponde a un valor bajo del umbral.



3.3 Sistema de vigilancia añadido a la red ART-1

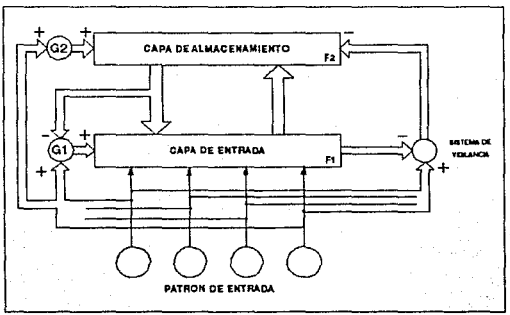
Cuando ocurre una discrepancia de patrones en la capa de entrada, debe garantizarse la disminución de la actividad total de la capa para que el sistema de vigilancia opere correctamente. Con este fin se añade a la red una sección, llamada subsistema de control, que consta de dos controles de ganancia.

El primer control de ganancia, conocido como G1, recibe el patrón de entrada de la misma manera en que lo recibe el sistema de vigilancia [Figura 3.4]. Todas las conexiones excitatorias que salen y entran al control de ganancia tienen pesos fijos. Adicionalmente, G1 recibe también una entrada inhibitoria de la capa de almacenamiento. Esta inhibición es tan fuerte que la mínima actividad en la capa de almacenamiento es suficiente para impedir que el control de ganancia actúe sobre los nodos de la capa de entrada.

Como se puede observar en la figura 3.4, la capa de entrada tiene tres posibles fuentes de señales de entrada: el patrón de entrada, el patrón de prueba proveniente de la capa de almacenamiento y el sistema de control de ganancia. Los umbrales de los nodos de la capa de entrada tienen un valor tal que dos de las tres posibles entradas deben estar presentes para que algún nodo se active. Este requisito es llamado la regla de los $^2/_3$, que es el elemento crucial para la operación de las redes ART-1.

El segundo sistema de control de ganancia llamado G2, recibe el patrón de entrada de la misma manera que G1. Cuando se presenta un patrón, G2 envía una pequeña señal excitatoria a todos los nodos en la capa de almacenamiento, esto sirve para que la capa tenga una mejor respuesta al patrón deseado. Adicionalmente, en algunas implementaciones, el control de ganancia G2 ayuda a moderar la competencia entre los nodos de la capa de almacenamiento, reduciendo el tiempo necesario para que la inhibición lateral decida cual es el nodo ganador.

La arquitectura que define el modelo de resonancia adaptiva se puede apreciar en la figura 3.4.



3.4 Sistema de control de ganancia añadido a la red ART-1

MODO DE OPERACION.

Al presentar un patrón binario de entrada a la red, éste se envía a tres lugares: a la capa de entrada, al control de ganancia y al sistema de vigilancia. Debido a que la capa de entrada está inactiva, cuando el patrón de entrada llega al sistema de vigilancia, una señal de reset global es transmitida a la capa de almacenamiento, desactivando cualquier nodo que pudiese estar activo, preparando de ésta manera a la capa de almacenamiento para responder al patrón de entrada. Mientras tanto, el patrón de entrada trata de generar actividad en la capa de entrada, no puede hacerlo por sí solo debido a que es tan solo una de las tres fuentes de estímulo a dicha capa.

El patrón de entrada también va al control de ganancia G1. Como el sistema de vigilancia desactivó todos los nodos de la capa de almacenamiento, el control de ganancia se activa y proporciona el segundo estímulo a la capa de entrada. La capa de entrada tiene dos fuentes de estimulación y puede activarse libremente. Debido a que el patrón es binario, esta acción reproduce exactamente el patrón de entrada en la capa de entrada.

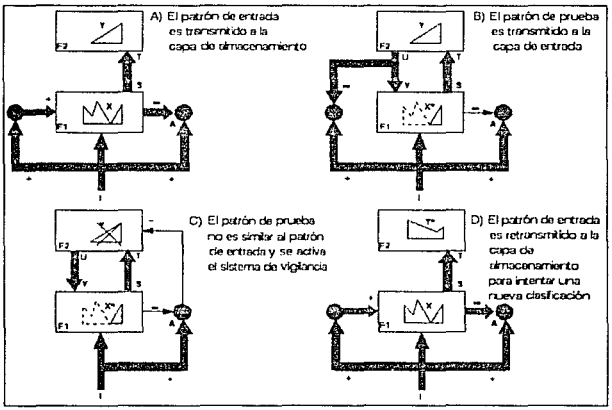
Una vez activa la capa de entrada, su actividad es transmitida al sistema de vigilancia, el cual balancea la excitación del patrón de entrada y cesa la señal global de reset. A su vez, el patrón deseado es presentado a la capa de almacenamiento. El patrón deseado y la arquitectura de competencia por inhibición lateral de la capa de almacenamiento producen actividad en un solo nodo o categoría. Esta señal es transmitida como el patrón de prueba a la capa de entrada y también a través de la conexión fuertemente inhibitoria al control de ganancia, el cual es apagado.

La capa de entrada ve ahora señales de entrada de sólo dos fuentes, el patrón de prueba y el patrón de entrada aún presente. El patrón de prueba genera cierta actividad en la capa de entrada. Solo los nodos que reciben actividad tanto del patrón de entrada como del patrón de prueba permanecen activos. Este proceso garantiza que el total de nodos activos en la capa de entrada será menor o igual que el número de nodos activos en el patrón de entrada.

En caso de que no coincidan, la actividad en la capa de entrada y la inhibición en el sistema de vigilancia disminuye. Si ésto provoca que la disparidad entre la excitación del patrón de entrada y la inhibición de la capa de entrada supere el umbral del sistema de vigilancia, éste se activa, enviando una señal global de reset a la capa de almacenamiento. Esto, por supuesto, termina con la actividad del nodo ganador en la capa de almacenamiento y le impide tratar de competir nuevamente.

Al terminar con la actividad en la capa de almacenamiento también retira la inhibición sobre el sistema de control de ganancia, logrando que éste actúe como un segundo

estímulo en la capa de entrada y permitiendo a la capa de entrada reproducir el patrón de entrada nuevamente. Adicionalmente el sistema de vigilancia es desactivado.



3.5 Modo de operación de una red ART-1

El proceso entero esta listo para comenzar de nuevo, excepto por dos cosas, los pesos han sido ligeramente modificados, y el nodo específico que trataba de clasificar el patrón en una categoría no interviene más en el proceso, es decir, la capa de almacenamiento debe buscar una categoría nueva. En la figura 3.5 aparecen los cuatro pasos básicos en la clasificación de una red ART-1.

ALGORITMO ART-1

1.-Inicialización.

Los pesos que van de la capa de almacenamiento a la capa de entrada: $v_{ij}(0)=1$.

Los pesos que van de la capa de entrada a la capa de almacenamiento:

$$w_{ij}(0)=\frac{1}{1+N}.$$

En estas ecuaciones los subíndices ij indican las conexiones entre los nodos de entrada i y los nodos de salida j. Donde:

$$0 \leq i \leq N-1.$$

$$0 \leq j \leq M-1.$$

N denota el número de nodos de entrada y M el número de nodos de salida o clases.

El parámetro de vigilancia para el sistema de vigilancia, denotado ß, puede tomar los siguientes valores: $0 \leq \beta \leq 1$.

2.-Mostrar un patrón de entrada a la red.

Los patrones se irán presentando en forma secuencial y será un arreglo binario [0,1] denotado como x_i.

3.-Generar actividad en la capa de almacenamiento.

$$u_j = \sum_{i=0}^{N-1} w_{ij}(t)\ x_i$$

Donde u_j es el nivel de activación del j-ésimo nodo de la capa de almacenamiento.

4.-Mediante inhibición lateral escoger el nodo ganador.

$$u_{j^*} = \max\{u_j\}$$

5.-Parámetro de vigilancia.

Verificar que el patrón de entrada x_i realmente pertenece a la clase elegida mediante las siguientes operaciones.

$$X = \sum_{i=0}^{N-1} x_i$$

$$V^*X = \sum_{i=0}^{N-1} v_{ij^*}x_i$$

Si, $\frac{V^*X}{X} > \beta$ entonces paso 7 sino paso 6.

6.-Dar un reset al nodo ganador.

Debido a que el vector de entrada X no pertence a la clase elegida en el paso 4, el nodo ganador es desactivado temporalmente hasta que se encuentra la clase. El patrón de entrada debe continuar presente y se repite el proceso desde el paso 3.

7.-Adaptar los pesos para el nodo ganador.

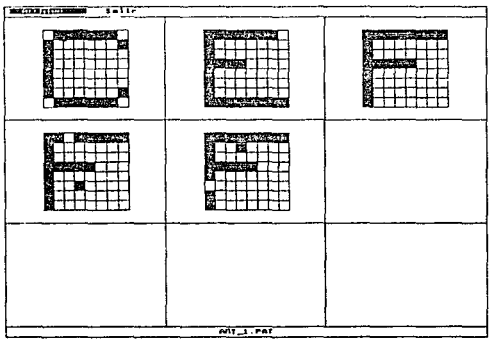
$$w_{ij}(t+1) = \frac{v_{ij*}(t)\ x_i}{0.5 + \sum_{i=0}^{N-1} v_{ij*}(t)\ x_i}$$

$$v_{ij*}(t+1) = v_{ij*}(t)\ x_i$$

8.-Repetir el proceso a partir del paso 2 para cada uno de los patrones de entrada.

EJEMPLO

Para ilustrar el algoritmo ART-1 se eligió el ejemplo que aparece en el artículo de Richard P. Lippmann titulado "An Introduction to Computing with Neural Nets", en el cual se pretende clasificar un conjunto de cinco letras, mostradas en la figura 3.6, con un parámetro de vigilancia elevado. La red empleada cuenta con 64 neuronas de entrada y 4 neuronas de salida.



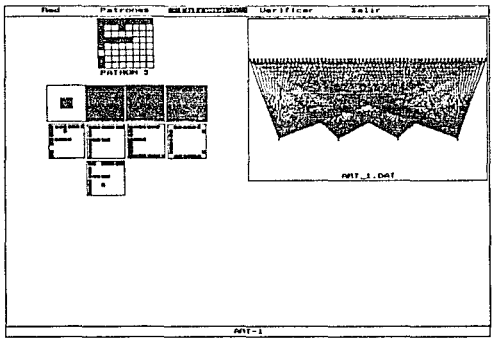
3.6 Despliegue del archivo de patrones ART-1

Al entrenar una red bajo el algoritmo ART-1 en el simulador, se muestra en la parte superior derecha de la pantalla la arquitectura básica de dos capas que utiliza este algoritmo. La capa de entrada es la superior y la capa de salida es la inferior.

Del lado izquierdo de la pantalla se muestra el patrón binario de entrada en una retícula de las mismas dimensiones que los patrones de entrenamiento generados. Por debajo del patrón se muestra una hilera de rectángulos que representa a las neuronas de la capa de salida de la red. La disposición de estos rectángulos es la misma que tienen las neuronas de salida en la arquitectura.

Durante el entrenamiento, la competencia de las neuronas de la capa de almacenamiento para clasificar al patrón de entrada puede apreciarse en los rectángulos correspondientes. La inhibición lateral se simula para distinguir la actividad de las neuronas cuando ésta ocurre; una mayor actividad corresponde a un rectángulo mayor.

Al término de la inhibición, es decir, cuando solo una neurona permanece con actividad, se verifica si ha clasificado correctamente al patrón, de ser así, éste será puesto debajo del rectángulo que representa a la neurona ganadora. En caso contrario se podrá apreciar el efecto del sistema de vigilancia que hará que la neurona no participe más en la competencia. La neurona sin actividad se representa con el rectángulo completamente blanco.



3.7 Clasificación de patrones en una red ART-1

En la figura 3.7 se muestra la clasificación de los cinco patrones. De los rectángulos que representan a la capa de almacenamiento, los tres de la derecha que se encuentran completamente llenos, representan a las neuronas que han sido apagadas por el sistema de vigilancia. El rectángulo de la izquierda, con cierta actividad, clasificó al patrón que aún esta presente en la parte superior.

Los parámetros que es necesario establecer para el entrenamiento son:

a) Nivel de activación de los procesadores: 1.0

Este parámetro es necesario para la simulación de la inhibición lateral y representa la pendiente de la función de transferencia o activación para las neuronas de la capa de almacenamiento.

b) Parámetro de vigilancia: 0.9

El parámetro de vigilancia está vinculado con la clasificación de los patrones de entrada. Si está muy cercano al uno la clasificación es muy precisa, es decir, la diferencia entre los patrones de una misma clase debe ser mínima. Un valor muy cercano al cero implica una clasificación deficiente.

Al hacer una verificación del aprendizaje, aparecen en la pantalla la arquitectura, el patrón de prueba y la hilera de rectángulos con los patrones previamente clasificados.

El patrón es presentado a la red y ésta trata de clasificarlo en una de las neuronas de salida mediante la inhibición lateral. Si el patrón pertenece a una clase establecida la neurona con mayor actividad representa dicha clasificación, en caso contrario todas las neuronas serán apagadas por el sistema de vigilancia.

BIBLIOGRAFIA

1.- Gail A. Carpenter, Stephen Grossberg.

ART 2: self-organization of stable category recognition codes for analog input patterns.

Applied Optics, Vol.26 No.23, pp.4919-4930, Dic. 1987.

2.- Gail A. Carpenter, Stephen Grossberg.

The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network.

Computer, Vol 21, pp. 77-88, Marzo 1988.

3.- Stephen Grossberg.

Neural Networks And Natural Intelligence.

MIT Press, Cambridge Massachusetts, 1988.

4.- Maureen Caudill.

Resonance Adaptive Theory.

AI Expert, Neural Networks Primer, 1989.

3.3 MAPAS DE AUTO-ORGANIZACION

TEUVO KOHONEN

Los mapas de auto-organización son redes neuronales de dos capas de unidades que pueden organizar un mapa topológico a partir de una configuración aleatoria. Después del entrenamiento, el mapa resultante representa la relación categórica que existe entre los patrones que fueron mostrados a la red.

