33
24

Universidad Nacional Autónoma de México

FACULTAD DE CIENCIAS

Desarrollo de un Simulador Digital de Redes Neuronales Artificiales Tipo Memoria Asociativa Bidireccional

T E S I S

QUE PARA OBTENER EL TITULO DE

F I S I C O

P R E S E N T A

J. JESUS GONZALEZ FERNANDEZ

MEXICO, D. F. 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

CAPITULO 1
INTRODUCCION

1.1 ANTECEDENTES

Durante mucho tiempo el hombre se ha preocupado por conocer el funcionamiento del cerebro, es decir, los mecanismos con los que puede recordar, razonar, imaginar, diseñar, soñar .... Antiguamente se acostumbraba fabricar muñecos que podían realizar ciertos trabajos como dibujar o bailar por medio de mecanismos de relojería. Estos tipos de muñecos o autómatas repiten su tarea al infinito (si es que no se "gastan"), sin ninguna capacidad de aprendizaje o conocimiento. Los autómatas (s. XVIII) carecían y carecen de algo muy importante: sus creadores no se inspiraron en la biología, es decir, no la tomaron como ejemplo para reproducir lo que un animal puede hacer cuando juega, ¡ y qué mejor ejemplo tomar lo que la evolución ha desarrollado en millones de años ! Cuando mucho reprodujeron la fisonomía de este o de algún monstruo imaginario. El estudio fisiológico de los órganos y principalmente el de los sistemas fisiológicos demostró que el cerebro es el gran órgano controlador de estos sistemas. A su vez este sistema controlador está formado por otros sistemas llamados neuronas [2].

Para Aristóteles la información ("espíritu animal" o "animado" como él lo llamaba, por referirse al ánima o alma) residía en la sangre, la cual era impulsada por el corazón, asiento del alma. Esta idea persistió durante el gran atraso que ha sufrido la humanidad: la llamada Edad Media. El siguiente pensador importante ocupado del tema con ideas originales fue René Descartes, en la primera mitad del siglo XVII, amplió considerablemente este concepto aristotélico de un sistema hidráulico en el que se movía el "espíritu animado". Un siglo

después Luigi Galvani observó que al pasar una corriente eléctrica de un metal a otro, colocando las patas de una rana entre los metales, los músculos de esas patas se contraían, que él llamo "electricidad animal". Posteriormente Galvani demostró que lo mismo ocurría cuando, en vez de la médula espinal, eran los nervios que llegaban a los músculos de la pata los que eran estimulados eléctricamente. Poco después, Alessandro Volta, demostraría que en los experimentos de Galvani la electricidad no se originaba en los tejidos directamente, sino en los metales con que estaban en contacto, y que lo que realmente hacían los nervios era conducir esa electricidad hasta el músculo, provocando que éste se contrajera [33]. Los trabajos de Santiago Ramón y Cajal enseñaron que los elementos de que está formado el sistema controlador llamado cerebro son las neuronas y luego Sherrington indicó cómo funcionan los sistemas de neuronas. De esta manera, un animal es un sistema formado de órganos y a su vez estos órganos son sistemas [2].

1.2 EL CEREBRO

El cerebro es un tejido. Un tejido muy complicado que está compuesto por células -como lo está cualquier tejido- muy especializadas en comunicaciones. Estas células se llaman neuronas. La neurona es la célula nerviosa que transmite impulsos eléctricos por una larga fibra única (el axón) y los recibe de otras neuronas en múltiples fibras cortas (las dendritas). Estas células especializadas funcionan siguiendo las leyes que rigen a todas las demás células. Sus señales eléctricas y químicas pueden detectarse, registrarse e interpretarse, y sus sustancias químicas pueden identificarse. El cerebro está constituído por un número muy elevado de subdivisiones, cada una con una arquitectura y un diagrama de circuito especiales; describir una de ellas no es describirlas todas [29].

El número de células nerviosas, o neuronas, que constituyen los aproximadamente 1350 gramos del cerebro del hombre es del orden de 10^{11} a 10^{20}. Una neurona típica consta de un cuerpo celular, que tiene de cinco a 100 micrómetros (milésimas de milímetro) de diámetro, del que emanan una fibra principal, el

axón, y varias ramas fibrosas, las dendritas. El axón puede producir ramas en torno a su punto de arranque y con frecuencia se ramifica cerca de su extremo. En términos generales, las dendritas y el cuerpo celular reciben señales de entrada; el cuerpo celular u otra región de la neurona las combina y las integra (para decirlo en términos sencillos, las promedia) y emite señales de salida. El axón transporta las señales de salida a los terminales axónicos, que distribuyen la información a un nuevo conjunto de neuronas. El sistema de señales es doble: eléctrico y químico. La señal generada por una neurona y transportada a lo largo de su axón es un impulso eléctrico, pero la señal es transmitida de una célula a otra mediante moléculas de sustancias transmisoras que fluyen a través de un contacto especializado, la sinapsis, entre un suministrador de información (una terminal de axón) y un receptor de información (una dendrita, un cuerpo celular o, a veces, un terminal axónico). Por lo general, una neurona es alimentada por cientos o miles de otras neuronas, y, a su vez, ella alimenta desde una hasta cientos de neuronas. Esto puede bastar para establecer una comparación entre el cerebro y la computadora. Las computadoras han sido inventadas por el hombre, mientras que el cerebro fue creado por evolución. Ambos procesan información y los dos trabajan con señales que se pueden calificar como eléctricas. En las versiones más grandes, ambos tienen muchos elementos; pero no parece ser tan fácil aumentar los elementos de una computadora y por otra parte, las neuronas son mucho más numerosas. Otra diferencia todavía más importante es de orden cualitativo. El cerebro no depende de nada que se parezca a un programa secuencial lineal, mientras que la computadora depende de programas [29].

Por ésta y más razones se ha intentado diseñar neuronas artificiales que conformen las llamadas redes neuronales artificiales y que simulen ciertas operaciones que realizan las redes neuronales biológicas. Actualmente hay una gran polémica sobre los modelos de las redes neuronales artificiales y las biológicas, ésta últimas estudiadas por los electrofisiólogos. Una de las razones la muestra la historia de las redes neuronales artificiales.

A pesar de lo poco que se sabía sobre el cerebro, el fisiólogo mexicano Arturo Rosenblueth y el matemático Norbert Wiener y otros lograron integrar en una disciplina que llamaron Cibernética, los problemas de regulación de los seres vivos y de las máquinas. Así se trató de entender, por primera vez, los sistemas tecnológicos desde el punto de vista de los sistemas biológicos. Así en 1943, Warren McCulloch y Walter Pitts [19] publicaron un modelo matemático de una neurona. Este artículo desató una gran actividad en la investigación sobre los modelos de redes neuronales. El psicólogo Frank Rosenblatt inventó unas redes neuronales artificiales que llamó perceptrones, publicado en su libro Principles of Neurodynamics en 1962, invento que fue rechazado por sus colegas que demostraron que los perceptrones no tenían aplicaciones. Dos de ellos en 1969, Marvin L. Minsky y Seymour A. Papert en su libro Perceptrons [30], cuentan que los perceptrones causaban tal concentraciòn de los recursos para investigación que decidieron revisar detenidamente los resultados publicados. Así demostraron que los perceptrones no podían resolver el problema XOR. Minsky propuso otras alternativas y se creó la inteligencia artificial, por McCarthy. Desde entonces las redes neuronales artificiales fueron casi olvidadas y todo lo que tenía que ver con el modelado y simulación de las redes neuronales artificiales casi desapareció. Las redes neuronales biológicas tuvieron igual suerte debido a su conexión con la Electrofisiología. Así empezó la época del oscurantismo de las redes neuronales. No fue sino hasta principios de los ochenta cuando apareció el conexionismo. J.J. Hopfield y otros atrajeron la atención de muchos investigadores de distintas disciplinas, no sólo de la Matemática, Física y Electrónica, sino también de la Neurobiología, Biofísica, Psicología, Química y otras más. Hopfield retomó el tema de las neuronas formales y los perceptrones y demostró que no habían sido estudiados tan a fondo como Minsky y Papert aseguraron en su libro [2],[30].

1.3 EL CONEXIONISMO

El *Conexionismo* es una rama de la ciencia de la computación donde la arquitectura de una computadora está basada en la utilización de muchos procesadores (antes perceptrones, que se

definirán en el Cap. 2) interconectados entre sí de manera masiva, lo que implica que la computadora trabaja masivamente en paralelo, es decir, que en un momento dado todos los procesadores y todas las conexiones pueden estar activos. El Conexionismo está inspirado en los sistemas nerviosos de los animales ya que estos tienen una gran cantidad de procesadores -las neuronas- y una gran capacidad de comunicación entre ellas por medio de los axones, los árboles dendríticos y las sinapsis. En consecuencia, *una Red Neuronal Artificial se define como un Sistema Dinámico No Lineal Multivariable con Procesamiento Distribuído en Paralelo.* En gran parte, el Conexionismo ha optado por tomar de los sistemas nerviosos la arquitectura en capas, las conexiones masivas y el procesamiento distribuído en paralelo. Los procesadores son tan sencillos como los perceptrones y la organización de las conexiones no está inspirada biológicamente, sino más bien en la utilizada en redes de computadoras. Para construir o simular una red neuronal artificial se necesitan de procesadores, conexiones y un método (algoritmo) de aprendizaje.

1.4 MEMORIA ASOCIATIVA BIDIRECCIONAL (MAB)

Una de nuestras habilidades cognoscitivas consiste en recordar personas, eventos y objetos a partir de información completa, incompleta o confusa. En otras palabras, podemos asociar y recordar así algo que, pudiéramos pensar, habíamos olvidado por completo. Leer por ejemplo algún libro sobre la Independencia de México traería una avalancha de memorias sobre el tema: Don Miguel Hidalgo y Costilla: Alhóndiga de Granaditas: Guanajuato, Gto.: Festival Cervantino, etc...

Esta memoria, llamada asociativa, es una de las sutilezas del cerebro de los seres vivos. Así que no es raro que una de las aplicaciones de redes neuronales artificiales sea simular una memoria asociativa. Se puede decir entonces que *una Memoria Asociativa es una abstracción matemático-computacional de uno de los aspectos de la capacidad cognoscitiva de los seres vivos.*

Teuvo Kohonen, uno de los investigadores que permaneció en la clandestinidad durante la "Edad Media" de las redes

neuronales, ha escrito abundantemente sobre el tema [11]-[13], pero la implementación de memoria asociativa por medio de redes neuronales artificiales ha sido un logro reciente y existen diversas aplicaciones, sobre todo para el reconocimiento y clasificación de patrones borrosos e ilegibles. Se requiere de sensores especializados para transducir una imagen -por ejemplo-, escena o patrón específico en un patrón de entrada a la red neuronal. En muchos casos, la señal transducida tiene que transformarse, condicionarse o comprimirse para que sea adecuada para la entrada de la red neuronal y esta produzca una salida adecuada de acuerdo a su aprendizaje [2]. Un problema para aplicar las redes neuronales es seleccionar o diseñar los sensores de la red para comunicarla con el mundo exterior, además de la conversión, reducción o digitalización de las señales pertinentes.

1.5 OBJETIVO

Debido al enorme interés y falta de conocimiento en estos campos y, también a la dificultad de comprender los paquetes ya elaborados de redes neuronales artificiales existentes en el mercado, la presente tesis es un estudio sobre los diferentes algoritmos para una arquitectura de una Red Neuronal Artificial tipo Memoria Asociativa Bidireccional. Este trabajo hace un estudio comparativo del comportamiento de los algoritmos y sus limitaciones y propone algunos cambios en estos y propone nuevos algoritmos. Se desarrolló un paquete de computación llamado MAB, programado en lenguaje Turbo Pascal Versión 5.0, donde se muestra la implementación y dinámica de varios modelos de memorias asociativas bidireccionales.

CAPITULO 2
ALGORITMOS DE APRENDIZAJE

Se describen los diferentes algoritmos de aprendizaje para una Red Neuronal Artificial tipo MAB que permite almacenar varias asociaciones o patrones. Una asociación consta de un par de patrones definidos como vectores. La propiedad de bidireccionalidad en estas redes permite el flujo de información en dos sentidos. En la primera parte se discuten los algoritmos deterministas, en la segunda parte los algoritmos adaptivos y por último, la Metodología que se utilizó para obtener los resultados del Cap. 3.

2.1 INTRODUCCION A LAS MEMORIAS ASOCIATIVAS BIDIRECCIONALES

Un elemento *procesador* (neurona artificial) está inspirado en una neurona biológica. Este parecido radica en que el procesador recibe múltiples entradas que una a una son ponderadas y luego integradas. El resultado de esta integración es presentado como entrada a una función de activación que produce una respuesta. Hasta aquí el parecido biológico [2].

De esta manera, los componentes de un procesador son el integrador y la función de activación. En la **Fig. 2.1**, se muestra un procesador donde las entradas u_i son ponderadas por los pesos p_i y la integración de todo ello da el potencial total Suma. Esta Suma es presentada a una función no-lineal de activación f(x) que produce la Salida (respuesta) f(Suma) [2]. Si la función de activación fuese lineal, entonces se puede demostrar que la red se reduce a una sola capa (caso de los perceptrones) [30].

Sólo una vez que se ha definido al procesador, se puede hablar de redes neuronales. Una red neuronal artificial es un sistema de procesadores masivamente interconectados en arquitecturas específicas que generalmente son en cascada. La organización de las interconexiones no está inspirada biológicamente, sin embargo, la red en su totalidad es capaz de aproximar tareas como las que realizan las redes neuronales biológicas.