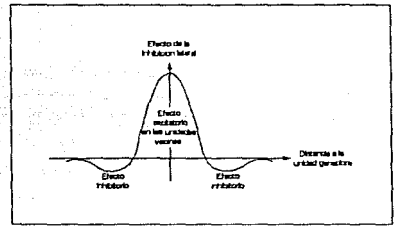
El modelo de Kohonen muestra como una red neuronal artificial puede simular procesos semejantes al mapeo topológico de fenómenos sensores y motores que ocurren en la corteza cerebral.

La primera capa de la red, es la capa de entrada cuya única función es la de enviar el patrón de entrada a la segunda capa a través de las conexiones. La segunda capa, típicamente organizada como un arreglo bidimensional, es competitiva; las unidades realizan inhibición lateral para que aquella que mejor represente categóricamente al patrón de entrada permanezca activa (véase Apéndice A).

El sistema diseñado por Kohonen como un modelo biológico utiliza las interconexiones de la capa competitiva para regular la competencia. Cada unidad en la capa utiliza su nivel de activación para actuar como entrada inhibitoria para cada una de las unidades restantes. Esta entrada inhibitoria tiende a reducir el nivel de activación de las demás unidades y, por consecuencia, reduce su correspondiente entrada inhibitoria.

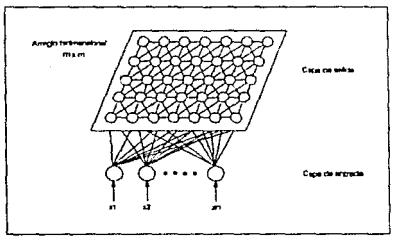
En la mayor parte de las interconexiones, el efecto es totalmente inhibitorio, pero en las unidades vecinas es excitatoria. Esto provoca que cada unidad trate de ganar la competencia y, de no ser posible, ayuda a las unidades vecinas a hacerlo mediante la excitación.

La efectividad de la inhibición lateral utilizada por Kohonen puede ser representada como una función de la distancia de las unidades vecinas a la unidad que ganó la competencia. Gráficamente esta función tiene cierto parecido a la función conocida como sombrero mexicano [Figura 3.8].



3.8 Función "Sombrero Mexicano"

Cuando es presentado un patrón, cada unidad en la primera capa toma el valor correspondiente a la entrada. En la segunda capa las unidades suman sus entradas y compiten para encontrar solo una unidad ganadora. Una vez encontrada la unidad vencedora se actualizan los pesos en forma radial. Todas las conexiones van de la capa de entrada a la capa competitiva, las dos capas están totalmente interconectadas [Figura 3.9].



3.9 Arquitectura de la red empleada por mapas de auto-organización

Al principio del entrenamiento los pesos tienen valores aleatorios, sin embargo, para obtener un aprendizaje más acelerado el valor inicial de los pesos se asigna sumando un pequeño valor aleatorio al valor promedio de los patrones de entrada.

Un patrón de entrada a la red es denotado como un vector:

$$X = [x_1, x_2, x_3, ..., x_n]$$

Las conexiones de la capa de entrada a la capa competitiva se denotan como Wij, donde el subíndice i representa a la i-ésima unidad en la capa competitiva y el subíndice j representa la j-ésima unidad en la capa de entrada.

MODO DE OPERACION

El primer paso en la operación de la red de Kohonen es encontrar el nivel de activación de cada unidad en la capa competitiva cuando es presentado un patrón de entrada. Este valor indica el parecido que existe entre el vector de entrada y el valor de los pesos. Para la i-ésima unidad el nivel de activación está dado por:

$$| X - W_i | = (\sum_j (x_j - w_{ij})^2)^{\frac{1}{2}}$$

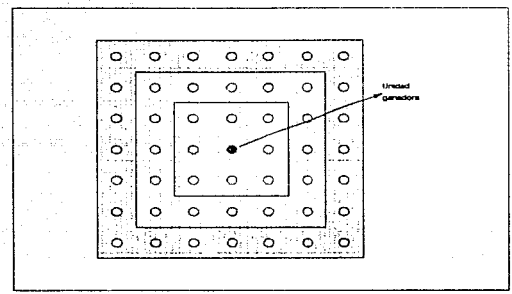
Que es la distancia entre los vectores X y W_i.

La unidad con la menor distancia es la que gana la competencia, pues el parecido que existe entre los pesos conectados a ella y el patrón de entrada es el mejor. Por notación la unidad que obtiene el mejor parecido se identifica con un asterisco, que es elegida mediante:

$$| X - W^* | = \min_i \{ | X - W_i | \}$$

Donde el mínimo se toma sobre todas las unidades en la capa competitiva. Si dos unidades llegaran a tener el mismo nivel de activación entonces, por convención, se toma a la unidad con el menor índice.

Una vez que se ha elegido a la unidad ganadora, el siguiente paso es identificar las unidades vecinas a ésta, es decir, las unidades en la capa competitiva que están cerca de la ganadora en el arreglo bidimensional. Las unidades vecinas son aquellas que están distribuidas en forma radial tomando como centro a la unidad ganadora, el conjunto generado se representa como V^* [Figura 3.10].



3.10 Vecindades definidas para la unidad ganadora

El tamaño del radio de unidades vecinas cambia de acuerdo al avance en el aprendizaje. Los pesos son actualizados para todas las unidades que se encuentran en la vecindad de la ganadora. El cambio en los pesos se realiza de la siguiente manera:

$$w_{ij}(t+1) = w_{ij}(t) + \begin{cases} \alpha(x_j - w_{ij}(t)), \text{ si la unidad i pertenece a } V^* \\ 0 \quad \text{De cualquier otra manera.} \end{cases}$$

Donde α es el factor de aprendizaje.

De esta manera se logra que los pesos que van de la capa de entrada a la unidad ganadora y a sus vecinas sean más parecidos al patrón de entrada. Como resultado la unidad ganadora tendrá mejor respuesta a patrones similares mostrados posteriormente.

El proceso de aprendizaje se lleva al cabo presentando repetidamente el conjunto de patrones a la red. Cada vez que es presentado un patrón, los pesos de la unidad ganadora y de sus vecinas son actualizados. Conforme avanza el entrenamiento el factor de aprendizaje debe disminuir proporcionalmente al número de iteraciones realizadas, ésta misma regla se aplica para definir la vecindad de las neuronas ganadoras en cada iteración.

El factor de aprendizaje α tiene valores entre 0 y 1. Al iniciar el entrenamiento su valor debe ser relativamente alto, durante el proceso de aprendizaje debe disminuir hasta alcanzar el valor de cero. Un método aceptable para disminuir el valor de α linealmente al número de iteraciones, está dado por la ecuación:

$$\alpha_t = \alpha_0 (1 - \frac{t}{T})$$

Donde: α_t, es el valor de α en la iteración t.

α_0, es el valor inicial de α.

t, es el número de iteración actual, y

T, es el número total de iteraciones a realizar.

La vecindad definida en el contorno de una unidad ganadora tiene que disminuirse también conforme el entrenamiento avanza para obtener una clasificación más nítida en las subsecuentes presentaciones de los patrones. El radio de vecindad inicial debe ser relativamente grande, los valores típicos se encuentran entre la mitad y una tercera parte del ancho del arreglo bidimensional de la capa competitiva. El valor de la vecindad al término del entrenamiento debe ser igual a uno, por lo que este valor tiene que ser disminuido de acuerdo a la ecuación:

$$v_t = 1 + v_0 (1 - \frac{t}{T})$$

Donde: v representa el radio o ancho de la vecindad.

v_t, es el valor del radio en la iteración t.

v_0, es el valor inicial del radio.

t, es el número de iteración actual, y

T, es el número total de iteraciones a realizar.

ALGORITMO DE APRENDIZAJE

1.- Inicializar los pesos con valores alrededor del promedio de los patrones de entrada.

2.- Definir el radio de vecindad v_0 para comenzar el entrenamiento.

3.- Definir el factor de aprendizaje α_0 para la red.

4.- Mostrar un patrón de entrada.

5.- Definir el nivel de activación para cada neurona del arreglo de salida:

$$| X - W_i | = (\sum_j (x_j - w_{ij})^2)^{\frac{1}{2}}$$

6.- Encontrar la neurona que mejor representa al patrón de entrada:

$$| X - W^* | = \min_i \{ | X - W_i | \}$$

7.- Modificar el peso de las neuronas de acuerdo a:

$$w_{ij}(t+1) = w_{ij}(t) + \begin{cases} \alpha(x_j - w_{ij}(t)), \text{ si la unidad i pertenece a } V^* \\ 0 \quad \text{De cualquier otra manera.} \end{cases}$$

8.- Repetir desde el paso 3 para todos los patrones de entrada.

9.- Disminuir la constante de aprendizaje y la vecindad:

$$\alpha_t = \alpha_0(1 - \frac{t}{T})$$

$$v_t = 1 + v_0(1 - \frac{t}{T})$$

Donde α_t, es el valor de α en la iteración t.
α_0, es el valor inicial de α.
v_t, es el radio de vecindad en la iteración t.
v_0, es el radio de vecindad inicial.
t, es el número de iteración actual, y
T, es el número total de iteraciones a realizar.

10.- Repetir desde el paso 3, hasta completar el número de iteraciones deseado.

ALGORITMO DE APRENDIZAJE

1.- Inicializar los pesos con valores alrededor del promedio de los patrones de entrada.

2.- Definir el radio de vecindad v_0 para comenzar el entrenamiento.

3.- Definir el factor de aprendizaje α_0 para la red.

4.- Mostrar un patrón de entrada.

5.- Definir el nivel de activación para cada neurona del arreglo de salida:

$$| X - W_i | = \left(\sum_j (x_j - w_{ij})^2 \right)^{\frac{1}{2}}$$

6.- Encontrar la neurona que mejor representa al patrón de entrada:

$$| X - W^* | = \min_i \{ | X - W_i | \}$$

7.- Modificar el peso de las neuronas de acuerdo a:

$$w_{ij}(t+1) = w_{ij}(t) + \begin{cases} \alpha(x_j - w_{ij}(t)), \text{ si la unidad i pertenece a } V^* \\ 0 \quad \text{De cualquier otra manera.} \end{cases}$$

8.- Repetir desde el paso 3 para todos los patrones de entrada.

9.- Disminuir la constante de aprendizaje y la vecindad:

$$\alpha_t = \alpha_0 \left(1 - \frac{t}{T}\right)$$

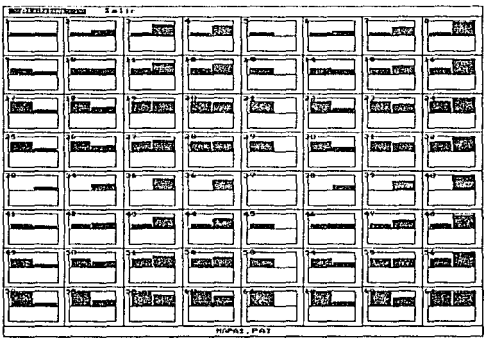
$$v_t = 1 + v_0 \left(1 - \frac{t}{T}\right)$$

Donde α_t, es el valor de α en la iteración t.
α_0, es el valor inicial de α.
v_t, es el radio de vecindad en la iteración t.
v_0, es el radio de vecindad inicial.
t, es el número de iteración actual, y
T, es el número total de iteraciones a realizar.

10.- Repetir desde el paso 3, hasta completar el número de iteraciones deseado.

EJEMPLO

Para ilustrar el algoritmo de mapas de auto-organización se generó un conjunto de 64 patrones con distribución normal con el fin de poder apreciar, al término del entrenamiento, el mapa generado de acuerdo a como aparece en la literatura. En la figura 3.11 se puede apreciar que el archivo de patrones empleado tiene solamente dos valores para cada patrón. La red utilizada tiene, por consiguiente, dos neuronas de entrada y 64 de salida.



3.11 Despliegue del archivo de patrones MAPAS.PAT

Una red entrenada por el algoritmo de mapas de auto-organización tiene una arquitectura definida por dos capas. Las neuronas de la capa de salida deben de formar, forzosamente, un arreglo que defina una figura geométrica, por ejemplo, un triángulo, un cuadrado, un rectángulo, una circunferencia, etc. Para el simulador se decidió emplear un arreglo cuadrado para simplificarle al usuario la definición de la arquitectura.

El dibujo de la arquitectura de la red, por condiciones de espacio, no incluye el arreglo cuadrado de las neuronas de salida. En la gráfica generada por el simulador sólo aparecen las neuronas de entrada en la parte superior conectadas a las neuronas de salida, dispuestas en forma linear, en la parte inferior.

Durante el entrenamiento de la red se muestra en el lado superior izquierdo de la pantalla una gráfica que representa la organización de los pesos en la red conforme transcurre el aprendizaje. Si el número de neuronas de entrada es de dos, se utiliza la

representación que propone Kohonen, un arreglo bidimensional formado por el mismo número de vértices como neuronas en la capa de salida que, después de entrenada la red, parece una hoja cuadriculada.

Si el número de neuronas de entrada es mayor a dos, entonces se muestra una gráfica en la que aparecen las neuronas de salida dispuestas en un arreglo cuadrado en donde los pesos que van de ésta a la capa de entrada forman, para cada neurona, una estructura semejante a una estrella. Cada raya en la estrella representa una conexión de la neurona de salida a una neurona en la capa de entrada, el largo de la raya es proporcional a la magnitud del peso y la distancia entre cada raya esta espaciada sobre un círculo. El peso que va de la neurona de salida a la primera neurona en la capa de entrada está representado por la raya que apunta exactamente al norte, los siguientes están ordenados hacia la derecha.

Los parámetros necesarios para el entrenamiento por mapas de auto-organización son:

a) Epocas a realizar: 100

Este parámetro representa el número de veces que deberá ser mostrado el conjunto de patrones a la red para que ésta los clasifique.

b) Epocas de refinamiento: 50

Cuando ha terminado el entrenamiento básico es necesario, en ciertas ocasiones, llevar al cabo una etapa que ha sido llamada de refinamiento, en la que el cambio en los pesos se realiza exclusivamente para una neurona en la capa de salida.

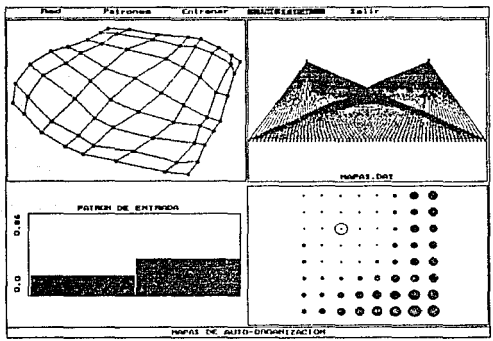
c) Alfa: 0.15

Es el valor inicial que se debe asignar para regular el cambio en los pesos. Este valor es disminuido paulatinamente conforme avanza el aprendizaje.

d) Vecindad: 8

La vecindad se define como el número de neuronas que son excitadas por la neurona que mejor representa a un patrón de entrada. El cambio en los pesos para la neurona que ha clasificado al patrón y para las neuronas vecinas es el mismo. El número de neuronas que definen la vecindad es disminuido conforme avanza el aprendizaje, por ésta razón es necesario asignarle un valor inicial.

Al realizar una verificación del aprendizaje aparece dentro de un recuadro en la parte izquierda inferior de la pantalla la entrada en una gráfica de barras. En la parte derecha inferior aparece la capa de salida dispuesta en un arreglo cuadrado antes establecido. El tamaño de cada neurona es proporcional a la actividad que tiene en respuesta al patrón de entrada, la neurona que mejor respuesta tiene al patrón de entrada se circula.



3.12 Verificación del aprendizaje en mapas de auto-organización

En la figura 3.12 se muestra la clasificación de un patrón de entrada escogido aleatoriamente del conjunto de patrones. En la representación del arreglo cuadrado de neuronas se puede apreciar el efecto de la inhibición que se lleva a cabo en la capa de salida cuando es categorizado un patrón de entrada. La actividad de las neuronas vecinas y de la neurona ganadora, que se distingue por estar dentro de un círculo, es muy semejante.

BIBLIOGRAFIA

1.- Teuvo Kohonen.

The Self-Organizing Map.

IEEE, Septiembre 1990, pp. 1464-1480.

2.- Kohonen T.

Self Organization and Associative Memory.

New York: Springer-Verlag, 2a. Edición, 1988.

3.- Yoh-Han Pao.

Adaptive Pattern Recognition and Neural Networks.

Addison Wesley, 1989.

4.- Judith E. Dayhoff.

Neural Network Architectures.

Van Nostrand Reinhold, Nueva York, 1990

CAPITULO 4

REDES NEURONALES ARTIFICIALES HIBRIDAS

4.1 CARACTERISTICAS GENERALES

Las redes neuronales artificiales híbridas son un caso particular de las redes neuronales. Se han resuelto problemas mediante la combinación de dos modelos distintos de redes neuronales, pero un modelo que emplee los dos tipos de aprendizaje para asociar patrones entrada/salida es poco frecuente.

Los modelos de redes neuronales híbridos utilizan aprendizaje supervisado y no supervisado durante su entrenamiento. Estan sujetos a un aprendizaje supervisado para asociar pares de patrones entrada/salida. Y utilizan el aprendizaje no supervisado, básicamente emplean arquitecturas de inhibición lateral, para rescatar las características estadísticas del conjunto de patrones y facilitar la asignación de pesos.

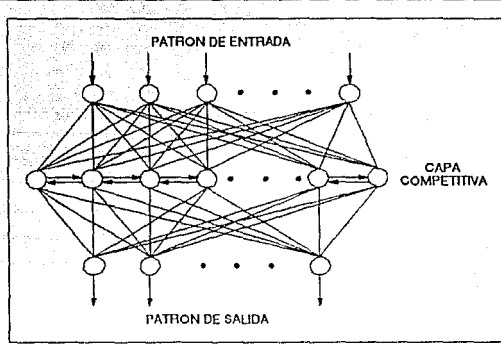
Estos modelos no han tenido el éxito esperado, sin embargo, su eficacia es digna de considerarse. Reducen el tiempo del aprendizaje notoriamente, en modelos como retropropagación éste proceso suele ser muy tardado. La gran desventaja que tienen es que exigen, para su funcionamiento, una cantidad de memoria mayor a la requerida por otros modelos.

4.2 CONTRA-PROPAGACION

HECHT-NIELSEN

Contrapropagación fue originalmente propuesto y desarrollado por Robert Hecht-Nielsen en 1987, como un sistema para el reconocimiento de patrones. Es utilizado, actualmente, para aplicaciones que requieren de completar o mapear patrones y puede servir de igual manera como una memoria asociativa bidireccional.

El algoritmo de contrapropagación emplea una red neuronal artificial construida a partir de dos modelos. La red es estrictamente de tres capas de unidades interconectadas totalmente entre sí; la primer capa es utilizada exclusivamente como entrada y su única función es enviar el patrón de entrada a través de las conexiones a la capa intermedia. La capa intermedia es una capa competitiva de Kohonen (véase capítulo 3) que realiza aprendizaje sin supervisión y la última es la capa de salida que es entrenada por la regla de Grossberg [Figura 4.1].



4.1 Arquitectura de una red de contrapropagación

Cuando se muestra un patrón a la red, las unidades de entrada generan, a través de las conexiones, actividad en la capa intermedia, las unidades de esta capa suman entonces sus entradas y compiten en respuesta al patrón de entrada; sólo una unidad resulta vencedora y es la única en la capa que permanece con actividad. Esta unidad representa una clasificación del patrón de entrada en un grupo particular. El nivel de actividad de la unidad vencedora es 1 y el de todas las restantes es 0. La unidad vencedora genera la salida de la red activando un patrón asociado al de entrada.

Los pesos que van de la capa de entrada a la capa intermedia son denotados como w_{ij}, y los pesos de la capa intermedia a la capa de salida se denotan como v_{jk}. Los pesos w_{ij} son inicializados con valores aleatorios y después son normalizados para cada neurona de la capa intermedia, es decir, $|w_j|=1$. El procedimiento de actualización mantiene esta normalización. Los pesos v_{jk} tienen valores iniciales aleatorios y el algoritmo de aprendizaje se encarga de modificarlos.

Cuando se presenta un patrón de entrada a la red, el vector representa el nivel de activación de la capa de entrada. Las unidades de la capa intermedia realizan la suma de sus entradas y definen su nivel de activación de la siguiente manera:

$$u_j = \Sigma\, x_i w_{ij}$$

Donde x_i es el patrón de entrada y u_j es el nivel de activación de la j-ésima unidad de la capa intermedia.

Para elegir la unidad ganadora basta con encontrar el máximo sobre todos los niveles de activación:

$$u_* = \max_j u_j$$

En el caso de un empate entre unidades se escoge, por lo general, a la unidad más a la izquierda de la configuración como ganadora. Después de la competencia los niveles de activación cambian para las unidades de la capa intermedia:

$$u_* = 1$$

$$u_j = 0, \text{ para toda } j \neq *$$

Donde u_* es el nivel de activación de la unidad ganadora y u_j es el nivel de activación (nulo) para el resto de las unidades en la capa intermedia.

Una vez realizada la competencia, la red ha clasificado al patrón de entrada en un grupo particular utilizando un vector de referencia asociado. Los pesos que van de la capa de entrada a la unidad ganadora son actualizados en este punto y los pesos restantes permanecen iguales. La actualización de los pesos se realiza de la siguiente manera:

$$w_{i*}(t+1) = w_{i*}(t) + \alpha(x_i - w_{i*}(t))$$

Donde α es la constante de aprendizaje ($0 < \alpha << 1$).

Cuando se han adaptado los pesos, el vector w_j es renormalizado dividiéndose entre su mismo módulo:

$$w_* = \frac{w_*}{|w_*|}$$

Una vez generada la actividad en la capa intermedia las unidades de la capa de salida realizan la suma de sus entradas, llevando al cabo la operación:

$$u_k = \Sigma u_j v_{jk}$$

Donde u_k es el nivel de activación de la k-ésima unidad de la capa de salida. Debido a que tan solo una unidad en la capa intermedia permanece con nivel de activación unitario, el nivel de activación de las unidades de salida es precisamente el valor de los pesos conectados a ellas y a la unidad vencedora de la capa intermedia:

$$u_k = v_{*k}$$

Los pesos v_{jk} son inicializados con valores aleatorios, por ésta razón, deben ser corregidos para aproximar el vector de salida deseado.

La adaptación de éstos pesos se lleva al cabo con la regla de Widrow-Hoff o Grossberg:

$$v_{jk}(t+1)=v_{jk}(t) + \alpha u_j (y_k - u_k)$$

Donde y_k es el patrón de salida deseado.

Sin embargo, esta regla se simplifica debido a que el nivel de activación de la neurona vencedora es uno y el de las restantes es cero, por lo que la regla de adaptación de pesos se puede reescribir de la siguiente manera:

$$v_{jk}(t+1) = \begin{cases} v_{*k}(t+1) = v_{*k}(t) + \alpha (y_k - u_k) \\ v_{jk}(t+1) = v_{jk}(t), \text{ para toda } j \neq * \end{cases}$$

Así, los únicos pesos ajustados para la salida de la k-ésima unidad son los conectados a la unidad ganadora de la capa intermedia.

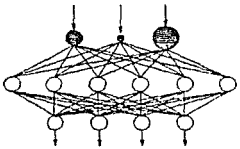
Para el entrenamiento de la red se requiere de presentar consecutivamente el conjunto de patrones entrada/salida hasta que la diferencia en el cambio de los pesos sea mínima. Para establecer un parámetro métrico que indique el grado de aprendizaje de la red se puede utilizar el error cuadrático:

$$E = \tfrac{1}{2} \sum_p \sum_k (y^p_k - u^p_k)^2$$

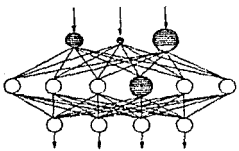
Donde y^p_k es el nivel de activación deseado para la k-ésima unidad de salida correspondiente al patrón p, y u^p_k es el nivel de activación generado por la k-ésima cuando se presentó el patrón p.

En la figura 4.2 se muestra gráficamente el proceso de aprendizaje que ocurre en el entrenamiento por contra-propagación.

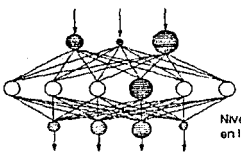
El algoritmo sufre de ciertas deficiencias, pero la convergencia es mucho más rápida que en otros algoritmos. Al ajustar los pesos que van de la capa de entrada a la capa intermedia, la normalización tiene el efecto de mover los pesos W_g hacia el vector de entrada en una esfera de radio unitario. Si se presenta otro vector de entrada que coincida con la misma neurona ganadora, los pesos W_g son reorientados hacia el nuevo vector. Al finalizar el entrenamiento los pesos representan el promedio de todos los vectores de entrada para los cuales una misma neurona resultó ganadora.



A) Se muestra el patrón en la capa de entrada



B) Neurona ganadora de la capa intermedia



C) Ajuste de los pesos de la neurona ganadora
de acuerdo a los patrones entrada/salida

4.2 Proceso de aprendizaje por contrapropagación

Sin embargo, el problema más fuerte que tiene el algoritmo se presenta cuando dos o más vectores de salida, no del todo similares, son asociados a una misma neurona ganadora en la capa intermedia, la red no es capaz de aprenderlos. El ajuste en los pesos de la capa intermedia a la capa de salida almacena precisamente en los lazos los valores de los vectores de salida; si no son coincidentes el error cuadrático nunca será mínimo.

Para garantizar, hasta cierto punto, un buen desempeño de la red, es necesario que el número de unidades de la capa intermedia sea mayor o igual al número de patrones.

ALGORITMO DE APRENDIZAJE

1.- Inicilaizar los pesos de la red con valores aleatorios entre [-1,1].

2.- Normalizar los pesos que van de la capa de entrada a la capa intermedia.

3.- Mostrar un patrón de entrada.

4.- Definir la actividad de la capa intermedia:

$$u_j = \Sigma\ x_i w_{ij}$$

Donde u_j es la actividad de la j-ésima neurona de la capa intermedia.

5.- Escoger la clase a la que pertence, es decir, la neurona con mayor actividad:

$$u_* = \max_j u_j$$

6.- Actualizar los pesos que van de la capa de entrada a la capa intermedia:

$$w_{i*}(t+1) = w_{i*}(t) + \alpha(x_i - w_{i*}(t))$$

7.- Normalizar los pesos:

$$w_* = \frac{w_*}{|w_*|}$$

8.- Definir la actividad de la capa de salida:

$$u_k = v_{*k}$$

9.- Actualizar los pesos de la capa intermedia a la capa de salida:

$$v_{jk}(t+1) = v_{jk}(t) + \alpha u_j(y_k - u_k)$$

10.-Repetir desde el paso 3 para cada patrón de entrada.

11.-Calcular el mínimo error cuadrático:

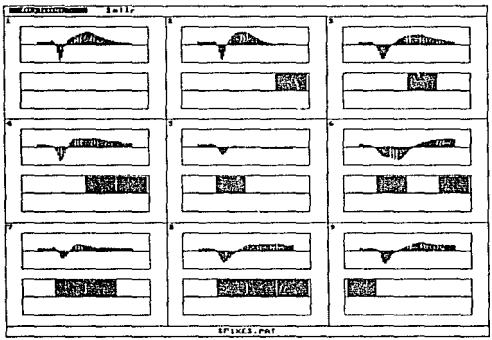
$$E = \tfrac{1}{2} \sum_p \sum_k (y_k^p - u_k^p)^2$$

12.-Repetir desde el paso 3 hasta que el mínimo error cuadrático sea muy pequeño.

EJEMPLO

Para ilustrar el algoritmo de contrapropagación, al igual que en retropropagación (Capítulo 2), se ha elegido un conjunto de 9 señales bioeléctricas, potenciales de acción, obtenidas en un experimento de laboratorio y transformadas a números en código ASCCI mediante un convertidor.