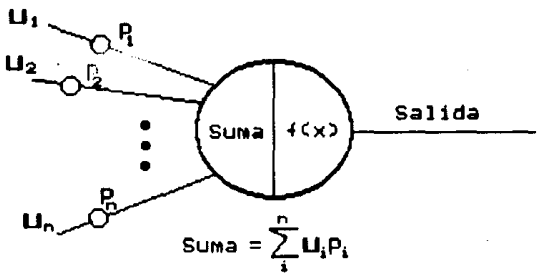


Fig. 2.1 Elemento procesador o neurona artificial.

Tanto en las redes neuronales biológicas, como en las artificiales, su importancia radica en la capacidad para comunicarse unas con otras. Un procesador puede excitar o inhibir a muchos otros y, además, el efecto de un procesador sobre otro está cuantificado o ponderado a la manera que ocurre en las sinapsis neuronales [2].

Las *Memorias Asociativas* son sistemas que pueden recordar o reconstruir patrones almacenados por especificación de todo o una porción de uno de los patrones previamente "almacenados" por un algoritmo de aprendizaje. El aprendizaje de la red está almacenado en las conexiones en la red neuronal. Las características generales de una Memoria Asociativa son: **(a)** almacenar varios pares de patrones (asociaciones), **(b)** realizar lo anterior por medio de un proceso de auto-organización, **(c)** almacenar la información en forma paralelo distribuída, **(d)** obtener una respuesta apropiada al estímulo de entrada y **(e)** generar una respuesta correcta si el estímulo de entrada está incompleto [32].

Los modelos matemáticos de memoria simulados en computadora, en esta tesis, son todos conexionistas. Una red neuronal tipo MAB consta de dos capas de neuronas artificiales -de ahora en adelante se llamarán procesadores- que se interconectan entre sí (sinapsis), en las que no se permite la inhibición lateral (**Fig. 2.2**); es decir, no existen lazos de conexión entre los procesadores de una misma capa. La inhibición lateral no interesa en este tipo de arquitectura. Sean $\{a_1,a_2,\ldots,a_n\}$ y $\{b_1,b_2,\ldots,b_m\}$ los procesadores de las capas A y B, respectivamente. Entonces una matriz M_{nxm} de números reales puede utilizarse como representación de las conexiones entre las dos capas de procesadores (matriz de pesos). Por conveniencia, se utilizarán a_i y b_j para indicar el estado del procesador. Así el elemento p_{ij} de la matriz M_{nxm} representa la conexión sináptica entre a_i y b_j. El signo de p_{ij} determina el tipo de conexión sináptica, será excitatoria si $p_{ij}>0$ e inhibitoria si $p_{ij}<0$. La magnitud de p_{ij} representa entonces la intensidad o fuerza sináptica (peso) de la conexión [14],[16].

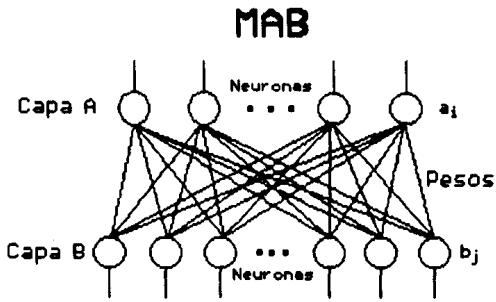


Fig. 2.2 Red Neuronal Artificial tipo MAB.

Así, una asociación consta de un par de patrones que forman un estado de equilibrio dinámico del sistema red neuronal tipo MAB. En resumen, el aprendizaje de la red está determinado por la conectividad entre ambas capas, y ésta conectividad es determinada por los algoritmos para calcular los pesos.

2.2 PRELIMINARES A LOS ALGORITMOS

Todas las memorias asociativas convencionales son unidireccionales. Los patrones, o vectores renglón, $A_1, A_2, \ldots, A_p$ son almacenados en una matriz de pesos M. Cuando un patrón de entrada A (*patrón ruidoso*) es presentado a la red neuronal realizando la multiplicación matricial AM y alguna operación subsecuente, tal como la realizada por la función de activación, se produce una respuesta A'. A' es a su vez presentada a M, la cual produce A", y así sucesivamente. Una memoria estable, más adelante se definirá este concepto, producirá una respuesta fija A^*. Si la memoria es una Memoria Direccionable por Contenido (MDC), entonces A^* será uno de los patrones $A_1, A_2, \ldots, A_p$. Este procedimiento de retroalimentación actúa de la siguiente manera:
$A \rightarrow M \rightarrow A' \rightarrow M \rightarrow A'' \rightarrow \cdots \rightarrow A^* \rightarrow M \rightarrow A^* \rightarrow \cdots$

La MDC unidireccional es autoasociativa. Una porción de un patrón A_k como entrada puede reproducir al patrón completo. De esta manera, las memorias autoasociativas almacenan los pares $(A_1,A_1),(A_2,A_2),\ldots,(A_p,A_p)$. En general, las memorias asociativas son heteroasociativas. Estas almacenan patrones con número de elementos diferentes en cada capa $(A_1,B_1),(A_2,B_2),\ldots,(A_p,B_p)$; donde los A_k y B_k son vectores renglón en distintos espacios, $A_k \epsilon R^n$ y $B_k \epsilon R^m$ y k=1,2,..p.

Considérese la ruta A-->M-->B. Supóngase que A (patrón ruidoso) es más parecido a A_k que a cualquier otro patrón A_f con k,f=1,2,..,p. Supóngase además que M_{nxm} es una matriz de pesos. Pero B<>B_k (B diferente a B_k) y se desearía que mediante algún proceso de retroalimentación se mejorara la precisión hacia la solución esperada. El camino más sencillo es el de multiplicar el patrón obtenido B por una matriz de pesos de mxn, y la más simple, es la matriz transpuesta de M, que es M^t. Esto abre una nueva ruta B-->M^t-->A', en donde posiblemente A' sea más parecido al patrón A_k que A. Se puede entonces aplicar el proceso inverso presentando A' a la matriz de pesos M; A'-->M-->B'. Continuando con este proceso bidireccional se producirán aproximaciones de la asociación almacenada (A_k,B_k): (A,B),(A',B'),(A",B"),... Idealmente, ésta secuencia convergerá rápidamente al par de patrones (A_k,B_k) o a alguna asociación parecida.

Una MAB se comporta como una MDC heteroasociativa si las vías de evolución de la dinámica de la red son:

$$\begin{aligned} &A \dashrightarrow M \dashrightarrow B \\ &A' \dashleftarrow M^t \dashleftarrow B \\ &A' \dashrightarrow M \dashrightarrow B' \\ &A'' \dashleftarrow M^t \dashleftarrow B' \\ &\quad\vdots \\ &A^* \dashrightarrow M \dashrightarrow B^* \\ &A^* \dashleftarrow M \dashleftarrow B^* \\ &\quad\vdots \end{aligned}$$

Así una asociación (A,B) define a un estado estable del sistema Red Neuronal Artificial tipo MAB. Este sistema es no lineal, en el que el análisis de estabilidad se estudia por medio de la Teoría de Estabilidad de Liapunov. Un estado estable del sistema (A,B) puede identificarse con una función de Liapunov (función de energía). En el caso autoasociativo, cuando la matriz de pesos M es simétrica y la diagonal cero, J.J. Hopfield [8]-[10] utiliza la función de energía como $E(A)=-AMA^t$. Bart Kosko [14]-[17] utiliza, de manera más general, la energía de una asociación (A,B) para el caso heteroasociativo como:

$$E(A,B) = - AMB^t \qquad (2\text{-}1)$$

El proceso de recuperación de patrones por la MAB es mediante un procedimiento de retroalimentación. Cada procesador a_i en la capa A y cada b_j en la capa B evolucionan independientemente obteniendo una respuesta de acuerdo a las respuestas de los procesadores en la capa anterior; entonces el estado del sistema MAB cambia si las respuestas de los procesadores cambian. La entrada (suma ponderada e integrada) al procesador b_j es el producto:

$$AM^j = \sum_i^n a_i p_{ij} \qquad (2\text{-}2)$$

donde M^j es la j-ésima columna de M. Y de manera similar, la entrada al procesador a_i es:

$$BM_i^t = \sum_j^m b_j p_{ij} \qquad (2\text{-}3)$$

donde M_i es el(la) i-ésimo(a) renglón(columna) de $M(M^t)$.

Existen diferentes funciones de activación, aquí se utilizaron dos de ellas: la función de activación On-Off y la función de activación Sigmoidal.

Función de Activación On-Off. Tomando el valor cero como el umbral para todas las neuronas, las funciones de activación para los procesadores a_i y b_j se definen como (ver **Fig. 2.3**):

$$a_i = \begin{cases} 1 & \text{si } BM_i^t > 0 \\ 0 & \text{si } BM_i^t < 0 \end{cases} \qquad b_j = \begin{cases} 1 & \text{si } AM^j > 0 \\ 0 & \text{si } AM^j < 0 \end{cases} \qquad (2\text{-}4)$$

de tal manera que si $a_i=1$, se dice el procesador "se activa" -es decir, que excita a todos los procesadores $b_1, b_2, \ldots, b_m$-, de otra manera no lo hace.

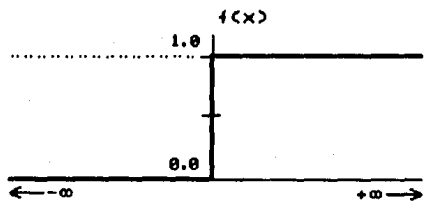


Fig. 2.3 Función de Activación On-Off.

Función de Activación Sigmoidal. Para la función de activación sigmoidal se tiene la siguiente definición:

$$a_i = \begin{cases} 1 & \text{si } f(BM_i^t) > 0.5 \\ 0 & \text{si } f(BM_i^t) < 0.5 \end{cases} \qquad b_j = \begin{cases} 1 & \text{si } f(AM^j) > 0.5 \\ 0 & \text{si } f(AM^j) < 0.5 \end{cases} \qquad (2\text{-}5)$$

con $f(x)=[1+Exp(-(x+\theta)/\theta_0)]^{-1}$. En ésta expresión, el parámetro θ sirve como un umbral, de tal manera que si $f(x)>f(\theta)=0.5$ entonces

el procesador "se activa". El efecto producido por $\theta>0$ es mover la función de activación hacia la izquierda a lo largo del eje horizontal (si $\theta<0$, la mueve hacia la derecha); y el efecto de θ_0, es modificar la forma de la sigmoide. Un valor muy pequeño de θ_0 tiende a una función On-Off; de otra manera, un alto valor de θ_0 dará como resultado una función que varíe suavemente, como se ilustra en la **Fig. 2.4**. El significado de θ es el producir una especie de activación espontánea en los procesadores, en comparación con las neuronas biológicas (cuando $\theta>0$).

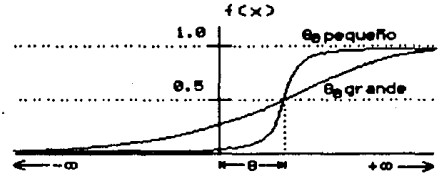


Fig. 2.4 Función de Activación Sigmoidal.

Por otra parte, cuando una asociación (A,B) se presenta a la MAB, los procesadores en ambas capas toman los valores del conjunto {0,1} (vectores binarios). Estos procesadores continúan sus cambios de estado hasta que se encuentra un estado estable (A^*,B^*). Se ha demostrado [16] que para cualquier matriz de pesos M, además de que la energía del estado estable corresponde a un mínimo de la ecuación (2-1). Por lo que se puede afirmar que toda matriz de pesos M es bidireccionalmente estable.

Una asociación (A,B) se puede ver como una implicación lógica, si A entonces B. Sin embargo, la condición de bidireccionalidad en la MAB implica que la asociación (A,B) también representa a la implicación lógica inversa, si B entonces A. Así la relación entre los patrones A y B es simétrica. De aquí que el camino sea memorizar la asociación (A,B) formando una matriz de correlación o producto de vectores. Así la matriz de correlación distribuye la información de la asociación (A,B) en un medio de almacenamiento en paralelo, una matriz.

Los siguientes algoritmos ofrecen varias opciones sobre cómo almacenar varias asociaciones $\{(A_1,B_1),(A_2,B_2),\ldots,(A_p,B_p)\}$ en una matriz de pesos.

2.3 ALGORITMOS DETERMINISTAS

Algoritmo de Bart Kosko [14]-[16]. Una manera de realizar lo anterior es unir las p asociaciones por una suma de matrices de correlación, en la forma:

$$M = \sum_{i}^{p} A_i^t B_i \qquad (2\text{-}6)$$

de tal manera que la matriz M^t es:

$$M^t = \sum_{i}^{p} (A_i^t B_i)^t = \sum_{i}^{p} B_i^t A_i \qquad (2\text{-}7)$$

donde $A_i, B_i \epsilon R^n$ son vectores renglón binarios y $A_i^t (B_i^t)$ son vectores columna con las mismas componentes de los vectores renglón $A_i (B_i)$.

Sin embargo, el esquema aditivo de (2-6) implica que si se utilizan solamente vectores binarios, M no tendrá elementos negativos (inhibición). Así las entradas a los procesadores BM_i^t y AM^j nunca serán negativas y las funciones de activación (2-4) para a_i y b_j serán iguales a 1, una vez que se activen; excepto cuando patrones que se le presenten a la red (iniciales) sean nulos o que la matriz de pesos M sea nula; en cualquier caso, $a_i=b_j=0$.