La intención del ejemplo es que la red asocie a cada patrón de entrada un número binario de 4 posiciones. En la figura 4.3 se muestra el archivo de patrones empleado.



4.3 Archivo de patrones SPIKES.DAT

La red utilizada tiene 64 neuronas de entrada, 9 neuronas intermedias, que corresponden al número de patrones, y 4 neuronas de salida. En el simulador, la pantalla de presentación mostrada durante el entrenamiento de una red por contrapropagación integra a la arquitectura y a una gráfica del error cuadrático producido por la red (véase figura 4.4).

La gráfica del error cuadrático, mostrada en la parte superior izquierda de la pantalla, tiene como eje horizontal las iteraciones que debe realizar la red para su entrenamiento y como eje vertical el porcentaje de error que produce la red cuando es mostrado el conjunto de patrones. Por debajo de la gráfica aparece el número de iteración en el que se encuentra el entrenamiento y porcentaje de error para esa iteración.

La arquitectura, mostrada en la parte superior derecha de la pantalla, debe estar formada estrictamente por tres capas de neuronas. La capa de entrada es la más superior y la de salida la más inferior.

Los parámetros a establecer para el entrenamiento por contra-propagación se enumeran a continuación:

a) Intervalo de los pesos : 1.0

Es el valor en el que se deben de encontrar los pesos cuando una red comienza el entrenamiento. Este valor solo puede ser cambiado antes de que la red sea entrenada por primera vez.

b) Alfa: 0.5

Es la constante de aprendizaje que regula el cambio de los pesos que conectan a la capa de entrada con la capa intermedia.

c) Beta: 0.5

Es la constante de aprendizaje para los pesos que van de la capa intermedia a la capa de salida.

d) Número de iteraciones: 50

Es el número de veces que deberá ser mostrado el conjunto de patrones para el aprendizaje de la red. El proceso de aprendizaje se detiene cuando se alcanza el número de iteraciones establecido o cuando el error generado por la red es menor al establecido.

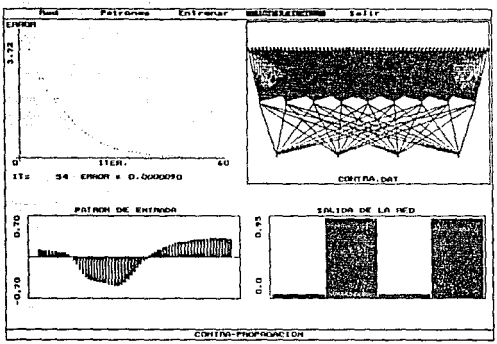
e) Error total: 0.00000001

Es el porcentaje de error máximo que debe tener la red para considerar que el conjunto de patrones ha sido aprendido.

Al hacer una verificación del aprendizaje se muestran nuevamente las gráficas del error y de la arquitectura. Además se integran una gráfica del patrón de entrada en la parte inferior izquierda de la pantalla y una gráfica de la actividad neuronal de la capa de salida en la parte inferior derecha de la pantalla.

Como se puede apreciar en la figura 4.4, la gráfica del patrón de entrada esta incluida en un pequeño recuadro. Debido a que son patrones continuos se establecen, del lado izquierdo del recuadro, el máximo y el mínimo para el conjunto de patrones y el patrón temporal de prueba . Dentro del recuadro se dibuja al patrón utilizando barras.

La actividad neuronal de la capa de salida, también dentro de un recuadro, se presenta de igual manera que el patrón de entrada, es decir, mediante barras.



4.4 Verificación del aprendizaje en contrapropagación

BIBLIOGRAFIA

1.-Hecht Nielsen

Counterpropagation Networks.

Applied Optics, Vol. 26, pp 4979-4984, 1987

2.-R. Hecht Nielsen.

Applications of Counterpropagaction Networks.

Neural Networks, Vol. 1, pp. 131-139, 1988.

3.-Judith E. Dayhoff.

Neural Network Architectures.

Van Nostrand Reinhold, Nueva York, 1990

CAPITULO 5

TUTORIAL

5.1 CARACTERISTICAS GENERALES

El simulador de redes neuronales artificiales está conformado por una serie de programas desarrollados en Turbo Pascal Versión 5.0, con interface gráfica que permite implementar y utilizar distintos modelos de redes neuronales de forma amable y rápida. Los algoritmos que incluye son:

ART-1, Máquina de Boltzmann, Contra-Propagación, Hopfield, Memoria Asociativa Holográfica, Mapas de Auto-Organización y Retro-Propagación.

Para cada uno de los modelos antes mencionados el simulador cuenta con las siguientes herramientas:

A) Generar Patrones. Los patrones de entrenamiento para las redes son archivos de datos con caracteres en código ASCII. El simulador cuenta con una rutina que sirve para generar patrones binarios en forma de retículas o patrones continuos para señales analógicas.

B) Almacenar redes. El aprendizaje en las redes neuronales suele ser un poco lento, por lo que el simulador tiene la capacidad de interrumpir dicho entrenamiento en cualquier momento y capturar las características de la red para su uso posterior.

C) Monitoreo del entrenamiento. Para todos los casos se ha elegido una forma representativa de la actividad de la red durante su entrenamiento, el monitoreo sirve exclusivamente para apreciar el desempeño de la red durante la fase de aprendizaje.

D) Verificación. Una vez que ha terminado el entrenamiento, o durante el mismo, se pueden realizar pruebas de eficiencia que permiten verificar el aprendizaje de la red.

El simulador está dirigido hacia estudiantes o investigadores que tengan cuando menos un conocimiento general acerca de redes neuronales artificiales y cómo deben ser operadas, aunque también puede ser utilizado por novatos en el área con el fin de familiarizarse con las estructuras, capacidades y modo de operación de las redes neuronales.

Las necesidades del simulador para su operación son las básicas, es decir, se requiere de una computadora personal (compatible con IBM), una unidad de disco y un

monitor que soporte gráficos. En partícular, el simulador fue programado y utilizado en una computadora IBM modelo 50.

El simulador opera de acuerdo a la capacidad de la máquina y trata siempre de obtener la mayor eficiencia posible, sin embargo, la apreciación gráfica se dificulta en monitores de baja resolución. La programación incluye una interface para mouse que hace al simulador más versátil.

5.2 CONTENIDO DEL DISKETTE

Los archivos que conforman al simulador están contenidos en un diskette con formato de 720 Kbytes. La distribución de los mismos se especifica a continuación.

En el directorio raíz del diskette, se encuentran los programas ejecutables necesarios para la operación del simulador:

- ART.EXE
- BOLTMANN.EXE
- CONTRA.EXE
- HOLGRAFI.EXE
- HOPFIELD.EXE
- MAPAS.EXE
- RETROPRO.EXE
- SRN.EXE

El programa SRN.EXE es el programa administrador, y es el que comienza la ejecución del simulador; las siglas significan "Simulador de Redes Neuronales". El resto de los programas ejecutables corresponden a los algoritmos de aprendizaje y pueden ser utilizados de forma autónoma (véase Apendice C).

Además se encuentran una serie de archivos con extensión BGI que son los manejadores (Drivers) para los distintos monitores que soporta el simulador:

- ATT.BGI
- CGA.BGI
- EGAVGA.BGI
- HERC.BGI
- IBM8514.BGI
- PC3270.BGI

Existen también tres subdirectorios que contienen un ejemplo de aplicación para cada uno de los algoritmos. El subdirectorio < REDES > incluye los archivos de datos,

pesos y arquitectura de cada uno de los ejemplos; en el subdirectorio < PATRONES > están los patrones necesarios para el entrenamiento de las redes de ejemplo y el subdirectorio < TEXTOS > contiene una pequeña semblanza de cada uno de los algoritmos.

El programa no requiere de instalación, no obstante, es necesario que los archivos ejecutables y los manejadores antes mencionados se encuentren siempre en el mismo directorio.

Los programas fuentes y las utilerías empleadas se encuentran en el subdirectorio < PROGRAMAS >, los programas desarrollados están completamente documentados, mientras que las utilerías propias del lenguaje mantienen la documentación de los autores.

5.3 CONVENCIONES

Las extensiones de los archivos que genera normalmente el simulador se describen a continuación:

* Archivo.DAT es utilizado para almacenar los datos que requiere una red para su entrenamiento.
* Archivo.PES es utilizado para almacenar los pesos de una red previamente entrenada.
* Archivo.ARQ es utilizado para almacenar la arquitectura de la red empleada.
* Archivo.ERR es utilizado para los algoritmos que durante el entrenamiento manejan algún parámetro de error que les permite controlar el aprendizaje.
* Archivo.PAT es utilizado para almacenar los patrones de entrenamiento para las redes.
* Archivo.PRU es generado al hacer una verificación del aprendizaje, contiene los patrones de prueba para la red.

El programa utilizará esta convención por defecto; una vez corriendo el simulador no es necesario escribirlas.

El simulador puede ser operado de dos formas distintas. Si se utiliza exclusivamente el teclado, los distintos menús pueden ser seleccionados oprimiendo la letra mayúscula según sea el caso o con el movimiento del selector empleando las flechas y oprimiendo < INTRO > una vez hecha la selección. Si se cuenta con mouse, la tecla izquierda es en todos los casos el < ESC >, al ser oprimido suprime la selección hecha previamente. La

tecla derecha es el < INTRO > o selector, al ser oprimido sobre alguna opción ésta se lleva al cabo.

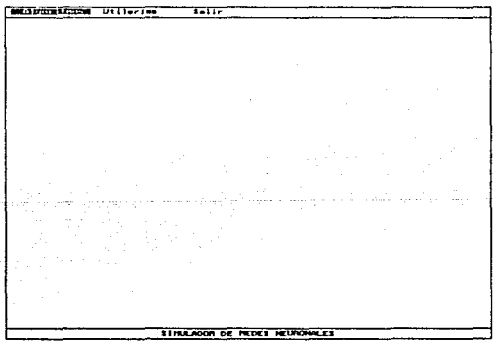
Para interrumpir algún entrenamiento basta con oprimir cualquier tecla; el mouse no interviene durante el entrenamiento.

5.4 MENU PRINCIPAL.

Para iniciar una sesión en el simulador basta con ejecutar el programa SRN:

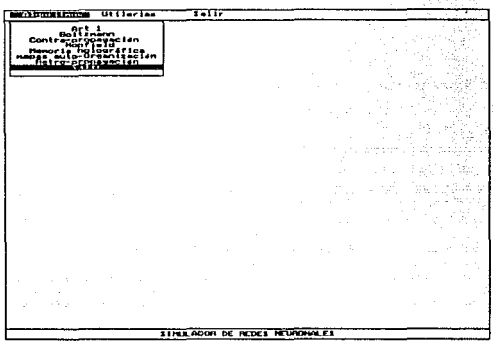
A:\SRN < INTRO >

Una vez activo, el programa presenta una pantalla como se muestra en la figura 5.1. En la parte superior de la pantalla aparece el menú principal del simulador, éste contiene tres posibles opciones a ejecutar: Algoritmos, Utilerías y Salir. La opción activa se muestra en video inverso. En la parte inferior se muestra el título "Simulador de Redes Neuronales".



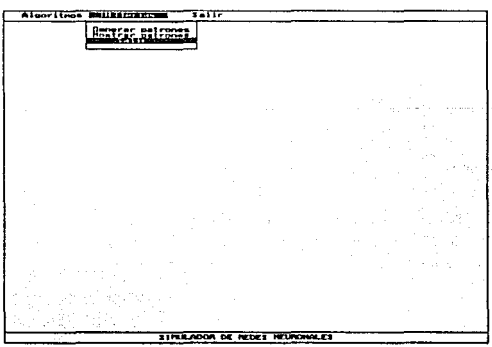
5.1 Pantalla de presentación del simulador

Cuando ha sido seleccionada alguna opción aparece un menú vertical que despliega las posibles tareas que se pueden ejecutar.



5.2 Menú de algoritmos

Al elegir la opción de Algoritmos, se muestra el menú que contiene a los siete distintos modelos de aprendizaje que soporta el simulador, figura 5.2. Al elegir cualquiera de ellos aparece su menú correspondiente, en el cual el nombre del algoritmo aparece como título del mismo.



5.3 Menú de utilerías

Al seleccionar la opción de "Utilerías" se despliegan las opciones de generar patrones y mostrar patrones, figura 5.3. El menú de utilerías está dispuesto de ésta manera porque de acuerdo al uso que se dé al simulador vendrán necesidades posteriores, es decir, utilerías nuevas como generar patrones aleatorios con alguna distribución estadística, capturar patrones mediante alguna tarjeta de adquisición de datos, etc.

5.5 MENU DE ALGORITMOS

El menú de algoritmos contiene cuatro opciones principales que permiten manipular cualquier modelo de red neuronal, el título de la pantalla es precisamente el algoritmo de aprendizaje seleccionado.

Cuando se ha elegido el algoritmo de aprendizaje por emplear y se desea operar con una red neuronal específica es necesario seguir la siguiente secuencia de pasos, de acuerdo con el menú de la figura 5.4:

1) DISEÑO DE LA RED. En este punto se establece la arquitectura de la red que será empleada. Si se ha almacenado una arquitectura particular basta con cargar el archivo de datos para dicha red.

2) PARAMETROS DE LA RED. En general los algoritmos de aprendizaje tienen ciertas variables que definen la ruta del entrenamiento. El simulador tiene para cada algoritmo los valores más comunes de los parámetros, no obstante, si se conoce bien el problema estos parámetros pueden ser definidos con la finalidad de mejorar el aprendizaje.

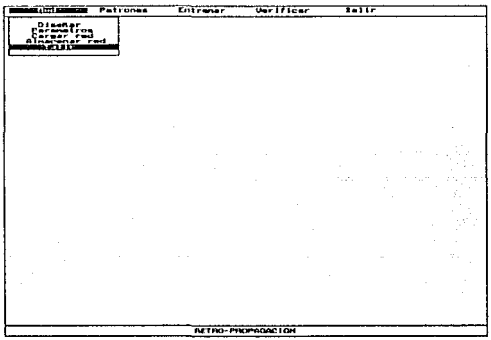
3) PATRONES. Los patrones que debe aprender la red deben de estar previamente diseñados, para definir qué patrones emplear es necesario recuperar el archivo que los contiene desde la opción Patrones en el algoritmo seleccionado.

4) ENTRENAMIENTO. Una vez establecidos los puntos anteriores, la red está lista para aprender. El entrenamiento es necesario para que la red ajuste sus pesos y sea capaz de aprender un conjunto de patrones.

5) VERIFICAR. Si la red ha sido entrenada y se desean realizar pruebas de eficiencia, se captura un patrón de entrada y de forma automática es mostrado a la red para poder observar su respuesta.

5.5.1 RED

La opción de " RED ", como se puede observar en la figura 5.4, contiene lo relacionado con la arquitectura y parámetros de la red neuronal que se ha de utilizar.



5.4 Opción red del menú de algoritmos

Para definir la arquitectura se debe elegir la opción de " Diseñar ", cada algoritmo tiene una arquitectura particular que varía en el número de capas y en la distribución de las neuronas.

5.5.1.A DISEÑAR

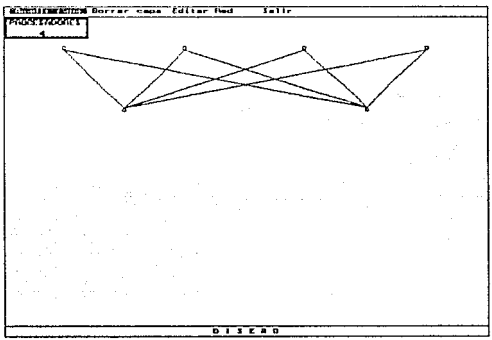
Cuando es seleccionada la opción de diseñar aparece una pantalla titulada " DISEÑO ". Como se puede apreciar en la figura 5.5 el menú tiene tres opciones principales; si se cuenta con mouse sólo son dos opciones principales.

1) Añadir capa.

Al elegir "Añadir capa" aparece un pequeño recuadro en el que se pregunta por el número de procesadores o neuronas que debe tener la red en cada una de las capas. La primera capa o capa de entrada es dibujada en la parte más alta de la pantalla, las capas posteriores se dibujan hacia la parte más baja de la pantalla. Cada capa puede tener

hasta 100 neuronas; en la caso de las redes de Hopfield hasta 120. Para terminar con el proceso de añadir capas es necesario oprimir la tecla < ESC >.

En la figura 5.5 se muestra un ejemplo de "Añadir capa". La red mostrada cuenta con cuatro neuronas en la capa de entrada, y se distinguen con un cuadrado. La capa siguiente tiene dos neuronas y el recuadro de procesadores pregunta por una tercera capa.



5.5 Añadir capa

Existen tres casos particulares en el diseño de una red. El primero es cuando se trata de una red de Hopfield cuya arquitectura es de una sola capa en donde las neuronas están conectadas en un arreglo circular. El segundo, es el caso de las redes de mapas de auto-organización, donde el número de neuronas de la capa de salida debe de ser cuadrado perfecto para que pueda ser formado el arreglo cuadrado de neuronas. El último caso es el de memorias holográficas, donde la cantidad de neuronas de la capa intermedia se determina cuando se conoce el número de patrones que deben ser aprendidos por la red, en este caso se asigna un número de neuronas intermedias que no representa al operador holográfico.

2) Borrar capa.

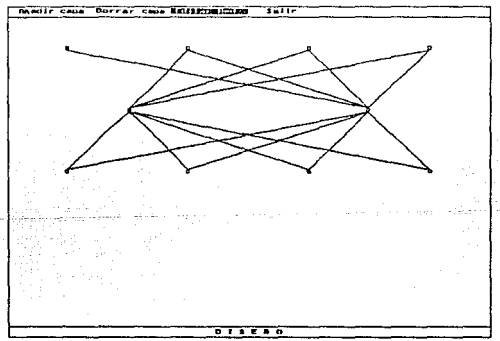
Al elegir la opción de "Borrar capa", la última capa y las conexiones que van de ésta a la capa inmediata anterior se borran.

3) Editar red.

Si se cuenta con mouse, al terminar el proceso de añadir capas, el simulador se pone de forma automática en el modo de edición de red, por lo que esta opción es descartada del menú. En este modo se pueden conectar o desconectar un par de neuronas seleccionándolas al presionar cualquiera de las teclas del mouse.

Si se trabaja con el teclado es necesario seleccionar la opción de "Editar red" para cambiar las conexiones de la red. Una vez dentro del modo de edición la neurona más a la izquierda de la capa de entrada indica la posición del cursor de edición.

Para poder navegar en la estructura de la red hay que emplear las flechas del teclado, cuando se ha elegido una neurona con oprimir el < INTRO > queda seleccionada y permanece en ese estado hasta que se elige otra neurona de la misma capa u otra neurona que no esté en las capas inmediatas superior o inferior.



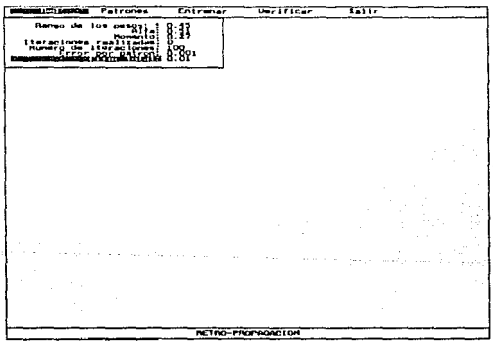
5.6 Editar red

Una vez que ha sido seleccionada una neurona hay que moverse a la capa inmediata superior o inferior para desconectarla o conectarla a otras neuronas. El lazo que une a la neurona seleccionada con la neurona en la que se encuentra el cursor de edición puede ser establecido o removido oprimiendo la tecla < INTRO >. En la figura 5.6 se puede observar que la conexión entre la primer neurona de la capa de entrada y la primer neurona de la capa intermedia ha sido eliminada (Comparar con la figura 5.5).

Al terminar de editar la red se debe oprimir la tecla < F10 > para regresar al menú de diseño. Los distintos algoritmos requieren de una arquitectura mínima para poder operar, si la tarea de diseño es suspendida sin cumplir con este requisito el simulador no tomará en cuenta la arquitectura definida.

5.5.1.B PARAMETROS

Al elegir la opción de parámetros aparece una lista de variables en una ventana como se muestra en la figura 5.7. Esta lista varía de acuerdo al algoritmo y contiene los valores que dirigen el aprendizaje de la red. Estos valores tienen asignada una cantidad por defecto, misma que ha sido elegida de acuerdo a la experiencia y a la literatura pertinente.



5.7 Lista de parámetros

Los valores de las variables pueden ser cambiados utilizando ya sea el teclado o el mouse para elegirlos y asignarles valores numéricos.

El funcionamiento de la ventana de parámetros es similar al de los menús verticales. La opción que se encuentra en video inverso es a la que se le puede cambiar el valor, oprimiendo la tecla < INTRO > o directamente un caracter alfanumérico la lectura del nuevo valor comienza y termina al oprimir la tecla < INTRO >.

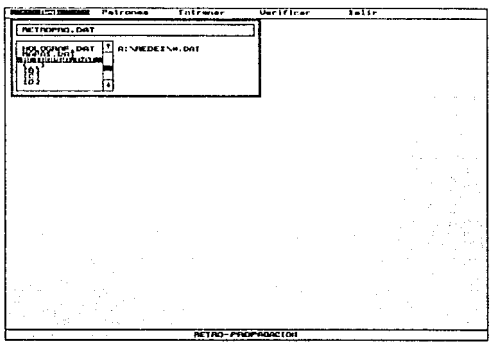
La selección de una variable se puede realizar utilizando las flechas del teclado o posicionando el mouse sobre alguna de las opciones y oprimiendo la tecla derecha.

Cuando se está satisfecho con el valor de todas las variables es necesario oprimir la tecla < ESC > del teclado o la tecla izquierda del mouse para regresar al menú de red. Estos valores se conservan para la red durante toda la sesión.

5.5.1.C CARGAR Y ALMACENAR RED.

Las opciones de cargar y salvar una red tienen el mismo efecto. Como se puede observar en la figura 5.8 aparece una ventana en la que se muestran en un recuadro los archivos que tienen la extensión *.DAT. En la parte superior de la ventana aparece el nombre del archivo que pudiera ser elegido, y a la derecha del recuadro de archivos aparece el subdirectorio en el que se encuentra el archivo.

Si se desea cargar o bautizar una red especifica, al oprimir la tecla de tabuladores el control de la ventana es dirigido a la lectura del nombre del archivo (el nombre no debe incluir la extensión). Si es factible de recuperar o de almacenar la tarea de la ventana ha concluido, en caso contrario, suena una señal de alarma indicando que se ha cometido un error. Si se desea salir de la captura, la tecla de tabuladores regresa el control al recuadro de archivos.



5.8 Ventana de Archivos

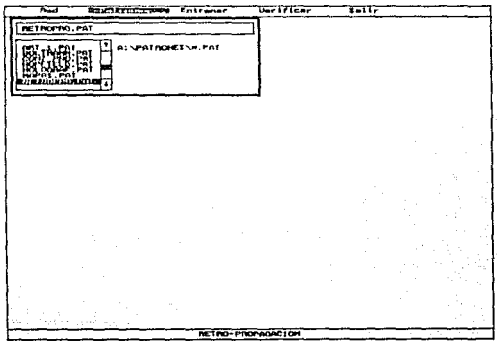
El recuadro de archivos puede ser manipulado como un menú vertical si se emplea el teclado. Las flechas permiten moverse de arriba a abajo en la lista de archivos, al oprimir < INTRO > el archivo, subdirectorio o unidad que se encuentre en video inverso es seleccionado. Si es un archivo válido la tarea de la ventana ha concluido, si se trata de un subdirectorio o de una unidad se muestra una lista nueva de archivos contenidos en el subdirectorio recién elegido.

Si se cuenta con mouse la filosofía es distinta. La tecla izquierda del mouse sirve para moverse sobre la lista de archivos y la tecla derecha sirve para seleccionar el archivo o subdirectorio que se encuentra en video inverso.

En la parte derecha del recuadro de archivos se encuentra una barra que indica el porcentaje de archivos que tiene el subdirectorio actual. Las flechitas de la barra por una parte indican las flechas del teclado y por otra pueden ser utilizadas mediante la tecla izquierda del mouse para desplazar la lista de arriba a abajo.

5.5.2 PATRONES

Una vez que se han establecido la configuración y las características de la red, es necesario cargar el conjunto de patrones que debe de aprender. Los patrones deben de estar forzosamente almacenados en un archivo de caracteres en código ASCII con el formato descrito en la sección de FORMATOS en el Apéndice C.



5.9 Cargar patrones

Al seleccionar la opción de Patrones aparece una ventana igual a la que se utiliza al cargar y almacenar archivos. Como se puede apreciar en la figura 5.9, los archivos que muestra la ventana son aquellos con la extensión *.PAT.

El modo de operar de la ventana de patrones es exactamente el mismo que el utilizado en almacenar y cargar red, con la única diferencia de la extensión para la búsqueda de archivos.

5.5.3 ENTRENAR

Si la red sobre la que se está trabajando cuenta con todos los elementos y parámetros descritos en los puntos anteriores, puede entonces ser entrenada para aprender el conjunto de patrones.

Al seleccionar la opción de entrenar, el simulador ejecuta el programa que responde al nombre del algoritmo. Por esta razón el control sobre el menú y sobre el mouse desaparecen, quedando sólo la opción de detener el proceso de aprendizaje al presionar cualquier tecla.

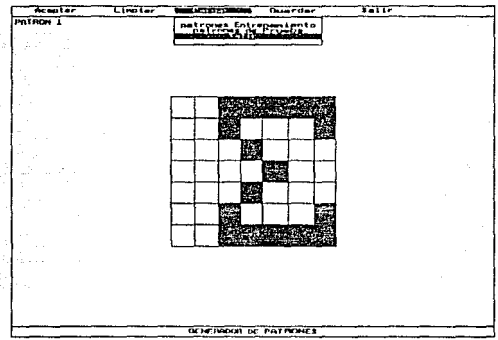
Cuando se realiza la tarea de entrenamiento se muestran en la pantalla la arquitectura de la red y alguna gráfica para el monitoreo del aprendizaje (Véase ejemplo de retro-propagación, página 40).

5.5.4 VERIFICAR

La opción de Verificar sirve para apreciar el desempeño que ha tenido la red después de haber sido entrenada. Cuando es elegida aparece una pantalla de captura para patrones binarios o continuos, según sea el caso.

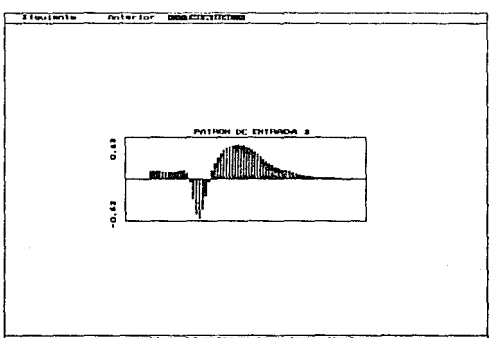
Como se puede apreciar en la figura 5.10, la pantalla de captura cuenta con 4 opciones principales que pueden ser accesadas oprimiendo la tecla < F10 > si se trabaja con teclado o posicionando el mouse sobre alguna de ellas y oprimiendo cualquier tecla. Para editar el patrón es necesario referirse a los puntos 5.6.1 A y B.

La opción de Aceptar toma al patrón recién generado como válido y lo muestra a la red. La opción de Limpiar hace a todos los elementos del patrón iguales con cero y permite editar nuevamente al patrón. La opción de Guardar almacena al patrón recién creado en un archivo que tiene el mismo nombre que tiene el archivo de patrones de entrenamiento pero con extensión PRU.



5.10 Pantalla de captura para patrones binarios

La opción de Cargar despliega un menú vertical con las opciones "patrones de Entrenamiento" y "patrones de Prueba". Estas dos opciones tienen el mismo efecto, solamente definen si se emplea el archivo de patrones de entrenamiento o el archivo de patrones que se ha generado durante las pruebas de eficiencia. La figura 5.11 muestra la pantalla que resulta de elegir alguna de las dos opciones anteriores.