Los vectores de estado bipolares no producen este problema. Supóngase que (X_k,Y_k) es la versión bipolar de la versión binaria de la asociación (A_k,B_k), es decir, los ceros binarios son reemplazados por unos negativos; dicho de otra manera $X_k=2A_k-I_n$ y $Y_k=2B_k-I_m$, donde $I_n(I_m)$ es el vector renglón identidad en $R^n(R^m)$. Así el elemento ij-ésimo de X^tY tendrá sinapsis excitatoria (+1) si los elementos $x_i \epsilon X$ y $y_j \epsilon Y$ son iguales en signo, o sinapsis inhibitoria (-1) si ambos elementos son distintos en signo. A esto se le llama la correlación del aprendizaje de Hebb. Entonces la matriz de pesos es la suma de matrices del producto externo de vectores bipolares:

$$M = \sum_{1}^{p} X_i^t Y_i \qquad (2\text{-}8)$$

donde $X_i, Y_i \epsilon R^n$ son vectores renglón bipolares y X_i^t es el vector columna con las mismas componentes del vector renglón X_i. Multiplicando y sumando las cantidades bipolares se producen conexiones tanto excitatorias como inhibitorias [14]-[15].

Nótese aquí que al almacenar p vectores binarios $A_1, A_2, \ldots, A_p$ en una matriz de pesos autoasociativa unidireccional, la ecuación anterior se reduce a la matriz simétrica $M=\Sigma X_i^t X_i$, que es el mecanismo de almacenamiento utilizado por Hopfield [8],[10]. Además el par de patrones (A_k, B_k) puede ser "borrado" de M, sumando $X_i^t Y_i^c = -X_i^t Y_i$ donde $Y_i^c = -Y_i$. Es decir, Y_i^c es el complemento de Y_i. Así también $X_i^{ct} Y_i^c = -(X_i^t + Y_i) = X_i^t Y_i$ [14]-[17].

Se ha demostrado que la recuperación por vectores bipolares es más precisa que por vectores binarios, además que el estado estable de la red neuronal tipo MAB ocurre en un mínimo local de la función de energía [23].

Algoritmo de Teuvo Kohonen [11]-[13]. Kohonen ha propuesto utilizar la matriz inversa generalizada de Moore-Penrose o *pseudoinversa* como un mecanismo para construir la matriz de pesos de una memoria asociativa. Aquí la matriz M_{mxn} también es un operador que contiene las asociaciones (A_k, B_k) con k=1,2,..,p y cualquier patrón B_r, r=1,2,..,p, puede ser recuperado por la operación lineal:

$$B_r = MA_r, \; r=1,2,..,p; \; A_r \epsilon R^n \text{ y } B_r \epsilon R^m \qquad (2\text{-}9)$$

donde $A=A^t (B=B^t)$ es un vector columna en $R^n (R^m)$. La ecuación anterior puede ser rescrita en forma compacta, introduciendo las matrices:

$$X_{nxp}=[X_1^t, X_2^t, .., X_p^t] \quad \text{y} \quad Y_{mxp}=[Y_1^t, Y_2^t, .., Y_p^t], \qquad (2\text{-}10)$$

de la forma:

$$Y = MX \qquad (2\text{-}11)$$

donde $X_r(Y_r)$, r=1,2,..,p, es el vector bipolar columna correspondiente al vector binario columna $A_r(B_r)$ [11]-[13],[21]. De tal forma que la matriz de pesos M_{nxm} está dada por:

$$M = M^t = (YX^+)^t \qquad (2\text{-}12)$$

donde X^+ es la pseudoinversa de X.

Algoritmo de Yoh-Han Pao [32]. Yoh-Han Pao proporciona información sobre la construcción de la matriz de pesos M_{nxm}, haciendo una modificación al algoritmo de Kosko [32]. Aquí, la matriz de pesos es:

$$M = \sum_{i}^{p} x_i y_i^t \qquad (2\text{-}13)$$

donde $x_i=X_i/|X_i|$ y $y_i=Y_i/|Y_i|$ de manera que $X_i(Y_i)$ es el vector bipolar correspondiente al vector binario $A_i(B_i)$, con i=1,2,..,p, y $|X_i|(|Y_i|)$ representa la norma del vector renglón $X_i(Y_i)$. Ahora, supóngase que se le presenta a la red un patrón X_k almacenado y considerando la recuperación por vectores bipolares:

$$X_kM = X_k \left(\sum_{i}^{p} x_i y_i^t\right)$$

$$= <X_k,x_k^t>y_k + \sum_{i<>k}^{p} <X_k,x_i^t>y_i \qquad (2\text{-}14)$$

donde el producto escalar de vectores $<X_k,x_k^t>=<X_k,x_k>=\sqrt{p}$ y $<X_k,x_i^t>=\delta_{ki}$ con i=1,2,..,p. Se observa que la recuperación de Y_k será completa si los vectores X_i en (2-13) son ortogonales, para tales patrones $\delta_{ki}=0$ para toda k<>i, y $<X_k,X_k>>0$ si y sólo si $|X_k|>0$. Pao en [32] utiliza x_kM y no X_kM, la razón de utilizar la segunda opción es que $<x_k,x_k>=1$ y $<X_k,X_k>>1$, es decir, que se "amplifica" la recuperación de su patrón más parecido Y_k; incluso aunque se traten de vectores no ortogonales. Supóngase ahora que se le presenta a la red un patrón ruidoso X, que es más parecido al patrón X_k que a cualquier otro patrón X_i. De esta manera, XM=(X_k+Z)M, donde $<X_k,Z>=<x_k^t,Z>=0$, entonces:

$$XM = (X_k+Z)\left(\sum_{i}^{p} x_i y_i^t\right) = (X_k+Z)\left(x_k y_k^t + \sum_{i<>k}^{p} x_i y_i^t\right)$$

$$= <X_k,x_k^t>y_k + \sum_{i<>k}^{p} <X_k,x_i^t>y_i + \sum_{i<>k}^{p} <Z,x_i^t>y_i \qquad (2\text{-}15)$$

En el caso de vectores ortogonales, sólo se tendría un término de degradación sobre el patrón de salida: el último término de la

derecha. En cambio, para vectores no ortogonales se tienen el segundo y tercer término de la derecha en la última ecuación como distorsión sobre el patrón de salida.

Corrección de Hopfield. Hopfield ha propuesto una forma alternativa de cambiar los valores de la matriz de pesos o valores en las conexiones, una vez que han sido determinados por algún algoritmo. En algunos casos, la recuperación de patrones no da la solución esperada. Hopfield mediante esta corrección hace que en la red ocurra un "desaprendizaje" de esa solución no esperada, llamada *solución espuria*. Una vez que la red se encuentra en un estado estable que es solución espuria (A^*,B^*), entonces se aplica la Corrección de Hopfield, haciendo:

$$M_{nueva} = M - \epsilon X^t Y^* \text{ donde } 0<\epsilon<<1 \qquad (2\text{-}16)$$

donde el par de vectores renglón (X^*,Y^*) es la versión bipolar del par de vectores renglón (A^*,B^*) [8]-[10].

Algoritmo de Yoh-Han Pao (Memoria Holográfica) [32]. La técnica de reconstrucción de frentes de onda, conocida como *Holografía*, encuentra numerosas aplicaciones tales como pruebas no destructivas, almacenamiento de datos, etc. Un holograma es un patrón de interferencia que surge de una onda de referencia coherente y una onda esparcida por el objeto (**Fig. 2.5**).

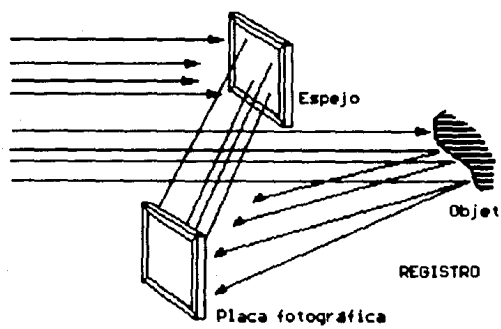


Fig 2.5(a) Registro Holográfico.

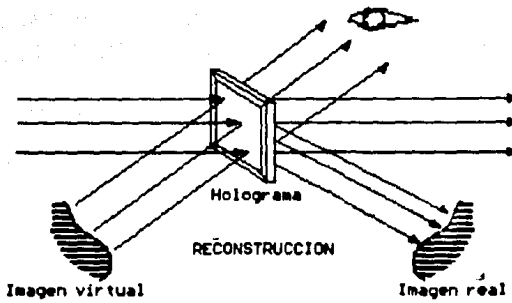


Fig. 2.5(b) Reconstrucción de una imagen.

Las ideas principales que sostienen la construcción de una memoria asociativa holográfica se muestran en las siguientes figuras. La **Fig. 2.6** muestra cómo la interferencia entre el campo de referencia β y un objeto α es registrado sobre una película. En la geometría de la **Fig. 2.6**, los campos β_k son todos ondas planas de la misma longitud de onda, pero difieren en su ángulo de incidencia con respecto al plano registrado; por consiguiente, las frecuencias espaciales son diferentes. Los campos en todos los casos son espacialmente coherentes. Los pares (α_k,β_k) son registrados individualmente, un par a la vez, y la película es producida sólo después de que los pares han sido registrados [32].

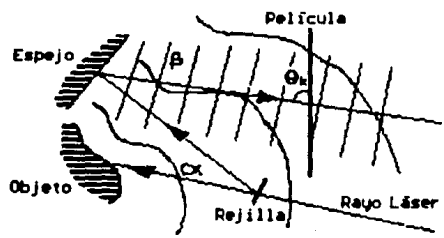


Fig. 2.6 Registro de campos asociados (α_k,β_k).

En la holografía ordinaria, se sabe que, si el holograma es iluminado con un rayo de referencia β_k, entonces el campo α_k es generado desde el holograma provocando una imagen virtual, como se ilustra en la **Fig. 2.7**. El proceso inverso es si el holograma es iluminado con el campo α_k, entonces el campo de referencia β_k es generado. Para éste caso, si los campos de referencia difieren sólo en su ángulo de incidencia, una lente puede ser utilizada para identificar cuál de los campos de referencia ha sido generado por un objeto. Este hecho se muestra en la **Fig. 2.8**. Así se construye una memoria asociativa holográfica presentando un objeto A_k para formar el campo α_k, entonces se unen los campos α_k y β_k. La intensidad del campo $(\alpha_k+\beta_k)$ es registrado sobre la película en el plano del holograma.

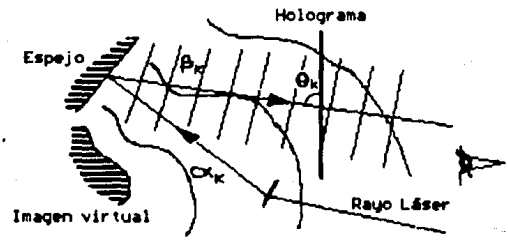


Fig. 2.7 Holografía Ordinaria.

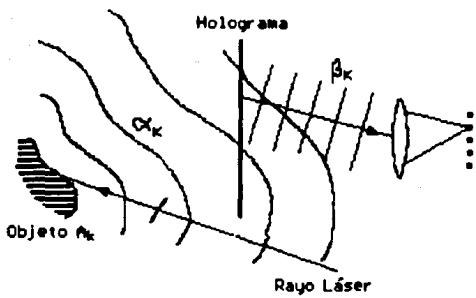


Fig. 2.8 Operación de una Memoria Asociativa Holográfica.

Sea $\{Z_1,Z_2,..,Z_l\}$ un conjunto de vectores renglón ortonormales que generan al espacio euclidiano l-dimensional, y sean M_1 y M_2 dos matrices de pesos dadas por:

$$M_1 = \sum_i^p X_i^t Z_i \qquad (2\text{-}17)$$

$$M_2 = \sum_j^p Z_j^t Y_j \qquad (2\text{-}18)$$

donde el producto escalar de vectores es $<Z_i,Z_j>=0$ y $<Z_i,Z_i>=<Z_j,Z_j>=1$ y como antes $X_i=2A_i-I_n$ y $Y_i=2B_i-I_m$, donde $I_n(I_m)$ es el vector renglón identidad en $R^n(R^m)$.

Comparando con los algoritmos anteriores, aquí existen dos operadores en vez de uno. Supóngase que se le presenta a la red un patrón X (en modo bipolar), la recuperación con el operador M es XM; aquí es XM_1M_2 (M_1M_2 toman el lugar del operador M y la recuperación inversa es $YM_2^tM_1^t$). Supóngase que se le presenta a la red un patrón almacenado X_k, por lo que se forma el producto:

$$X_kM_1 = <X_k,X_k^t>Z_k + \sum_{i<>k}^p <X_k,X_i^t>Z_i \qquad (2\text{-}19)$$

Aquí, se necesita formar una operación de proyección que descomponga X_kM_1 en sus componentes a lo largo de los ejes Z_i y guarde el vector para el cual se obtenga un valor máximo, conservando el signo de Z_i para estos elementos. Es decir, llevar a cabo la operación:

$$r = máx_l\{<Z_l^t,Z_k><X_k,X_k^t> + \sum_{i<>k}^p <Z_l^t,Z_i><X_k,X_i^t>\} \qquad (2\text{-}20)$$

y construir un vector Z; el valor de cada uno de sus elementos es el determinado por la operación anterior. Si se asume que $<X_k,X_k^t>>>X_k,X_i^t>$ para i<>k, entonces se recupera el valor correcto del índice k y se genera el vector $Z=rZ_k$, $0<r\epsilon R$. Si se presenta Z a M_2, entonces:

$$ZM_2 = rZ_k \sum_j^p Z_j^tY_j = r<Z_k,Z_k^t>Y_k = rY_k \qquad (2\text{-}21)$$

se generará una recuperación completa de la solución esperada. Si r=1 la recuperación será perfecta. Cabe aclarar que Yoh-Han Pao no muestra en [32] la bidireccionalidad para este tipo de redes; sin embargo, se puede utilizar el mismo procedimiento utilizado para los algoritmos anteriores, empleando las matrices transpuestas de M_2 y M_1.

Supóngase que se le presenta a la red un patrón ruidoso X, que es más parecido al patrón ruidoso X_k que a cualquier otro patrón X_i. Así $X=X_k+W$, donde $<X_k^t,W>=0$; entonces:

$$XM_1 = (X_k+W)M_1 = <X_k+W,X_k^t>Z_k + \sum_{i<>k}^{p} <X_k+W,X_i^t>Z_i$$

$$= <X_k,X_k^t>Z_k + \sum_{i<>k}^{p} <X_k,X_i^t>Z_i + \sum_{i<>k}^{p} <W,X_i^t>Z_i \qquad (2\text{-}22)$$

Si $<X_k,X_k^t>$ es mayor que los dos términos de la derecha de la ecuación anterior, entonces $Z=rZ_k$; de esta manera se obtiene una recuperación completa dada por (2-21) [32].

Por último, la energía atribuída a una asociación (A,B) para este algoritmo se define como:

$$E(A,B) = - AM_1M_2B^t \qquad (2\text{-}23)$$

Una opción para la construcción de los vectores ortogonales es el generado por la funciones de Walsh. Brevemente [24], las funciones de Walsh forman un conjunto de funciones ortogonales en el espacio 2^r. Así el elemento $z_1(k)$ del vector ortogonal Z_k es:

$$z_1(k)=(-1)^s \text{ con } s=\Sigma_i l_{r-1-i}k_i \qquad (2\text{-}24)$$

donde:

$$(l)_{decimal} = (l_{r-1}\ l_{r-2}\ \cdots\ l_1\ l_0)_{binario}$$
$$(k)_{decimal} = (k_{r-1}\ k_{r-2}\ \cdots\ k_1\ k_0)_{binario} \qquad (2\text{-}25)$$

y $r=\log_2 p$ con $l,k=0,1,..,p-1$. Una vez obtenido el vector ortogonal se normaliza para obtener el vector ortonormal.

2.4 ALGORITMOS ADAPTIVOS

En esta segunda parte se describen los algoritmos adaptivos. En estos métodos la red aprende las asociaciones que le son presentadas secuencialmente en pares entrada-salida. Se parte de una matriz inicial de pesos con valores aleatorios pequeños. Si la salida no es la deseada, se genera una diferencia o error que se minimiza por medio de cambios en los pesos de las conexiones en la red. Esto se hace para cada asociación y, después de un periodo de entrenamiento usualmente largo, tal error se minimiza y esto da por terminada la etapa de aprendizaje de la red. En términos de los pesos se tiene una nueva matriz de pesos resultado de la evolución dinámica a lo largo del proceso del

algoritmo. Estos pesos poseen de manera colectiva el aprendizaje de las asociaciones presentadas.

Algoritmo de Aprendizaje por Señal de Hebb (ASH) [15],[17]. En este algoritmo los pesos cambian de acuerdo a la información disponible que son las asociaciones que se desean aprender. Globalmente, las redes neuronales asocian patrones con patrones. Localmente, las sinapsis se utilizan para asociar señales con señales. La Ley de Aprendizaje por Señal de Hebb correlaciona señales, no activaciones; de forma que la ley de cambios de pesos es:

$$p_{ij}' = - p_{ij} + f(a_i)f(b_j) \qquad (2\text{-}26)$$

donde $p_{ij}'=dp_{ij}/dt$, p_{ij} es el valor de la componente de la matriz de pesos M en el renglón i y columna j, f es la función de activación sigmoidal mostrada en la **Fig. 2.4**. El procedimiento de aprendizaje consiste en, un principio, generar valores p_{ij} aleatoriamente, de tal manera que $-1<p_{ij}<1$ con $p_{ij}<>0$ (p_{ij} diferente de cero). Hasta entonces las asociaciones $\{(A_1,B_1),(A_2,B_2),..,(A_p,B_p)\}$ pueden ser presentadas a la red.

De esta manera, la asociación (A_{k+1},B_{k+1}) puede ser presentada a la red una vez que la red ha aprendido las asociaciones $\{(A_1,B_1),(A_2,B_2),..,(A_k,B_k)\}$ [15],[17]. Se dice que la red ha aprendido una asociación (A,B) cuando éste es un estado estable, es decir, cuando $A=f(BM^t)$ y $B=f(AM)$ donde f es una función de activación y A(B) son vectores renglón binarios en $R^n(R^m)$.

Algoritmo de Aprendizaje Diferencial de Hebb (ADH) [17]. Este algoritmo es una modificación al anterior, de tal forma que correlaciona señales y sus variaciones con el tiempo en la forma:

$$p_{ij}' = - p_{ij} + f(a_i)f(b_j) + f'(a_i)f'(b_j) \qquad (2\text{-}27)$$

donde $f'=df/dt$. De esta manera, se observa que el aprendizaje por señal de Hebb es un caso particular de este algoritmo [17].

Enseguida se presentan cuatro algoritmos adaptivos inspirados en el modelo de *Retropropagación* [32] propuestos por el autor de esta tesis, retomando los principios para minimizar el error del sistema en la recuperación cambiando los pesos de las conexiones, pero aplicado a la red neuronal tipo MAB. En estos algoritmos, a diferencia de los anteriores, se presentan todas las asociaciones para hacer los cambios en los pesos, a cada iteración; siguiendo este proceso hasta que el error sea pequeño y las asociaciones sean aprendidas.

Algoritmo Alg 1. Sean e_i y e_j las entradas a cada procesador en las capas A y B, respectivamente, así:

$$e_i = \sum_{j}^{m} P_{ij} s_j$$

$$e_j = \sum_{i}^{n} P_{ij} s_i \qquad (2\text{-}28)$$

con i=1,2,..,n y j=1,2,..,m y las salidas son:

$$s_i = f(e_i)$$

$$s_j = f(e_j) \qquad (2\text{-}29)$$

donde f es la función de activación sigmoidal definida en (2-5), como se muestra en la **Fig. 2.4.**

Defínase el error relativo a cada asociación como:

$$E_p = \tfrac{1}{2} \sum_{i}^{n} (t_i - s_i)^2 \sum_{j}^{m} (t_j - s_j)^2 \qquad (2\text{-}30)$$

donde t indica la salida deseada, de tal forma, que el error total del sistema es:

$$E = 1/2p \sum_{k}^{p} E_k = 1/2p \sum_{k}^{p} [\sum_{i}^{n} (t_i - s_i)_k^2 \sum_{j}^{m} (t_j - s_j)_k^2] \qquad (2\text{-}31)$$

Hipótesis. Se propone que los cambios de los pesos sean proporcionales a $-\partial E/\partial p_{ij}$, esto es:

$$\Delta p_{ij} = -\mu \, \partial E/\partial p_{ij} \text{ con } \mu \in R \qquad (2\text{-}32)$$

para cada asociación. Se omite el subíndice p por conveniencia.

Cambio de pesos, ruta A-->B. Como el error E, está expresado en términos de las salidas, entonces $\partial E/\partial p_{ij}$ se puede transformar utilizando la regla de la cadena:

$$\partial E/\partial p_{ij} = \partial E/\partial s_j \, \partial s_j/\partial e_j \, \partial e_j/\partial p_{ij} \qquad (2\text{-}33)$$

por lo que:

$$\Delta p_{ij} = -\mu_1 \, \partial E/\partial s_j \, \partial s_j/\partial e_j \, \partial e_j/\partial p_{ij} \qquad (2\text{-}34)$$

pero $\partial E/\partial s_j=(t_j-s_j)\Sigma(t_i-s_i)^2$, $\partial s_j/\partial e_j=f'(e_j)$ y $\partial e_j/\partial p_{ij}=s_i$, entonces:

$$\Delta p_{ij} = -\mu_1 \, (t_j-s_j)f'(e_j)s_i \sum_1^n (t_i-s_i)^2 \qquad (2\text{-}35)$$

donde $f'(x)=[f(x)-f^2(x)]/\theta_0$. Se observa que el cambio de pesos de acuerdo a la salida en la capa B, es proporcional al error que existe en la capa A; si el error es cero, $\Delta p_{ij}=0$.

Cambio de pesos, ruta B-->A. Siguiendo lo mismo que el método anterior se tiene que $\partial E/\partial p_{ij}$ se puede transformar utilizando la regla de la cadena, de la forma:

$$\partial E/\partial p_{ij} = \partial E/\partial s_i \, \partial s_i/\partial e_i \, \partial e_i/\partial p_{ij} \qquad (2\text{-}36)$$

por lo que:

$$\Delta p_{ij} = -\mu_2 \, \partial E/\partial s_i \, \partial s_i/\partial e_i \, \partial e_i/\partial p_{ij} \qquad (2\text{-}37)$$

pero $\partial E/\partial s_i=(t_i-s_i)\Sigma(t_j-s_j)^2$, $\partial s_i/\partial e_i=f'(e_i)$ y $\partial e_i/\partial p_{ij}=s_j$, entonces:

$$\Delta p_{ij} = -\mu_2 \, (t_i-s_i)f'(e_i)s_j \sum_j^m (t_j-s_j)^2 \qquad (2\text{-}38)$$

donde las ecuaciones (2-35) y (2-38) dan la regla de cambios en los pesos de las conexiones.

Siguiendo el método de aprendizaje, el cambio total para cada iteración presentando todas las asociaciones es:

$$\Delta p_{ij} = \sum_k^p (\Delta p_{ij})_k \qquad (2\text{-}39)$$

donde Δp_{ij} es el cambio total en una iteración, p es el número de asociaciones y Δp_{ij} es el cambio en los pesos para cada una de las asociaciones.

Algoritmo Alg 2. Defínase el error relativo a cada asociación como:

$$E_p = \tfrac{1}{2} \sum_1^n (t_i-s_i)^2 + \tfrac{1}{2} \sum_j^m (t_j-s_j)^2 \qquad (2\text{-}30)$$

y considerando la misma hipótesis del algoritmo anterior, con respecto a esta función de error, se tiene que los cambios en los pesos de las conexiones para ambas rutas (A-->B y B-->A) son, respectivamente:

$$p_{ij} = -\mu_1 \, (t_j-s_j)f'(e_j)s_i \qquad (2\text{-}41)$$

$$\Delta p_{ij} = -\mu_2 (t_i - s_i) f'(e_i) s_j \qquad (2\text{-}42)$$

que a diferencia del algoritmo anterior $\delta E/\delta s_j = (t_j - s_j)$ y $\delta E/\delta s_i = (t_i - s_i)$.

Algoritmo Alg 3. Ahora defínase otra función para el error atribuído a cada asociación como:

$$E_p = -\sum_i^n \{t_i \ln s_i + (1-t_i)\ln(1-s_i)\} \sum_j^m \{t_j \ln s_j + (1-t_j)\ln(1-s_j)\} \qquad (2\text{-}43)$$

de tal manera que al considerar la hipótesis inicial, que los cambios de los pesos sean proporcionales a $-\delta E/\delta p_{ij}$, se obtiene:

$$\Delta p_{ij} = \mu_1 \{t_j/s_j - (t_j-1)/(s_j-1)\} f'(e_j) s_i \sum_i^n \{t_i \ln s_i + (1-t_i)\ln(1-s_i)\} \qquad (2\text{-}44)$$

$$\Delta p_{ij} = \mu_2 \{t_i/s_i - (t_i-1)/(s_i-1)\} f'(e_i) s_j \sum_j^m \{t_j \ln s_j + (1-t_j)\ln(1-s_j)\} \qquad (2\text{-}45)$$

que también siguen el método de cambio de pesos como lo indica (2-39). La fórmula de error aquí empleada $\Sigma\{t\ln s + (1-t)\ln(1-s)\}$ se tomó del artículo [22].

Algoritmo Alg 4. Defínase ahora el error atribuído a cada asociación como:

$$E_p = -\sum_i^n \{t_i \ln s_i + (1-t_i)\ln(1-s_i)\} + \sum_j^m \{t_j \ln s_j + (1-t_j)\ln(1-s_j)\} \qquad (2\text{-}46)$$

y considerando las mismas hipótesis anteriores se tiene que:

$$\Delta p_{ij} = \mu_1 \{t_j/s_j - (t_j-1)/(s_j-1)\} f'(e_j) s_i \qquad (2\text{-}47)$$

$$\Delta p_{ij} = \mu_2 \{t_i/s_i - (t_i-1)/(s_i-1)\} f'(e_i) s_j \qquad (2\text{-}48)$$

que también siguen el método de cambio de pesos como lo indica (2-39).

2.5 METODOLOGIA

Con un paquete de computación hecho en TURBO PASCAL Versión 5.0 [37] se programaron todos los algoritmos. La red neuronal utilizada consta de 55 procesadores como se ilustra en la Fig. 3.1. Se hizo el reconocimiento de patrones con variaciones de ruido. Los mosaicos se dibujaron en PAINTBRUSH Versión 2.0 [35].

También se graficó la variación de error a cada cambio en los pesos para los algoritmos adaptivos, utilizando GRAPHPC Versión 2.31 para después guardar en archivo con el paquete FRIEZE de PAINTBRUSH. La simulación se hizo en una computadora GAMA BABY 286 con procesador 80286 a 10 MHz. Utilizando MATLAB Versión 1.51 [36], en una computadora ZENITH con coprocesador 8087, se graficaron las matrices de pesos, de tal forma que se produce una superficie en 3 dimensiones empleando los valores de la matriz como alturas desde un plano XY, que son los renglones y columnas. Los histogramas de correlación cruzada se hicieron empleando el programa CORRELAC del paquete NEURORED [4].