5.11 Visualización de los patrones

Las opciones de Siguiente y Anterior sirven para recorrer los archivos de patrones. En la esquina superior izquierda se indica el número del patrón que está en la pantalla. La opción de Terminar regresa el control al menú de captura.

Al salir de la pantalla de captura se muestran nuevamente las gráficas de la arquitectura y del monitoreo, además mediante alguna herramienta gráfica típica de la literatura de redes neuronales artificiales se muestra el desempeño de la red.

5.6 MENU DE UTILERIAS.

El menú de utilerías está planeado para incluir posteriormente en el simulador algunas herramientas o utilerías que no son distinguibles aún, es decir, de acuerdo al uso que se le dé al simulador surgirán necesidades, sobre todo de análisis, que deberán ser cubiertas y forzosamente anexadas a la versión actual.

Las opciones con las que cuenta actualmente el simulador son generar patrones y mostrar patrones. Como se ha dicho anteriormente los archivos de patrones tienen un formato establecido y los caracteres están en código ASCII, por lo que también se pueden generar de forma externa al simulador. Sin embargo, se ofrece la herramienta para generar patrones análogos o patrones binarios en forma de retícula de manera relativamente fácil.

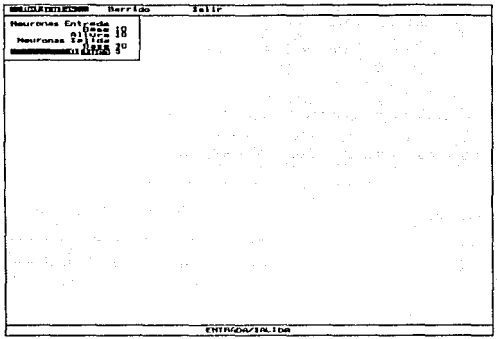
5.6.1 GENERAR PATRONES.

Al elegir la opción de Generar patrones aparece un nuevo menú que ha sido bautizado con el nombre de ENTRADA/SALIDA. Este menú tiene dos opciones principales.

5.6.1.A PATRONES POR RETICULA.

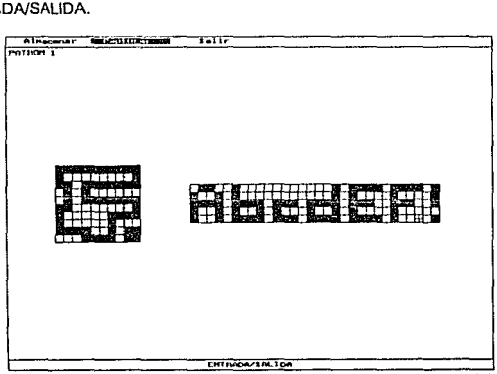
Como se puede apreciar en la figura 5.12, la opción de Retículas abre una ventana en la que se tienen que asentar los valores de la base y de la altura para una o dos retículas.

Para elegir una de las cuatro posibles opciones de la ventana se pueden utilizar las flechas del teclado o la tecla derecha del mouse. Al oprimir un número válido o el < INTRO > sobre alguna opción aparece un cursor que indica la lectura del número. Cuando se ha escrito el número correcto es necesario oprimir el < INTRO >.



5.12 Opción Retículas

Si se desea salir de la ventana al oprimir la tecla de < ESC > o la tecla izquierda del mouse se cierra la ventana de retículas. Si los valores que se introdujeron son válidos aparecerán en la pantalla una o dos retículas y el menú de captura de patrones. Si con los valores introducidos no se puede formar al menos una retícula el control regresa al menú de ENTRADA/SALIDA.



5.13 Edición de patrones reticulares

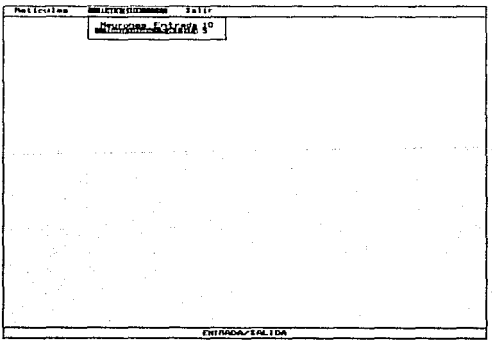
En la figura 5.13 se muestra un caso de captura de patrones binarios por retícula. Cuando se edita un patrón, exclusivamente con el teclado, aparece un cursor de edición en forma de un signo aditivo "+". El cursor puede ser movido de arriba a abajo y de derecha a izquierda con las flechas del teclado.

Para prender o apagar un cuadro de la retícula basta con posicionarse y oprimir la tecla < INTRO >. Si se están editando dos retículas, las teclas < Re. Pág. > y < Av. Pág > cambian el cursor de una retícula a otra. Cuando se han terminado de editar las retículas, la tecla < F10 > regresa el control al menú de captura.

Si se cuenta con mouse, la edición es más sencilla. Basta con posicionarse en algún cuadro de la retícula y oprimir cualquiera de las dos teclas del mouse para prenderlo o apagarlo.

5.6.1.B PATRONES POR BARRIDO

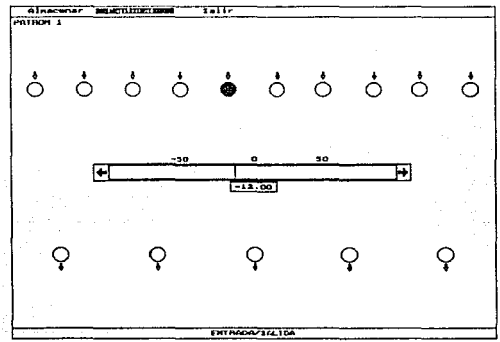
En la figura 5.14 aparece el caso en el que es elegida la opción de Barrido. Cuando es seleccionada se abre una ventana en la que aparecen el número de neuronas de entrada y de salida para el conjunto de patrones análogos que se van a generar.



5.14 Opción Barrido

El número máximo de neuronas de entrada que pueden ser establecidas es de 50 y el número mínimo es de 1. Para las neuronas de salida el máximo es 50 y el mínimo 0. Si los números asentados son válidos al presionar la tecla < ESC > aparece la pantalla de captura, en caso contrario el control regresa al menú de ENTRADA/SALIDA.

La pantalla de captura para patrones por barrido, mostrada en la figura 5.15, esta conformada por una o dos capas de neuronas, según sea el caso, y por una barra numérica para asentar el valor correspondiente a cada neurona. La capa de entrada se sitúa en la parte superior de la pantalla y se indica con flechas el sentido de entrada, en la parte inferior de la pantalla se encuentra la capa de salida, si es que existe, y de igual manera se muestra con flechas el sentido de salida.



5.15 Edición de patrones análogos

Si se cuenta sólo con el teclado para editar el patrón análogo el modo de operación de la pantalla de captura es el siguiente. Para desplazarse de izquierda a derecha en una capa se emplean las flechas del teclado. Para pasar de la capa de entrada a la capa de salida y viceversa se utilizan las teclas de < Re. Pág. > y < Av. Pág. >. Para cambiar el valor del patrón correspondiente a una neurona se puede utilizar el teclado numérico para introducir el número o bien las flechas de arriba y abajo del teclado para desplazar la posición numérica de la barra. La flecha hacia arriba desplaza la posición hacia la parte positiva mientras que la flecha hacia abajo desplaza la posición hacia la parte negativa, ambas en pasos de 0.5 decimales.

Los tabuladores tienen el mismo efecto que las flechas hacia arriba y hacia abajo pero los pasos de cambio en la posición son de 5 números. El tabulador derecho avanza hacia la parte positiva, el tabulador izquierdo hacia la negativa.

Si se cuenta con mouse, las neuronas de cualquiera de las dos capas pueden ser seleccionadas posicionando el mouse y oprimiendo cualquiera de las dos teclas. Para cambiar el valor del patrón se puede utilizar el teclado numérico o posicionar el mouse en alguna región dentro de la barra numérica y oprimir cualquiera de las dos teclas. Las flechas extremas de la barra pueden ser utilizadas con el mouse para desplazar la posición numérica de la barra en pasos de 0.5 decimales.

Al terminar de editar el patrón, se pueden elegir las opciones del menú de captura posicionando el mouse en cualquiera de ellas y oprimiendo alguna de las dos teclas, u oprimiendo la tecla < F10 > del teclado.

5.6.2 MENU DE CAPTURA.

El menú de captura consiste de tres opciones como se puede apreciar en las figuras 5.13 y 5.15. La opción de Almacenar, que incluye al patrón que se encuentra en la pantalla, abre una ventana de archivos que sirve para almacenar el conjunto de patrones recién creados. El modo de operación de esta ventana sigue la misma filosofía que la ventana para almacenar y cargar redes descrita en la sección CARGAR Y ALMACENAR RED de este capítulo.

La opción de Capturar, almacena en un archivo temporal el patrón recién generado y muestra nuevamente la pantalla de captura para generar un patrón nuevo.

Finalmente, la opción de Salir verifica que hayan sido almacenados los patrones, en caso contrario pregunta si deberán ser almacenados o no.

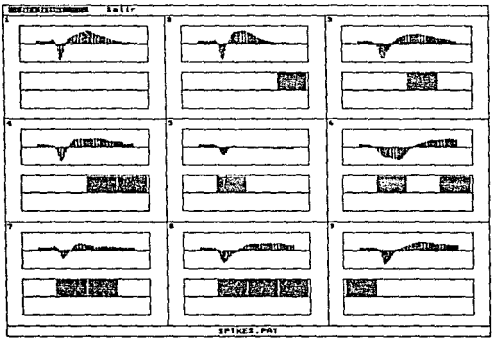
El menú de captura puede ser accesado posicionándose sobre la opción elegida con el mouse y presionando cualquiera de las dos teclas. O con el teclado oprimiendo la tecla < F10 > para que el control este en el menú y se puedan elegir las distintas opciones.

5.7 MOSTRAR PATRONES.

Al elegir la opción de mostrar patrones, aparece un nuevo menú que tiene sólo una opción principal. Al seleccionar ésta opción, Archivo, se abre una ventana que sirve para establecer el nombre del conjunto de patrones que se deben mostrar. El modo de

operación de esta ventana sigue la misma filosofía que la ventana para almacenar y cargar redes descrita en la sección CARGAR Y ALMACENAR RED de este capítulo.

Una vez seleccionado el archivo, se despliegan en pantalla los patrones que lo conforman, y aparece el nombre del archivo como título. En la figura 5.16 se muestra un ejemplo de patrones análogos.



5.16 Despliegue del archivo de patrones SPIKES.PAT

CAPITULO 6

CONCLUSIONES

CONCLUSIONES

Las redes neuronales artificiales han progresado enormemente desde su nacimiento en los años cincuenta. Los avances realizados a lo largo de cuatro décadas han evolucionado desde una arquitectura simple de dos capas de procesadores, que requirió de una implementación electrónica complicada, hasta la simulación de miles de procesadores en una computadora digital. Sin embargo, algunos problemas específicos y otras tantas limitantes quedan aún por resolver.

Es difícil imaginar una aplicación en la que se pueda emplear una red neuronal artificial. Al discutir sobre aplicaciones, es necesario distinguir entre la tarea de identificar donde puede ser utilizada una red neuronal artificial y la tarea de construir la red apropiada, muchas veces combinando diversas arquitecturas. En muchas áreas de aplicación no resulta trivial el hecho de encontrar problemas sin resolver que puedan ser atacados con redes neuronales. Una vez identificado el problema, es una tarea separada construir una red que pueda producir eficientemente las salidas necesarias cuando se presentan los datos de entrada.

Los algoritmos descritos en este documento son componentes de una gran cantidad de sistemas complejos, algunos aún en desarrollo, en los que las redes neuronales hacen las veces de cerebro. El mayor potencial de las redes neuronales está en la densa interconexión de miles de procesadores y en la rapidez con la que pueden realizar distintas tareas. Para eficientar el trabajo de las redes neuronales se han desarrollado distintos componentes computacionales tales como neurochips y tarjetas aceleradoras, además de muchas herramientas de programación. Una aplicación puede bastarse de un simulador como el desarrollado o de una cantidad enorme de equipo, incluyendo periféricos tales como cámaras CCD, sensores, actuadores, brazos robóticos, etc. La complejidad de las aplicaciones va generalmente acompañada de equipo periférico igualmente complejo.

El futuro de las redes neuronales puede beneficiarse enormemente de estudios biológicos, las estructuras encontradas en sistemas biológicos pueden ser la inspiración de nuevas arquitecturas de redes neuronales. El área de desarrollo de las redes neuronales, ahora conocida como Ingeniería Neuronal, encuentra sus bases en modelos conexionistas. Suele confundirse con otras áreas con las que se encuentra estrechamente vinculada, la Neurociencia Computacional es un ejemplo. Las redes neuronales artificiales contienen sólo unas cuantas estructuras biológicas básicas, y frecuentemente emplean estructuras adicionales que no se encuentran en la biología. La Neurociencia

Computacional integra aspectos fisiológicos del sistema nervioso más solisticados y utiliza modelos que intentan reproducir tareas que se llevan al cabo en el sistema nervioso. Encontramos también la Biofísica, área en la que se estudia la actividad neuronal a nivel molecular, es decir, detallan el funcionamiento de la estructura orgánica de la membrana de una neurona y de los canales neuronales.

Aún faltan por solucionar problemas específicos y algunas posibles limitantes. El futuro de las redes neuronales se encaminará a resolver problemas como la representación de gran cantidad de información y limitantes como la capacidad de memoria de las computadoras o la conjunción de muchas neuronas artificiales en circuitos de alta escala de integración. Aplicaciones tales como la visión y el habla requieren de enormes cantidades de datos y de la integración de equipo periférico muy complejo.

El objetivo de esta tesis es la de proporcionar una formación básica en redes neuronales a estudiantes universitarios. Es así que el simulador desarrollado para apoyar el conocimiento, fue diseñado para operar en una computadora personal sencilla. No requiere de mucha memoria y tampoco de coprocesador matemático.