El procedimiento de prueba es el mismo para todos los algoritmos. Este consiste en almacenar asociaciones calculando la matriz de pesos para cada uno de ellos. Las asociaciones son las mostradas en la Fig. 3.1. Las recuperaciones se hacen en los dos modos: binario y bipolar. Para los algoritmos adaptivos se consideraron las mismas condiciones iniciales. La evolución de la dinámica de las redes se muestran en forma de un par de mosaicos. El primer y segundo mosaico corresponde a la primera y segunda capa de procesadores, respectivamente, como en la Fig. 2.2.

El porcentaje de ruido en cada patrón ruidoso para las recuperaciones se calculó utilizando la distancia de Hamming. Se define distancia Hamming [14],[16] como:

$$H(P_i,R) = \tfrac{1}{2}\{N - \sum_{i}^{N} P_i R\} \qquad (2\text{-}49)$$

donde P_i es un vector bipolar de la asociación (A_i,B_i), R es el vector bipolar del patrón ruidoso y N es el número de elementos en los vectores. De esta manera, el porcentaje de ruido con respecto a la asociación (A_i,B_i) es $H(P_i,R)/N$.

CAPITULO 3
RESULTADOS

Los resultados se dividen en tres partes: la primera para algoritmos deterministas y la segunda para algoritmos adaptivos. La última parte trata sobre la comparación de ambos.

3.1 ASOCIACIONES UTILIZADAS

De los algoritmos descritos en el Cap. 2, fueron estudiados los siguientes: los algoritmos deterministas de Bart Kosko (*Kosko*), el de Teuvo Kohonen (*Kohonen*), el de Yoh-Han Pao (*Pao*) y el de Memoria Holográfica (*Holo*), y los algoritmos adaptivos Aprendizaje por Señal de Hebb (*ASH*), el Aprendizaje Diferencial de Hebb (*ADH*) y el algoritmo *Alg 4*, todos con las mismas asociaciones. En la Fig. 3.1, se muestran las cinco asociaciones que sirvieron de prueba para los algoritmos. En esta figura, cada cuadro en el mosaico indica la actividad de un procesador. El mosaico grande es para la capa A y el mosaico pequeño para la capa B, con 35 y 20 procesadores, respectivamente, en una arquitectura como la de la Fig. 2.2. Los cuadros en negro son procesadores activados y los cuadros en blanco son procesadores no-activados.

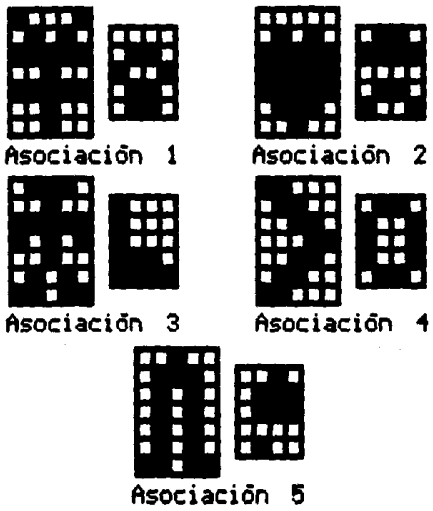


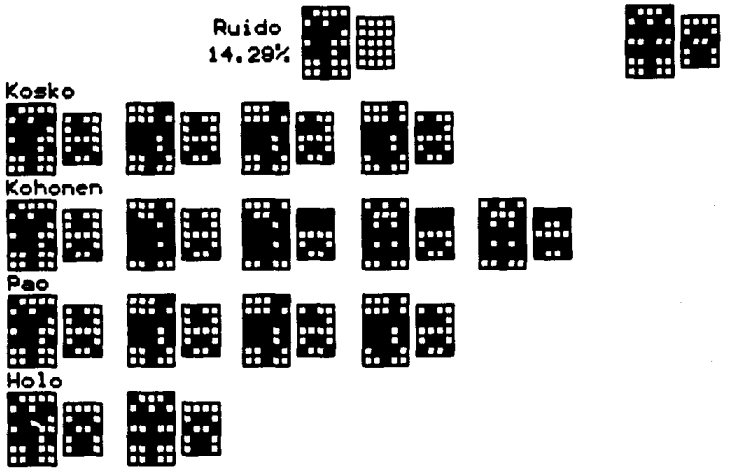
Fig. 3.1 Asociaciones de prueba para los algoritmos. Cada cuadro en el mosaico indica la actividad de un procesador. Los cuadros en negro son procesadores activados y los cuadros en blanco son procesadores no-activados. Asociación 1: Signo de Yen y Casco Espacial, Asociación 2: Corazón y Sombrero y Bigotes, Asociación 3: Robot y Pie del Robot, Asociación 4: Signo de Mayor que ... y Boca y Asociación 5: Bigotes de Yam-yam y Yam-yam y el Genio (Serie de caricaturas de televisión).

En las figuras siguientes se muestran las evoluciones de la dinámica de la red para cada uno de los algoritmos. La red utilizada posee una arquitectura como la mostrada en la Fig. 2.2. La capa A consta de 35 procesadores y la capa B de 20 procesadores. Las conexiones son bidireccionales y de acuerdo a la matriz M. En la parte superior derecha de cada figura se muestra la asociación a la que debería llegar la red para el patrón ruidoso dado, que se encuentra a la mitad de la parte superior y del que se hicieron las variaciones 14.29%, 30.00%, y 35.00%. Los cálculos de los errores se hacen utilizando la definición de Distancia Hamming, para una explicación del significado del ruido ver el Cap. 2. En algunos ejemplos se presentó el patrón ruidoso en la capa A y en otros en la capa B. En la parte inferior derecha se indica el modo de recuperación: binaria o bipolar. La secuencia de iteraciones en la evolución temporal de la red es de izquierda a derecha. El estado final (el par de mosaicos de la derecha) es un estado estable, es decir, a partir de ahí, la red permanece en ese estado. A continuación, se describirán y mostrarán los resultados obtenidos en cada caso.

3.2 ALGORITMOS DETERMINISTAS

En la **Fig. 3.2**, se muestran las evoluciones para los algoritmos de KOSKO, KOHONEN, PAO y HOLO (holográfico) para la asociación 1 de la Fig. 3.1 con 14.29% de ruido. Sólo HOLO encontró, en la segunda iteración, la solución esperada en los dos modos de recuperación. En los otros casos, se observa que en las recuperaciones existen soluciones espurias. Por ejemplo, las recuperaciones y soluciones utilizando KOSKO y PAO son las mismas, a diferencia de la recuperación por KOHONEN.

KOSKO y PAO tienden hacia una solución que se parece a la asociación 2 de la Fig. 3.1 para la recuperación binaria, a diferencia de la bipolar, que casi encuentran la solución esperada con un error del 1.81%. La tendencia de la red a encontrar la asociación 2 de la Fig. 3.1 se debe a que ésta asociación posee mayor energía negativa, de -200, que la energía de la asociación 1 de la misma figura, con -130. KOHONEN en modo binario llega a una solución espuria que no se parece a ninguna de las asociaciones en la Fig. 3.1. En modo bipolar, la recuperación es menos precisa que KOSKO Y PAO, ya que el error es de 18.18%.



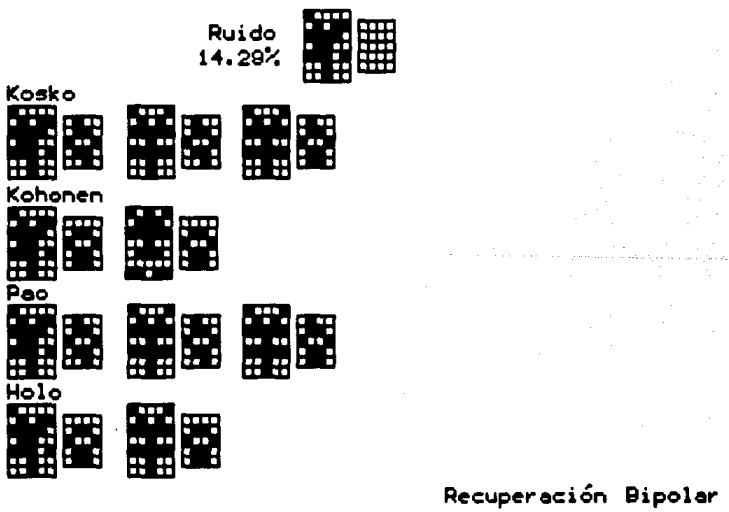
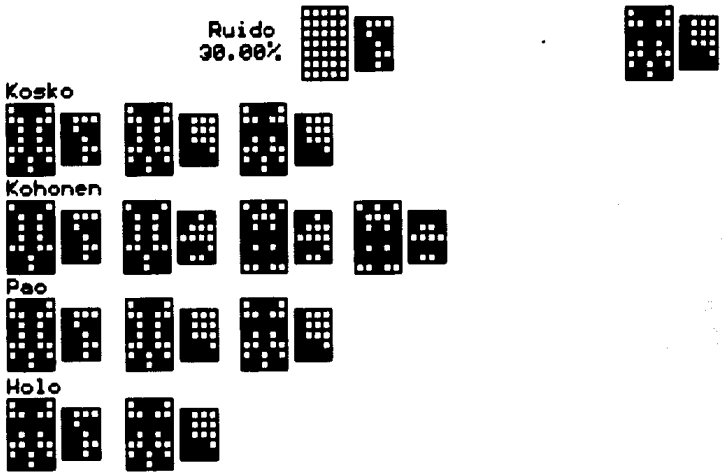


Fig. 3.2 Evolución de la red con 14.29% de ruido.

En las recuperaciones mostradas en las Fig. 3.3 y 3.4, que corresponden a las asociaciones 2 y 3 con un ruido de 30.00% y 35.00%, respectivamente, ocurre casi lo mismo, excepto en los siguientes casos.

En la Fig. 3.3 todos encuentran solución en la recuperación binaria, excepto KOHONEN que cae en una solución espuria con error del 47.27%. Esta solución espuria es la misma para el caso de la Fig. 3.2 en modo binario.

En la recuperación bipolar, KOHONEN encuentra la solución esperada en la segunda iteración, pero no es un estado estable y en la siguiente iteración cae a una solución espuria con error del 7.27%. Mientras que KOSKO, PAO y HOLO encuentran la solución esperada en la segunda iteración.



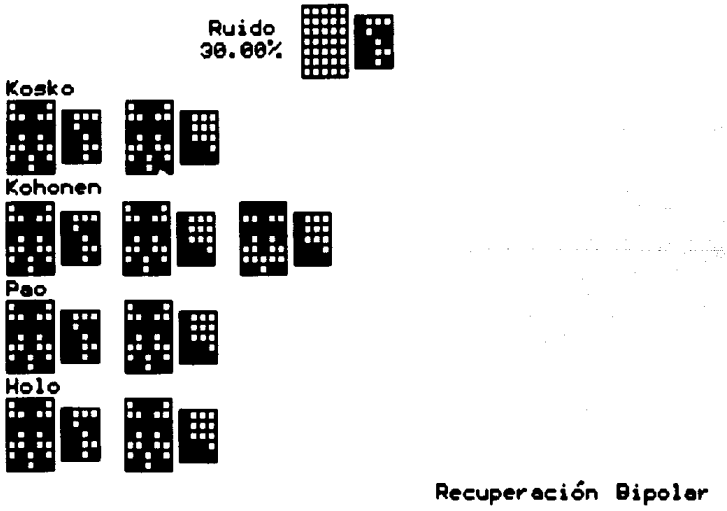
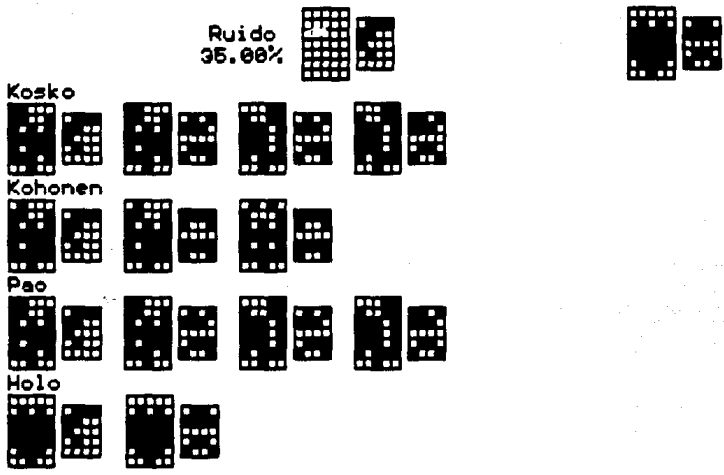


Fig. 3.3 Evolución de la red con 30.00% de ruido.

En la Fig. 3.4, en la recuperación binaria KOSKO, KOHONEN y PAO caen a soluciones espurias, con errores del 27.27%, 32.73% y 27.27%, respectivamente.

Bajo la recuperación en modo bipolar todos los algoritmos encuentran la solución esperada, es decir, la asociación 2 de la Fig. 3.1. HOLO la encuentra en la segunda iteración y KOSKO, PAO y KOHONEN en la tercera iteración.



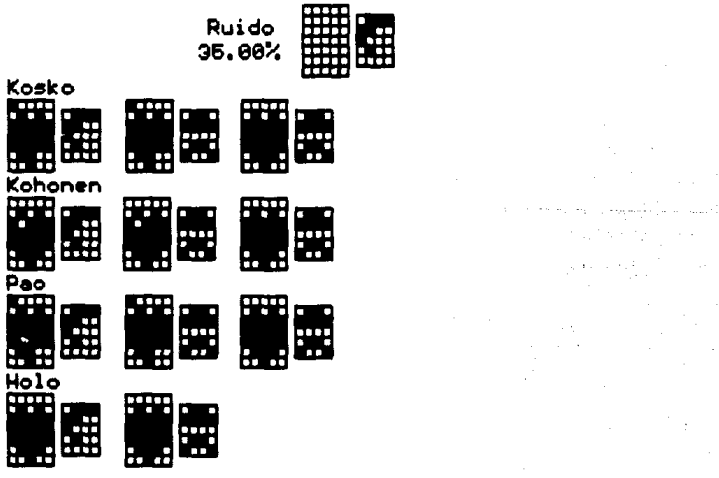


Fig. 3.4 Evolución de la red con 35.00% de ruido.

Se observa que HOLO, para todas las recuperaciones, binarias y bipolares de las Figs. 3.2, 3.3. y 3.4, siempre encuentra la solución esperada. KOHONEN, pareciera que a medida que aumenta el porcentaje de ruido, aumenta también la precisión en la recuperación.

3.2.1 APLICACION DE LA CORRECCION DE HOPFIELD

Por otra parte, en el Cap. 2, se describió la manera de aumentar la precisión en la recuperación aplicando la corrección de HOPFIELD a cada una de las matrices de KOSKO, KOHONEN y PAO. HOLO no es susceptible a esta corrección.

En las Figs. 3.5, 3.6 y 3.7 se muestran las recuperaciones con los mismos patrones ruidosos anteriores, después de haber aplicado la corrección de HOPFIELD. En estas figuras, HOLO se acarrea para completar la comparación en las recuperaciones. Los valores de ϵ para los algoritmos de KOSKO, KOHONEN y PAO fueron de 0.01, 0.001 y 0.001, respectivamente (ver ecuación 2-16).

En la **Fig. 3.5** se observa que después de la corrección, KOHONEN encuentra la solución, en modo binario, y KOSKO y PAO muestran como estado final soluciones espurias con error del 41.82% y 47.27%, respectivamente. En PAO aparece un estado estable que es el opuesto a la asociación 5 (ver Fig. 3.1). A este estado estable se le llama **solución ortogonal espuria**, ya que no corresponde al ortogonal de la solución esperada, al que se le llama simplemente solución ortogonal.

En modo bipolar, todos caen en la solución esperada una vez que se ha hecho la corrección de HOPFIELD. Se observa que las recuperaciones de KOSKO y PAO son diferentes y que la de KOHONEN es más rápida. Por lo que se puede afirmar que la corrección de HOPFIELD aumenta la precisión en la recuperación con ruido del 14.29% para la asociación 1 de la Fig. 3.1, al compararse con las recuperaciones de la Fig. 3.2 sin corrección.

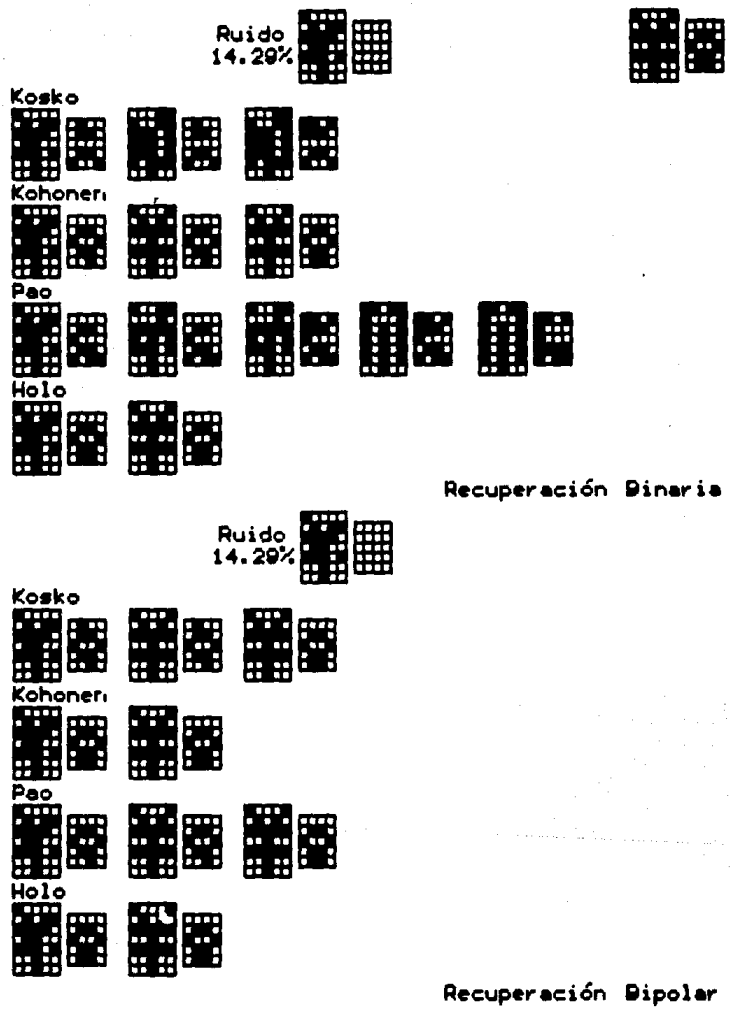


Fig. 3.5 Evolución de la red con corrección de HOPFIELD para KOSKO, KOHONEN y PAO. HOLO se acarrea de las figuras anteriores para completar la comparación. Ruido 14.29%.

En modo binario, en la **Fig. 3.6**, KOSKO encuentra la solución esperada en dos iteraciones. KOHONEN y PAO evolucionan hacia soluciones ortogonales espurias que al inicio parecían encontrar el estado solución en la segunda iteración.

En modo bipolar, KOSKO y KOHONEN casi encuentran la solución con error de 3.64% para ambos, y PAO, que al igual de los recuperaciones anteriores que en la segunda evolución encuentran la solución esperada, cae a una solución ortogonal espuria hasta la sexta iteración.

La corrección de HOPFIELD, para ruido del 30.00% respecto a la asociación 3 de la Fig. 3.1, no hace más precisa las recuperaciones para KOSKO y PAO, sino que aumenta el error. Para KOHONEN el error en la recuperación disminuye del 7.27% al 3.64% en el modo bipolar.

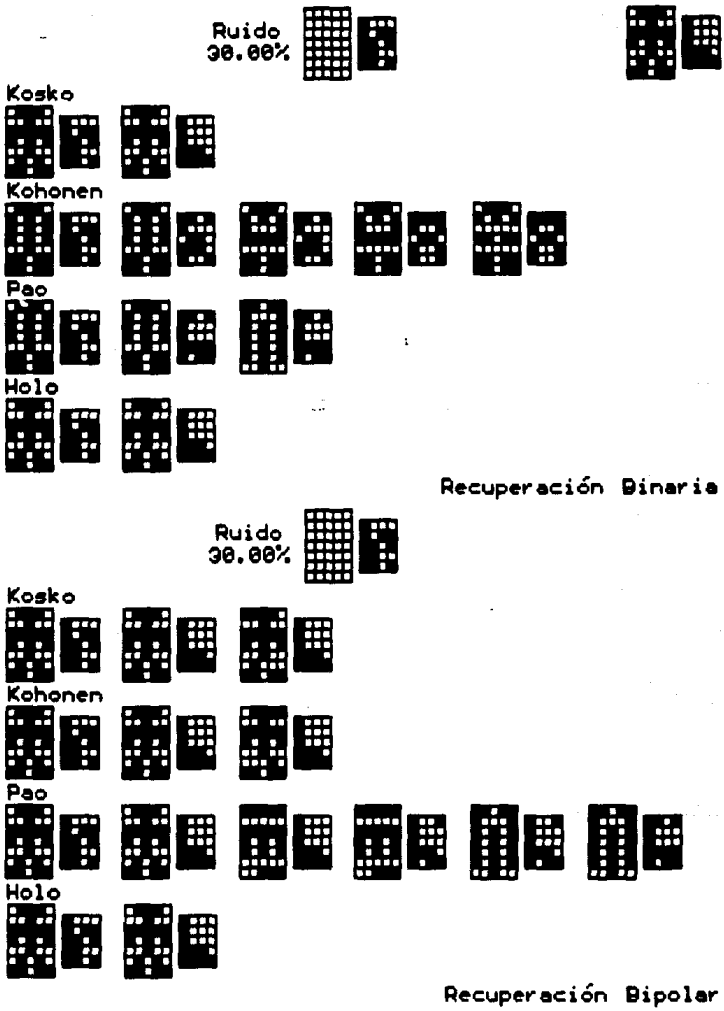


Fig. 3.6 Evolución de la red con corrección de HOPFIELD para KOSKO, KOHONEN y PAO. HOLO se acarrea de las figuras anteriores para completar la comparación. Ruido 30.00%.

En la **Fig. 3.7**, en modo binario, la red encuentra soluciones espurias, de las que son ortogonales para KOHONEN y PAO con error de 67.27% y 50.90%, respectivamente; para KOSKO el error es del 34.54%.

En la recuperación en modo bipolar, KOSKO obtiene una solución espuria con 18.18% de error y dos soluciones no esperadas, para KOHONEN y PAO, que corresponden a las asociaciones 4 y 1 de la Fig. 3.1, respectivamente. El ruido asociado a estas últimas soluciones es del 45.00%. La convergencia hacia soluciones no esperadas se explica con base en las energías. Para KOHONEN, la energía de la asociación 4 es de -2.97, y la energía de la solución esperada es de -2.84. Además la energía de la asociación 4 es la mayor de todas, por lo que la red tiende a encontrar la solución con mayor energía negativa. Caso similar ocurre con PAO, la energía de la solución no esperada (asociación 1) es de -3.17 y la esperada de -3.05.

Se observa en estas últimas recuperaciones que la corrección de HOPFIELD puede ser efectiva para algunos patrones y aumentar la precisión en la recuperación. Sin embargo, para otros patrones ruidosos puede ser menos precisa y llegar a borrar alguna asociación ya aprendida.

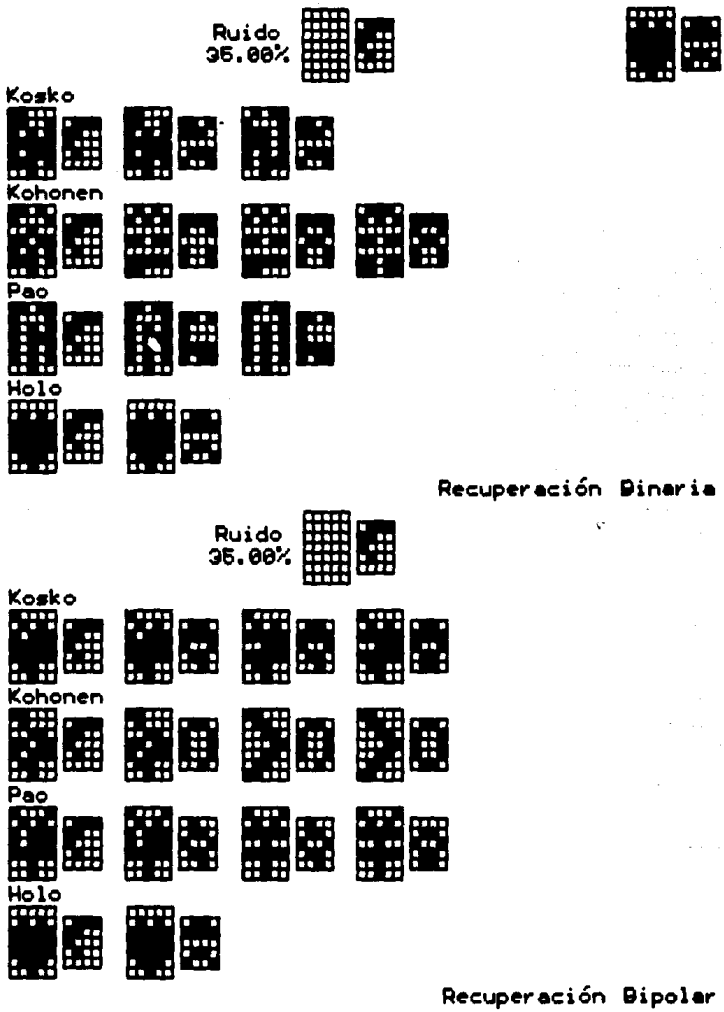


Fig. 3.7 Evolución de la red con corrección de HOPFIELD para KOSKO, KOHONEN y PAO. HOLO se acarrea de las figuras anteriores para completar la comparación. Ruido 35.00%.

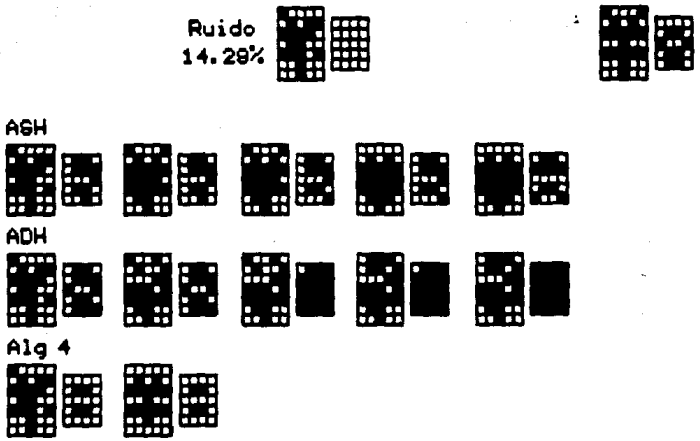
3.3 ALGORITMOS ADAPTIVOS

Para los algoritmos adaptivos se hizo el mismo análisis de los algoritmos deterministas. Se consideró una matriz inicial de pesos aleatorios, que con cada uno de los algoritmos adaptivos descritos en el Cap. 2, se llegó a una matriz final de pesos. Los resultados que se muestran a continuación son los de algoritmos adaptivos de KOSKO: Aprendizaje por Señal de Hebb (ASH) y el Aprendizaje Diferencial de Hebb (ADH), y el cuarto algoritmo propuesto por el autor de esta tesis (Alg 4).