Los alcances del simulador son hasta cierto punto limitados, sin embargo, permite desarrollar aplicaciones propias del usuario con la posibilidad de comprobar resultados. Ocurre frecuentemente que aplicaciones sencillas requieren de una gran cantidad de datos, como consecuencia la apreciación visual se dificulta y el simulador es prácticamente insuficiente. Debido a ésto y a que una aplicación específica requiere también de una representación e interactuación específicas, la persona interesada en resolver algún problema mediante la simulación con redes neuronales, se verá forzada a desarrollar su propio programa. El simulador solo será la base del aprendizaje y el medio para probar distintas opciones de redes neuronales con facilidad.

Los algoritmos que incluye el simulador tratan de abarcar los modelos más comúnmente empleados, los estudiantes universitarios interesados en incursionar en redes neuronales pueden comenzar con este texto y una computadora personal. Si su interés rebasa el alcance del simulador, será necesario adquirir o diseñar programas y equipo periférico para llevar al cabo sus aplicaciones.

La forma de operación del simulador, está pensada para llevar de la mano a personas que tengan un mínimo de conocimiento sobre redes neuronales. El simulador no ha sido probado por un número considerable de personas, sin embargo, se está utilizando en el laboratorio de cibernética y el resultado ha sido satisfactorio. El paquete ha sido depurado extensivamente, por lo que se garantiza su funcionamiento.

La gran variedad de ejemplos y de aplicaciones que se encuentran en la literatura, hacen prácticamente imposible el diseño de un simulador completamente general en el que se puedan reproducir particularidades. No obstante, el funcionamiento del simulador rescata las características fundamentales de cada algoritmo, y los ejemplos escogidos para demostración, que se pueden encontrar en la bibliografía respectiva, hacen evidente el desempeño de las redes neuronales.

El tutorial, que fundamenta la operación del simulador, ha sido releído varias veces y seguido con los ejemplos para comprobar su eficacia en la manipulación del simulador. El simulador es tan amable como es posible en este tipo de temas, queda a los usuarios la palabra final ya que el paquete puede aceptar muchas mejoras.

Recientemente se ha impulsado con gran empuje el desarrollo de las redes neuronales artificiales, la literatura especializada sobre el tema es cada vez más abundante y en todo el mundo se han realizado congresos, cursos y talleres con gran aceptación y asistencia.

Aún falta mucho camino por recorrer antes de que cualquier tipo de robot, verdaderamente inteligente, pueda ser fabricado. Es claro que todo el progreso en el diseño de redes neuronales a la fecha, es producto principalmente del análisis de un contexto o tarea particular que han venido haciendo los investigadores. La intención y el significado de las redes neuronales son dados estrictamente por el diseñador.

APENDICE A

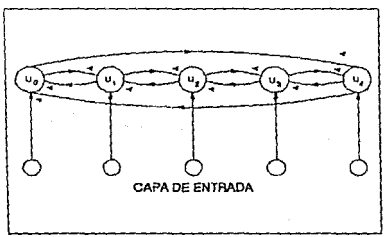
MODELO DE INHIBICION LATERAL

MODELO DE INHIBICION LATERAL

La inhibición lateral es una arquitectura de red neuronal utilizada por los algoritmos de aprendizaje no supervisado para clasificar un conjunto de patrones de entrada categóricamente. El aprendizaje no supervisado, también conocido como aprendizaje competitivo, se lleva a cabo en una red de dos capas de unidades, una capa de entrada y una capa competitiva. En la capa competitiva, las unidades "compiten" entre sí como respuesta al patrón de entrada. La unidad ganadora representa la clasificación categórica del patrón de entrada.

Existen diversas formas de modelar una capa competitiva, entre las más sencillas se encuentra la inhibición lateral. La red de inhibición lateral se representa mediante una sola capa de unidades (arreglo linear), en la que cada unidad interacciona únicamente con las unidades vecinas más cercanas.

Como se puede apreciar en la figura A.1, cada unidad en la red está conectada a sus vecinas con pesos inhibitorios fijos. Una vez que se activan las unidades de la capa, comienza la competencia. Cada unidad trata simultáneamente de mantenerse activa y de evitar que sus unidades vecinas se activen.



A.1 Red de inhibición lateral

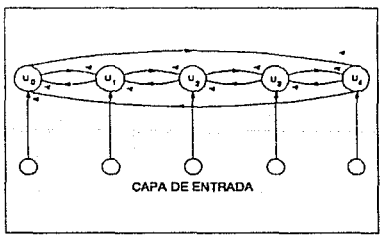
Debido a que los pesos son fijos, la inhibición resulta proporcional a la activación de la unidad. Por consiguiente, la unidad que mejor respuesta tenga al patrón de entrada, es decir, la unidad con mayor activación, inhibe fuertemente a sus unidades vecinas

MODELO DE INHIBICION LATERAL

La inhibición lateral es una arquitectura de red neuronal utilizada por los algoritmos de aprendizaje no supervisado para clasificar un conjunto de patrones de entrada categóricamente. El aprendizaje no supervisado, también conocido como aprendizaje competitivo, se lleva a cabo en una red de dos capas de unidades, una capa de entrada y una capa competitiva. En la capa competitiva, las unidades "compiten" entre sí como respuesta al patrón de entrada. La unidad ganadora representa la clasificación categórica del patrón de entrada.

Existen diversas formas de modelar una capa competitiva, entre las más sencillas se encuentra la inhibición lateral. La red de inhibición lateral se representa mediante una sola capa de unidades (arreglo linear), en la que cada unidad interacciona únicamente con las unidades vecinas más cercanas.

Como se puede apreciar en la figura A.1, cada unidad en la red está conectada a sus vecinas con pesos inhibitorios fijos. Una vez que se activan las unidades de la capa, comienza la competencia. Cada unidad trata simultáneamente de mantenerse activa y de evitar que sus unidades vecinas se activen.



A.1 Red de inhibición lateral

Debido a que los pesos son fijos, la inhibición resulta proporcional a la activación de la unidad. Por consiguiente, la unidad que mejor respuesta tenga al patrón de entrada, es decir, la unidad con mayor activación, inhibe fuertemente a sus unidades vecinas

reduciendo su actividad. El proceso de inhibición o competencia se realiza hasta que casi todas las unidades se apagan y solamente una, la ganadora, se mantiene activa.

El modelo de inhibición lateral puede ser sustituido por un proceso simple que seleccione a la unidad con mayor activación. Sin embargo, en modelos físicos resulta más sencillo implementar el principio de inhibición. Para fines de cómputo, la inhibición lateral puede ser simulada de la manera que se describe a continuación.

Las conexiones entre las unidades de la red, denotadas w_{ij}, tienen pesos fijos y asignados de la siguiente manera:

$$w_{ij} = \begin{cases} 1 & \text{si } i=j \\ -\epsilon & \text{si } i \neq j \end{cases}$$

donde $\epsilon < 1/M$ y M es el número de unidades; i,j=1,2,3,.....,M.

En la red, el proceso de inhibición se repite iterativamente con base en que la actividad de la unidad j en el tiempo discreto "t+1" está dada en términos de las actividades del resto de las unidades en el tiempo "t", mediante la relación:

$$u_j(t+1) = f_t\left(u_j(t) - \sum_{\substack{i=0 \\ i \neq j}}^{M-1} w_{ij}\, u_i(t)\right)$$

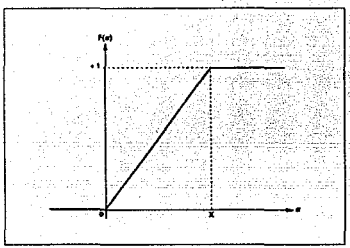
donde:

$$f_t(\alpha) \approx \alpha \text{ si } \alpha > 0$$
$$f_t(\alpha) = 0 \text{ si } \alpha <= 0$$

Si el argumento α de la función de activación $f_t(\alpha)$ es positivo, la actividad $u_j(t+1)$ será también positiva. Todas las unidades para las cuales α es negativo tenderán a la actividad nula.

El efecto de éste proceso es obligar a que la unidad con mayor activación, en relación a las demás unidades, gane. La única que permanezca positiva será la unidad con mayor activación inicial. El proceso se detiene cuando la red obtiene dicha convergencia.

La función de activación de las unidades, frecuentemente referida como función de activación lógica con umbral, en una red de inhibición lateral es no lineal. La gráfica A.2 muestra la forma de dicha función cuando el umbral es cero.



A.2 Función de activación lógica

La inhibición lateral aparece en muchos sistemas biológicos, por ejemplo en la retina, proporcionando una mejor respuesta visual a los bordes y contornos con el consiguiente incremento en el contraste. También aparece en el sistema auditivo, separando y codificando tonos.

APENDICE B

VECTORES ORTONORMALES

METODO PARA OBTENER UN CONJUNTO DE VECTORES ORTONORMALES

Un método sencillo para generar vectores ortonormales es el de las funciones de Walsh. Las funciones de Walsh son las análogas binarias de las funciones seno y coseno. Forman un conjunto de funciones ortogonales en un espacio de 2^m dimensiones.

Las funciones de Walsh pueden ser generadas recursivamente mediante la sustitución del arreglo H en sí mismo:

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Si se desea formar, por ejemplo, un espacio de 4 dimensiones de funciones ortogonales, se sustituye solo una vez el arreglo H en sí mismo:

$$\begin{bmatrix} (1) & \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \quad & (1) & \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ (1) & \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \quad & -(1) & \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{bmatrix}$$

Obteniendo así los vectores:

$$z_0 = (\ 1 \ \ 1 \ \ 1 \ \ 1 \ )$$

$$z_1 = (\ 1 \ -1 \ \ 1 \ -1 \ )$$

$$z_2 = (\ 1 \ \ 1 \ -1 \ -1 \ )$$

$$z_3 = (\ 1 \ -1 \ -1 \ \ 1 \ )$$

Finalmente, para obtener el conjunto de vectores ortonormales deseado, basta con normalizar a cada uno de los vectores z:

$$z_0 = (4)^{-\frac{1}{2}}(\ 1 \ \ 1 \ \ 1 \ \ 1 \ )$$

$$z_1 = (4)^{-\frac{1}{2}}(\ 1 \ -1 \ \ 1 \ -1 \ )$$

$$z_2 = (4)^{-\frac{1}{2}}(\ 1 \ \ 1 \ -1 \ -1 \ )$$

$$z_3 = (4)^{-\frac{1}{2}}(\ 1 \ -1 \ -1 \ \ 1 \ )$$

APENDICE C

CARACTERISITICAS DE PROGRAMACION DEL SIMULADOR

CARACTERISTICAS DE PROGRAMACION DEL SIMULADOR

El simulador está constituido por un programa administrador, cinco programas de utilerías y siete programas independientes que constituyen los siete algoritmos de aprendizaje implementados.

C.1 PROGRAMA ADMINISTRADOR

SRN.PAS

El programa SRN.PAS maneja las distintas pantallas de presentación y los menús de selección. Se le asigna el nombre de administrador porque es el que controla el acceso a los demás programas, verificando cada uno de los parámetros y archivos necesarios para su ejecución.

C.2 PROGRAMAS DE UTILERIAS

REDES.PAS

El programa REDES.PAS se utiliza exclusivamente para generar las distintas arquitecturas de los algoritmos de aprendizaje. Recibe como parámetros, del programa administrador, el número máximo y mínimo de capas y de unidades por capa. Regresa al programa administrador un archivo, con extensión *.ARQ, que contiene las características principales de la arquitectura generada.

I_O.PAS

El programa I_O.PAS sirve para generar archivos de patrones ya sea binarios o análogos. Recibe como parámetros, del programa administrador, el número de unidades de entrada y salida con los que se formarán los patrones. Este programa genera archivos con extensión *.PAT, que contienen los patrones formados.

LIBRERIA.PAS

Este programa contiene una serie de rutinas básicas empleadas por el programa administrador. Por mencionar algunas: lectura de números, reales o enteros, en modo gráfico; manejo de menús tipo pop-up; sonido de alarma; etc.

DIB_PAT.PAS

Este programa se encarga de desplegar en pantalla un archivo de patrones previamente elegido por el usuario.

CORRIDA.PAS

El programa CORRIDA.PAS incluye las rutinas empleadas por cada uno de los algoritmos de aprendizaje. Debido a que los programas de los algoritmos de aprendizaje son independientes, es necesario revisar si los parámetros y archivos son apropiados para comenzar la ejecución del programa, esta tarea se lleva a cabo en el programa CORRIDA.PAS. Además se incluyen rutinas para inicializar gráficos, dibujar arquitecturas en pantalla, dibujar patrones de entrada y salida en pantalla, etc.

C.3 PROGRAMAS DE LOS ALGORITMOS DE APRENDIZAJE

Para utilizar cualquier programa que responda al nombre de un algoritmo basta pasarle como parámetros el nombre del archivo de datos, el nombre del archivo de patrones y si debe de entrenar o reconocer. A continuación se muestra un ejemplo:

A:>HOPFIELD RED_HOP LETRAS ENTRENA < INTRO >

Donde RED_HOP es el nombre del archivo de datos, LETRAS es el archivo que contiene los patrones de entrenamiento y ENTRENA se refiere al aprendizaje.

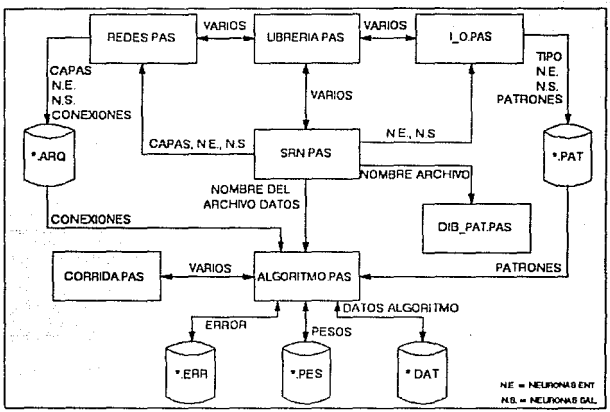
Si se desea realizar una prueba de eficiencia después de entrenada la red, es necesario generar un archivo temporal que incluya el patrón de prueba y pasar "RECONOCE" como parámetro en lugar de entrena:

A:>HOPFIELD RED_HOP LETRAS RECONOCE < INTRO >

Los nombres físicos de los archivos de los algoritmos se muestran a continuación:

NOMBRE FISICO	ALGORITMO DE APRENDIZAJE
ART.PAS	ART-1
BOLTMANN.PAS	MAQUINAS DE BOLTZMANN
CONTRA.PAS	CONTRAPROPAGACION
HOLGRAFI.PAS	MEMORIA ASOCIATIVA HOLOGRAFICA
HOPFIELD.PAS	HOPFIELD
MAPAS.PAS	MAPAS DE AUTO-ORGANIZACION
RETROPRO.PAS	RETROPROPAGACION

En la figura C.1 se muestra el diagrama de flujo de datos del simulador.



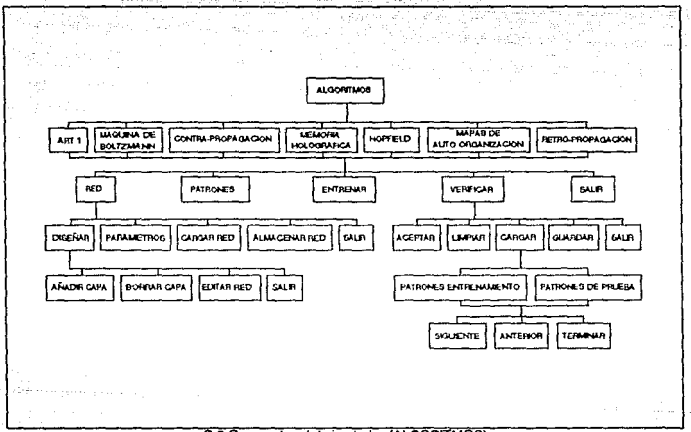
C.1 Diagrama de flujo de datos del simulador

C.4 PRINCIPALES COMANDOS DEL SIMULADOR

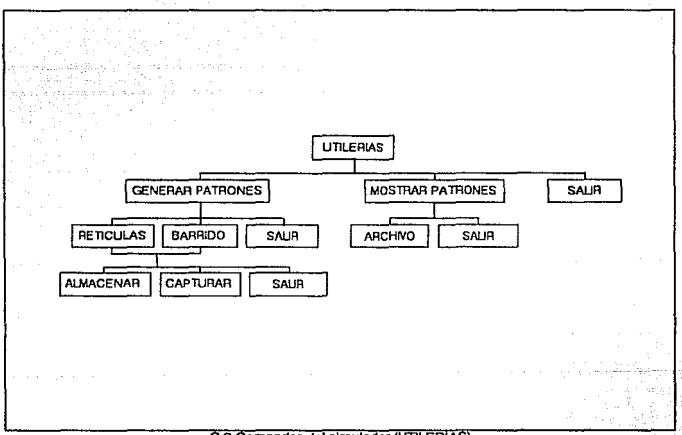
Las figuras C.2 y C.3 muestran los principales comandos con los que cuenta el simulador para su uso. Los nombres que aparecen en la figura son los mismos que se manejan en el simulador.

En la figura C.2 están los comandos que se deben emplear para poder ejecutar un algoritmo de aprendizaje y en la figura C.3 están los comandos que se utilizan para el manejo de las utilerías.

Cada cuadro en las gráficas representa un comando. En la parte inferior de cada comando, unidos con líneas, se encuentran los comandos que de éste se desprenden. Por ejemplo, al seleccionar el comando ALGORITMOS, de la figura C.2, se muestran los siete distintos algoritmos de aprendizaje que contiene el simulador.



C.2 Comandos del simulador (ALGORITMOS)



C.3 Comandos del simulador (UTILERIAS)

C.5 FORMATOS DE LOS ARCHIVOS

En esta sección se define el formato de los archivos que requieren los programas ejecutables de los distintos algoritmos para comenzar un entrenamiento. A excepción de los archivos de pesos y de error, deben estar escritos en código ASCII. Los archivos de pesos y error tienen el formato propio del lenguaje.

ARCHIVO DE DATOS

ART-1	Máquina Boltzmann
'A' (C) Pendiente Proc. (R) Par. Vigilancia (R) Patrones. Clas. (E)	'B' (C) Rango pesos. (R) Alfa. (R) Temp. inicial. (R) Iteraciones. (E) Temp.final. (R) Temples. (E) Estadísticas. (E) Iter. realizadas. (E)
Contra-propagación	**Hopfield**
'C' (C) Rango pesos. (R) Alfa. (R) Beta. (R) Iter. realizadas. (E) Núm. iteraciones. (E) Error total. (R)	'H' (C) Cte. desaprendizaje (R) Patrones aprendidos (E)
Mapas de auto-organización	**Memoria asociativa holográfica**
'O' (C) Epocas realizadas. (E) Epocas a realizar. (E) Epocas de refine. (E) Alfa. (R) Vecindad. (E)	'M' (C) Patrones aprendidos. (E)
Retro-propagación	
'R' (C) Rango pesos. (R) Alfa. (R) Momento. (R) Iterac. realizadas. (E) Núm. de iteraciones. (E) Error por patrón. (R) Error total. (R)	

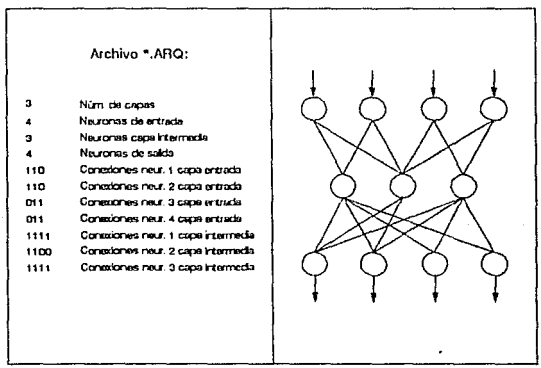
TABLA 5.1

El archivo de datos, cuya extensión es *.DAT, es distinto para cada algoritmo, sin embargo, el formato es el mismo. Cada uno de los datos ocupa un renglón. Al primer renglón se le asigna solo un caracter que indica a qué algoritmo pertenece; el caracter empleado es el mismo que se utiliza para elegir un algoritmo en el menú principal del simulador.

Los demás renglones son números reales o enteros, según sea el caso. Si es un número real debe de incluir la parte entera, el punto y la parte decimal. La tabla 5.1 muestra las variables del archivo de datos para cada algoritmo. Después del nombre, entre paréntesis, se especifica el tipo de dato: (C) caracter, (E) entero y (R) real.

ARCHIVO DE ARQUITECTURA

El archivo de arquitectura, con extensión *.ARQ, contiene la topología de la red que va a ser utilizada. El nombre del archivo debe ser el mismo que del archivo de datos. En éste tipo de archivos, generados en código ASCII, se establecen el número de capas, el número de neuronas por capa y las conexiones de capa a capa.



C.4 Ejemplo de un archivo de arquitecura

El formato del archivo se describe a continuación: El primer dato del archivo es el número de capas que tiene la red, el segundo dato es el número de neuronas de entrada, del tercer dato al establecido por el número de capas se especifica el número de

neuronas para cada capa escondida, inmediatamente después el número de neuronas de salida.

Para establecer las conexiones se utiliza una representación binaria, si la conexión está indicada con 0 entonces no existe, si está indicada con 1 entonces se establece. Después de asignar las neuronas de salida en el archivo, debe indicarse para cada neurona, comenzado de la capa de entrada, las conexiones hacia la siguiente capa. En la figura C.4 se muestra un ejemplo de un archivo de arquitectura y la red generada a partir del archivo.

ARCHIVO DE PATRONES

Existen dos tipos distintos de archivos de patrones, uno para patrones binarios y otro para patrones análogos. El formato de ambos tipos es el mismo en las primeras cuatro líneas. La primera línea indica el tipo de patrones, la segunda el número de patrones que conforma el archivo, la tercera el número de nodos de entrada y la cuarta el número de nodos de salida.

Para los patrones binarios se debe incluir en las líneas siguientes la base y la altura de la retícula de entrada, el ancho y el largo de la retícula de salida y el conjunto de patrones, ocupando un renglón por patrón. En la tabla 5.2 se muestra un ejemplo de archivo para un conjunto de 4 patrones reticulares, entre paréntesis se encuentra la descripción de cada dato.

```
RETICULAS              (Tipo de patrón)
4                      (Número de patrones)
10                     (Nodos de entrada)
4                      (Nodos de salida)
5                      (Ancho retícula de entrada)
2                      (Largo retícula de entrada)
4                      (Ancho retícula de salida)
1                      (Largo retícula de salida)
1101100101             (1er. Patrón de entrada)
0001                   (1er. Patrón de salida)
0010110111             (2o. Patrón de entrada)
0010                   (2o. Patrón de salida)
1110101110             (3o. Patrón de entrada)
0100                   (3o. Patrón de salida)
0000111011             (4o. Patrón de entrada)
1000                   (4o. Patrón de salida)
```

TABLA 5.2

Si la base y la altura de la retícula de salida son cero, lo que implica que solo existe el patrón de entrada, entonces entre los patrones de entrada solo hay un salto de línea y no un renglón vacío.

Para los patrones análogos, después del número de nodos de salida, se debe incluir el conjunto de patrones. Cada número ocupa seis posiciones: el signo, dos enteros, el punto decimal y dos números fraccionarios. Y entre cada número hay dos espacios.

Cada patrón ocupa un renglón y, al igual que los patrones binarios, si no existen nodos de salida entonces entre cada patrón de entrada debe ir sólo un salto de línea y no un renglón vacío.

En la tabla 5.3 se muestra un ejemplo de un archivo de patrones análogos para ocho nodos de entrada y cuatro de salida.

```
BARRIDO                (Tipo de patrones)
10                     (Número de patrones)
8                      (Nodos de entrada)
4                      (Nodos de salida)
   4.19    4.82   0.63   2.78   2.00   3.85  -4.15   4.42
   0.95    0.05   0.95   0.95
  -0.61    3.84  -1.13  -4.35  -2.18  -1.35  -2.52   4.57
   0.95    0.95   0.95   0.95
   0.66   -2.11  -4.20  -3.99   3.17  -0.60   3.82  -0.24
   0.05    0.05   0.05   0.95
  -1.33    2.41   4.75   3.74   1.46   4.91   4.46   3.35
   0.05    0.05   0.95   0.95
  -0.56   -4.18  -4.14   1.74  -2.00  -2.44  -3.83   4.83
   0.05    0.95   0.95   0.05
   1.84    3.05   3.27  -4.35   2.47  -0.70   1.33   0.81
   0.05    0.95   0.05   0.05
  -3.35   -2.94   3.73  -4.30  -2.82  -4.15  -1.75   0.92
   0.05    0.05   0.95   0.05
   0.75   -1.16   4.06  -4.71   1.75  -1.93  -2.36   3.98
   0.95    0.05   0.95   0.05
   0.92    4.19  -4.08   0.10   1.31   0.56  -0.91   4.49
   0.05    0.95   0.05   0.95
   2.01   -0.43   2.96  -3.83   4.67   3.90   0.50   3.30
   0.95    0.95   0.95   0.05
```

TABLA 5.3

Para realizar una prueba de eficiencia es necesario generar un archivo temporal que contenga al patrón de prueba para la red. El archivo debe llamarse forzosamente "TEMP" y el patrón que incluye, sea binario o análogo, ocupa el único renglón del archivo, manteniendo el formato de los archivos de patrones.

BIBLIOGRAFIA

BIBLIOGRAFIA

1.- Anderson James A., Rosenfeld Edward.
Neurocomputing: Foundations of Research.
MIT Press, 1988.

2.- Caudill Maureen.
Neural Networks Primer.
AI Expert, Miller Freeman Publications, 1989

3.- Eberhart Russell C., Dobbins Roy W.
Neural Network PC Tools, A Practical Guide.
Academic Press, San Diego,1990.

4.- Graubard Stephen R.
The Artificial Intelligence Debate.
The MIT Press,1988.

5.- Grossberg Stephen, Kuperstein Michael.
Neural Dynamics of Adaptive Sensory-Motor Control.
Pergamon Press, 1987

6.- Judith Dayhoff
Neural Network Architectures, An Introduction.
Van Nostrand Reinhold, 1990

7.- Kohonen Teuvo
Content-Addressable Memories.
Springer-Verlag, Segunda edición,1987

8.- Kohonen Teuvo.
Self-Organization and Associative Memory
Springer-Verlag, Segunda Edición, 1988.

9.- MacGregor Ronald J., Lewis Edwin R.
Neural Modeling
Plenum Press,1977

10.- McClelland James L., Rumelhart David E. and the PDP Research group.
Parallel Distributed processing: explorations in the microstructure of cognition.
Vol.1: Foundations, MIT Press, 1988

11.- McClelland James L., Rumelhart David E. and the PDP Research group.
Parallel Distributed processing: explorations in the microstructure of cognition.
Vol. 2: Psychological and biological models, MIT Press, 1988

12.- McClelland James L., Rumelhart David E.
Explorations in Parallel Distributed processing.
The MIT Press, Cambridge Massachusetts, 1988

13.- Maren Aliana, Harston Graig, Pap Robert.
Handbook of Neural Computing Applications.
Academic Press, 1990.

14.- Ronald J. MacGregor
Neural and Brain Modeling.
Academic Press, 1987

15.- Simpson Patrick K.
Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations.
Pergamon Press, 1990

16.- Schwartz Eric L.
Computational Neuroscience.
MIT Press, 1990

17.- Tapia Ricardo.
Las células de la mente.
SEP-FCE, 1987

18.- Touretzky David S.
Advances in Neural Information Processing Systems.
Vol. 1, Morgan Kaufmann, California, 1989.

19.- Touretzky David S.
Advances in Neural Information Processing Systems.
Vol. 2, Morgan Kaufmann, California, 1990.

20.- Touretzky, Elman, Sejnowski, Hinton
Connectionist Models, Proceedings of the 1990 Summer School.
Morgan Kaufmann, 1991

21.- Yoh-Han Pao.
Pattern Recognition and Neural Networks
Addison-Wesley,1989