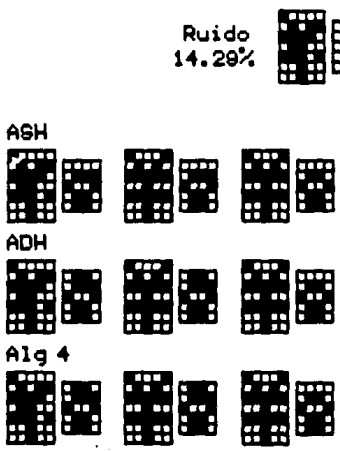
En las Figs. 3.8, 3.9 y 3.10, se muestran las recuperaciones con los mismos patrones ruidosos anteriores, después de que la red hubo aprendido las asociaciones (después del proceso de aprendizaje).

En la **Fig. 3.8**, en modo binario, ASH obtiene una solución no esperada (asociación 2, Fig 3.1), esto se debe a que esta solución tiene una energía de -76.82, mayor que la solución esperada de -63.39; para ADH y Alg 4 la red obtiene soluciones espurias con error de 49.09% y 12.73%, respectivamente. Nuevamente, ADH trata de obtener, igual que ASH, la misma solución no esperada.

En la recuperación bipolar, se obtiene la solución esperada para todos los algoritmos. Se observa además que se encuentra la solución en la tercera iteración y que la evolución es ligeramente diferente.



Recuperación Binaria



Recuperación Bipolar

Fig. 3.8 Evolución de la red con 14.29% de ruido.

En la **Fig. 3.9**, bajo la recuperación en modo binario, todos obtienen soluciones espurias. ASH obtiene una solución ortogonal espuria. Los errores en la recuperación para ASH, ADH y Alg 4 son de 45.45%, 47.27% y 36.36%, respectivamente.

En modo bipolar, ASH y Alg 4 encuentran la solución esperada, ASH en 5 iteraciones y Alg 4 en 3 iteraciones. ADH obtiene una solución espuria con error de 49.09%.

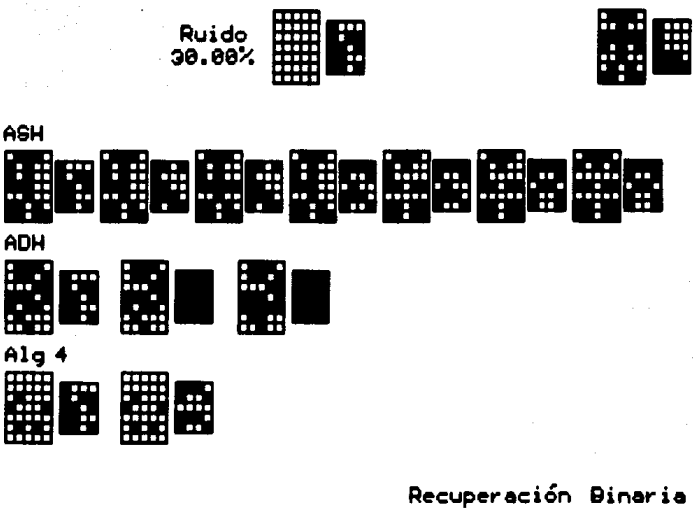
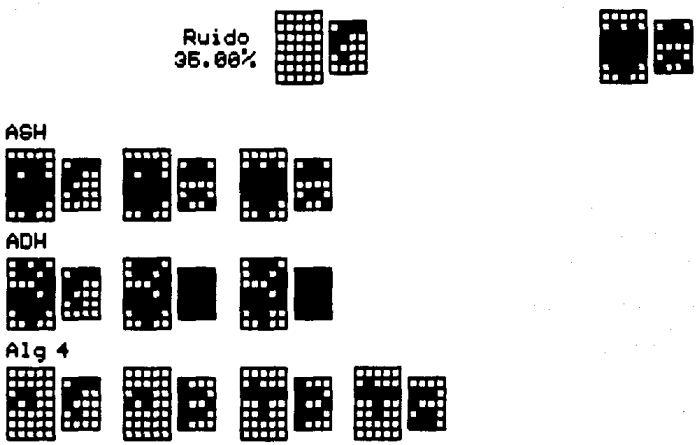


Fig. 3.9 Evolución de la red con 30.00% de ruido.

En la recuperación binaria de la **Fig. 3.10**, sólo ASH obtiene la solución esperada en 3 iteraciones. ADH y Alg 4 encuentran soluciones espurias con 38.18% y 36.36% de error, respectivamente.

En la recuperación bipolar, nuevamente sólo ASH encuentra la solución esperada. ADH obtiene una solución no esperada y Alg 4 una solución espuria con error del 47.27%.

En estas figuras se observa que en la recuperación de los patrones aparecen, nuevamente, muchas soluciones espurias. El algoritmo ASH obtiene la solución esperada en modo bipolar en todas las recuperaciones. El segundo en mayor precisión es el Alg 4.



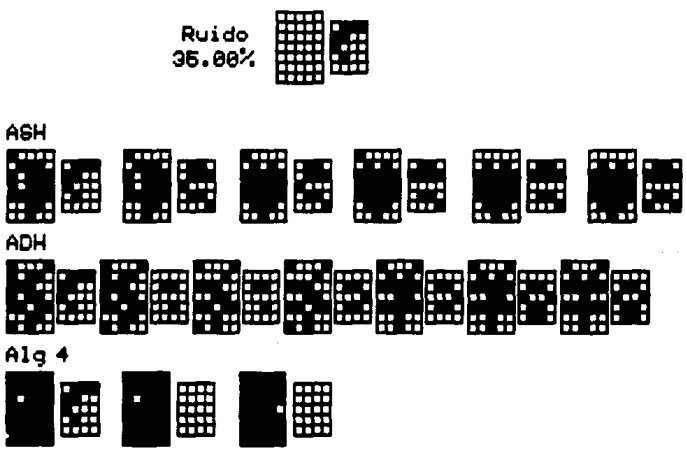


Fig. 3.10 Evolución de la red con 35.00% de ruido.

3.4 COMPARACIONES DE LOS ALGORITMOS

En las figuras siguientes se intenta hacer una comparación sobre los algoritmos de aprendizaje aquí expuestos, observando las recuperaciones bajo un mismo patrón ruidoso con 14.29%, 30.00% y 35.00% de ruido.

En la **Fig. 3.11** se muestra la evolución de la red con recuperación en modo binario con 14.29% de ruido. Se acarrearon las recuperaciones con la corrección de HOPFIELD para los algoritmos deterministas, en vista de que las recuperaciones son más precisas (Fig. 3.5). KOSKO, ADH y Alg 4 obtienen soluciones espurias con errores de 41.82%, 49.09% y 12.73%, respectivamente, PAO converge a una solución ortogonal espuria y ASH obtiene una solución no esperada que tiene mayor energía que la esperada. En cambio, KOHONEN y HOLO encuentran la solución esperada.

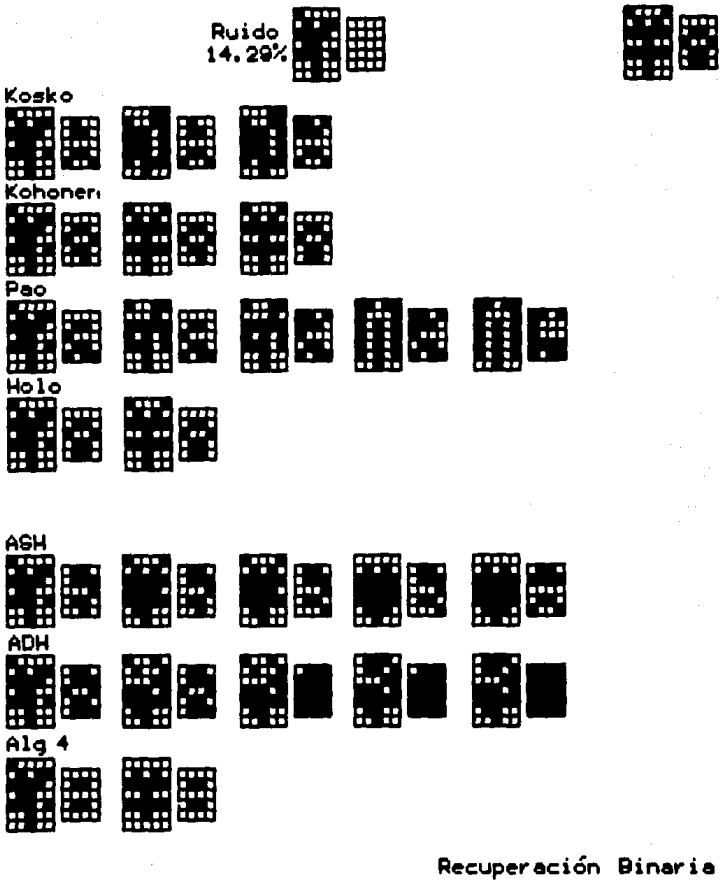


Fig. 3.11 Evolución de la red para todos los algoritmos con recuperación en modo binario y 14.29% de ruido.

La recuperación en modo bipolar se muestra en la **Fig. 3.12** con 14.29% de ruido, en donde todos encuentran la solución esperada. Se acarrearon las recuperaciones con la corrección de HOPFIELD para los algoritmos deterministas, en vista de que las recuperaciones son más precisas (Fig. 3.5). Se observa que todas las recuperaciones son diferentes, excepto HOLO y KOHONEN, que además resultan ser las más rápidas y las únicas que encuentran la solución en los dos modos de recuperación.

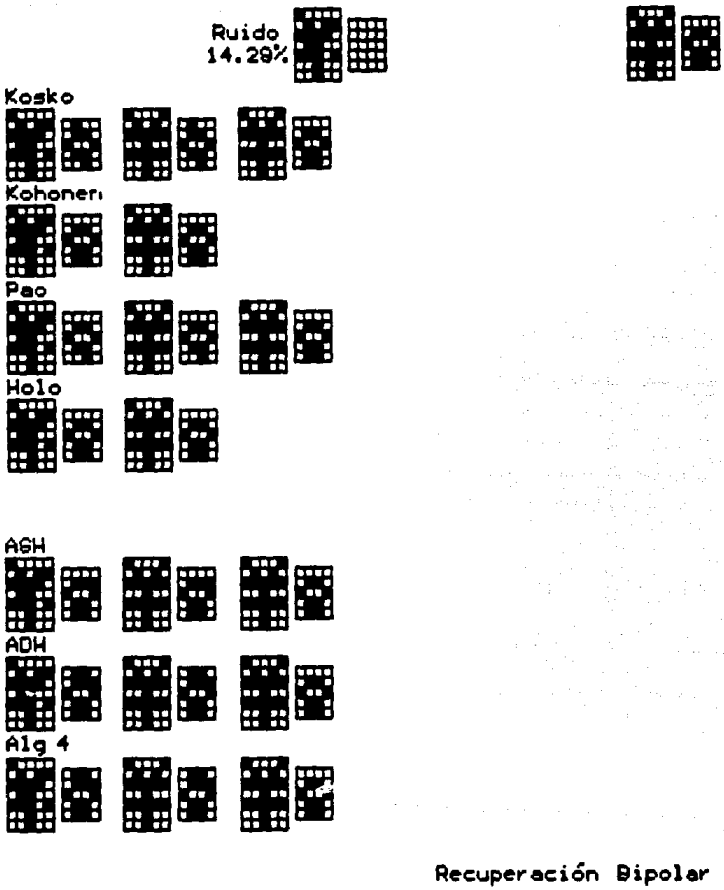


Fig. 3.12 Evolución de la red para todos los algoritmos con recuperación en modo bipolar y 14.29% de ruido.

En la **Fig. 3.13** se muestra la evolución de la red con recuperación en modo binario para 30.00% de ruido. Se acarrearon las recuperaciones sin la corrección de HOPFIELD, en vista de que sin la corrección, las recuperaciones son más precisas (**Fig.** 3.3). KOHONEN, ASH, ADH y Alg 4 obtienen soluciones espurias con errores de 47.27%, 45.45%, 47.27% y 36.36%, respectivamente, en la que ASH obtiene una solución ortogonal espuria. En cambio, KOSKO, PAO y HOLO encuentran la solución esperada en 3,3 y 2 iteraciones, respectivamente.

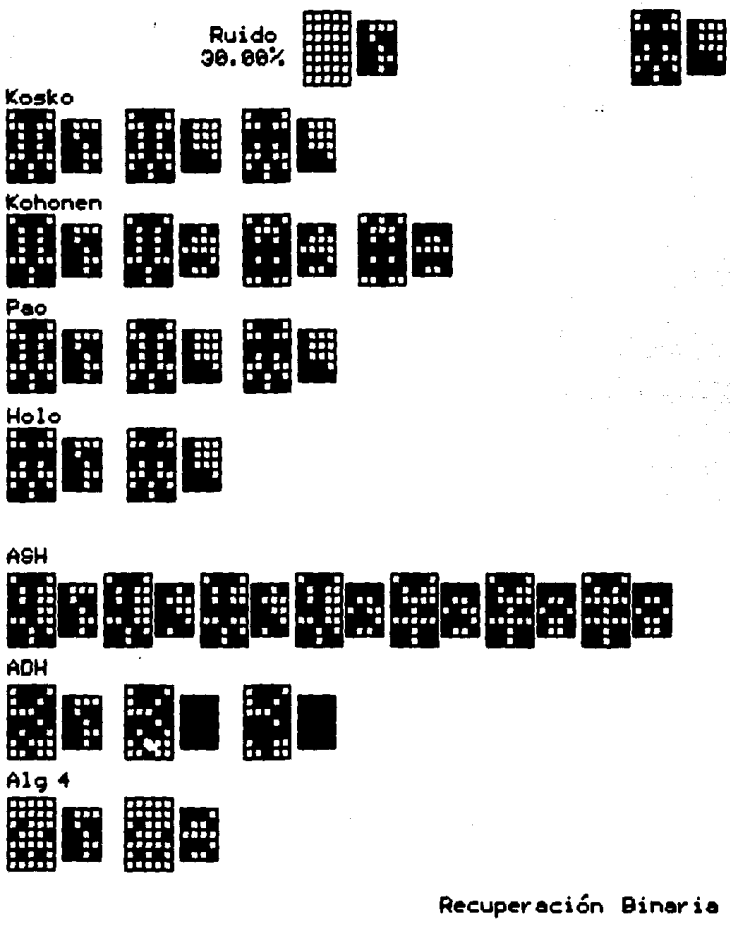


Fig. 3.13 Evolución de la red para todos los algoritmos con recuperación en modo binario y 30.00% de ruido.

La recuperación en modo bipolar se muestra en la Fig. 3.14 con 30.00% de ruido, en donde, excepto KOHONEN y ADH, todos encuentran la solución esperada. Se acarrearon las recuperaciones sin la corrección de HOPFIELD para los algoritmos deterministas, en vista de que las recuperaciones son más precisas (Fig. 3.3). Se observa que todas las recuperaciones en que se encuentra la solución esperada, para los algoritmos adaptivos, son diferentes; caso que no ocurre para las recuperaciones de los algoritmos deterministas en donde son iguales. Así KOSKO, PAO y HOLO obtienen las soluciones en la segunda iteración y ASH y Alg 4 en 5 y 3 iteraciones, respectivamente.

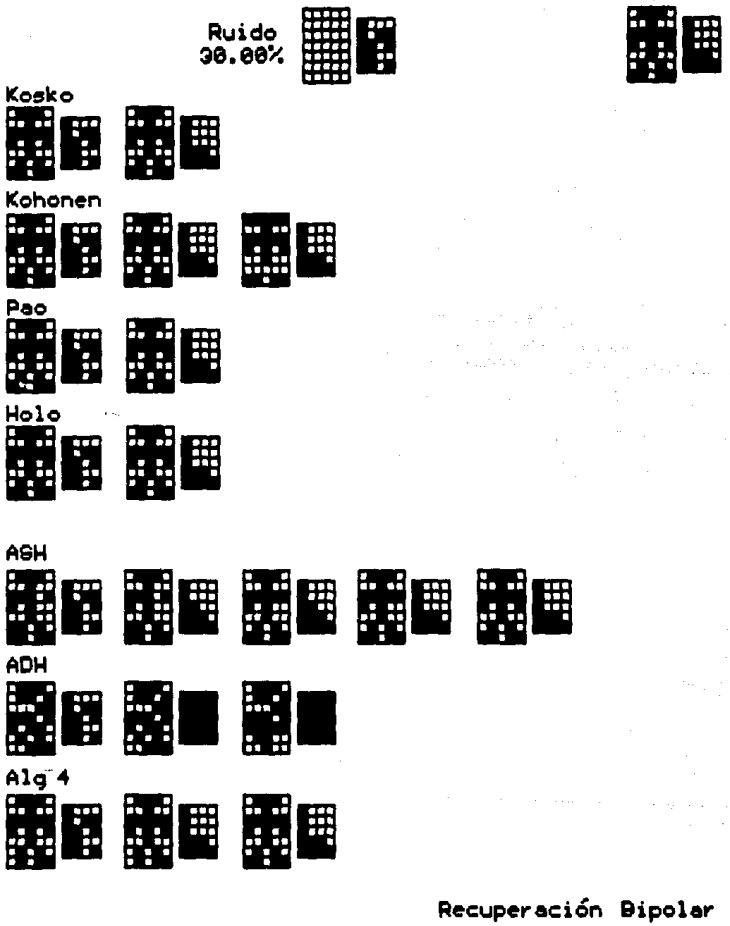


Fig. 3.14 Evolución de la red para todos los algoritmos con recuperación en modo binario y 30.00% de ruido.

En la Fig. 3.15 se muestra la evolución de la red con recuperación en modo binario para 35.00% de ruido. Se acarrearon las recuperaciones sin la corrección de HOPFIELD, en vista de que sin la corrección, las recuperaciones son más precisas (Fig. 3.4). KOSKO, KOHONEN, PAO, ADH y Alg 4 obtienen soluciones espurias con errores de 27.27%, 32.73%, 27.27%, 38.18% y 36.36%, respectivamente. En cambio, HOLO y ASH encuentran la solución esperada en 2 y 3 iteraciones, respectivamente.

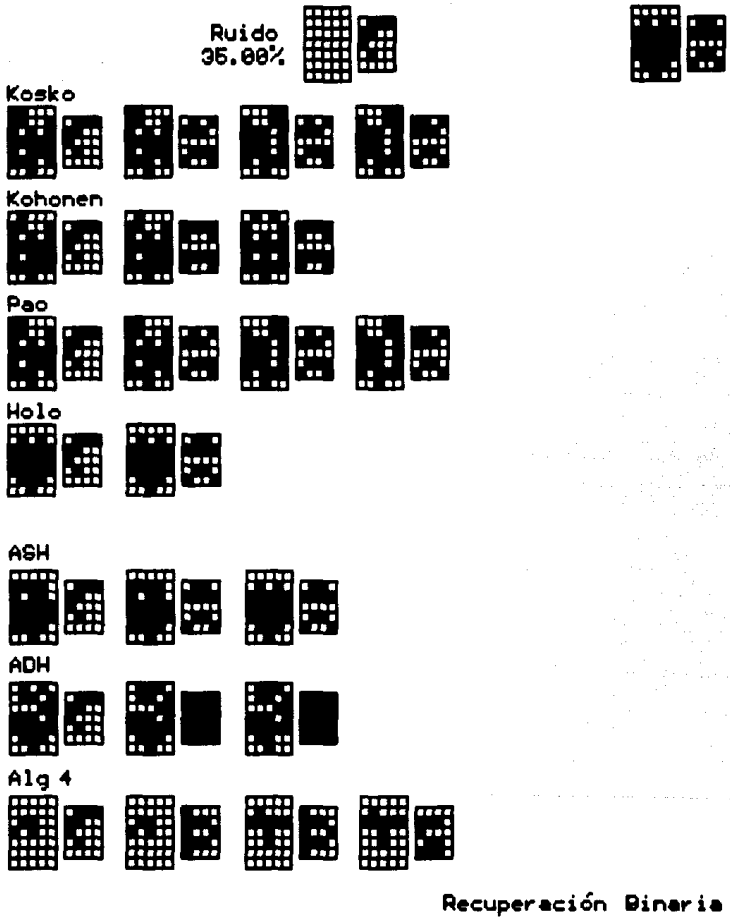


Fig. 3.15 Evolución de la red para todos los algoritmos con recuperación en modo binario y 35.00% de ruido.

La recuperación en modo bipolar se muestra en la Fig. 3.16 con 35.00% de ruido, en donde, excepto ADH y Alg 4, todos encuentran la solución esperada. ADH obtiene una solución no esperada. Se acarrearon las recuperaciones sin la corrección de HOPFIELD para los algoritmos deterministas, en vista de que las recuperaciones son más precisas (Fig. 3.4). Se observa que en todas las recuperaciones en que se encuentra la solución esperada, las recuperaciones de KOSKO y PAO son iguales en 3 iteraciones, KOHONEN diferente a las anteriores en 3 iteraciones y HOLO en sólo 2 iteraciones. ASH encuentra la solución esperada en 6 iteraciones.

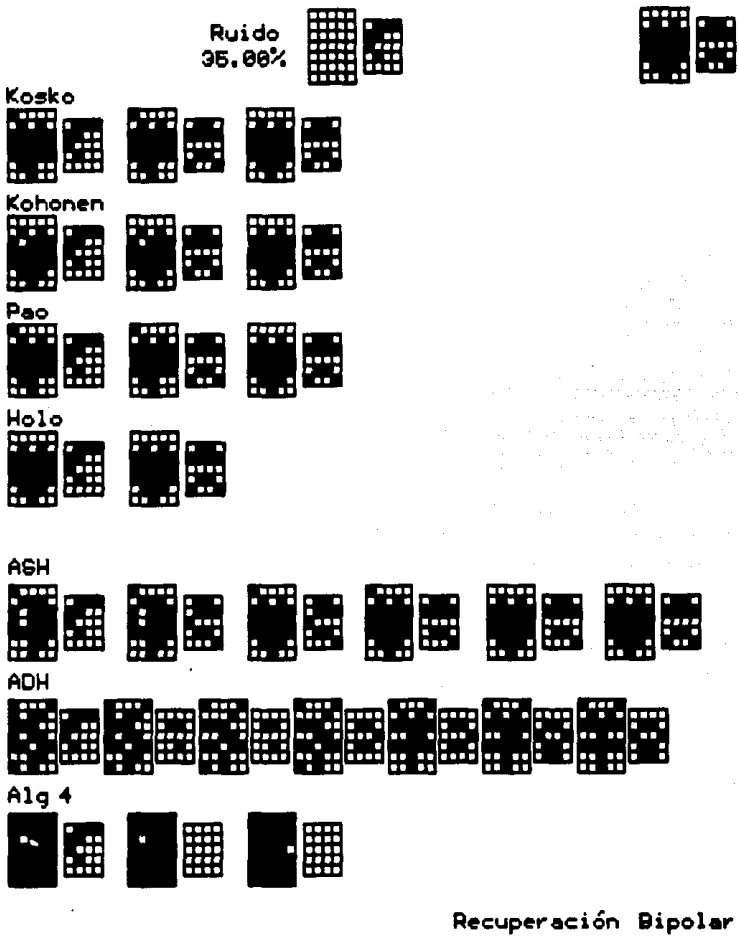


Fig. 3.16 Evolución de la red para todos los algoritmos con recuperación en modo binario y 35.00% de ruido.

3.5 RESUMEN DE LOS RESULTADOS Y ALGUNOS COMENTARIOS

Los resultados obtenidos sugieren que las recuperaciones en modo bipolar son más precisas que las recuperaciones en modo binario, tanto en los algoritmos deterministas como en los adaptivos. En este sentido, se puede decir que la "conducta" de la red neuronal mejora con la recuperación bipolar. Esto no tiene nada que ver con los algoritmos, ya que en las operaciones para la recuperación de patrones en modo binario sólo se toman algunos valores de la matriz; en cambio, en modo bipolar, se consideran todos los valores en la matriz de pesos.

Una observación más importante son las ventajas en la recuperación de patrones de un algoritmo con respecto a los demás. El algoritmo holográfico (HOLO) siempre encontró, para el ejemplo utilizado, la solución esperada, caso contrario del resto de los algoritmos deterministas y adaptivos que no siempre mostraron una solución satisfactoria, incluso una vez aplicada la corrección de HOPFIELD. En ciertos casos, esta corrección puede aumentar la precisión en la recuperación.

En todas las recuperaciones de patrones estudiadas, el algoritmo holográfico fue el único que no produjo soluciones espurias. Este algoritmo puede tolerar mayor porcentaje de error con respecto a los demás estudiados. Además, no presenta soluciones ortogonales.

Así pues, de los ejemplos y las simulaciones, se observa que el algoritmo determinista holográfico ofrece mayor precisión para la recuperación y el reconocimiento de patrones.

Algunos algoritmos "sirven mejor" que otros para ciertas asociaciones. Por ejemplo, se encuentran asociaciones que con el algoritmo de KOSKO no es posible reconocer ninguna de ellas, es decir, no es posible hacer recuperación alguna. En cambio, para el mismo caso es posible la recuperación de todas las asociaciones con el algoritmo de KOHONEN, por ejemplo.

Por otra parte, en los resultados se observa que para los algoritmos adaptivos de KOSKO: ASH y ADH, una vez aprendidas las

asociaciones, ya no se puede mejorar la precisión en la recuperación; es decir, que presentando un patrón ruidoso aún aparecen soluciones espurias.

3.6 OTROS TRABAJOS SIMILARES

De los algoritmos discutidos, sólo el de KOSKO ha sido estudiado con más detalle en el laboratorio [3]-[5],[7]. KOHONEN [13] ha discutido con detalle la memoria asociativa con matriz inversa generalizada, pero nada dice acerca de la bidireccionalidad, ni de la versión discreta, es decir, que los vectores tomen los valores discretos {-1,0,1}. Y ha demostrado que en el modelo continuo la recuperación será perfecta si los vectores asociados a los patrones son ortogonales. KOSKO afirma lo mismo para su modelo [14]-[17], al igual que PAO [32]. El modelo discreto será así un caso particular del modelo continuo. El algoritmo holográfico no presenta este problema, ya que se generaliza para el modelo continuo, es decir, se sigue el mismo procedimiento en el discreto y en el continuo. Sobre los algoritmos adaptivos de KOSKO, también se trabajó en la versión discreta.

Stiles y Denq [21] muestran que la recuperación depende del número de patrones asociados, del número de procesadores por capa y del patrón ruidoso (no muestran las evoluciones de la dinámica de la red en forma de activación de los procesadores). Aquí también se observó la misma situación, además que la recuperación también depende del algoritmo de aprendizaje empleado.

3.7 OTROS ALGORITMOS

Existen otros algoritmos de aprendizaje que se valen de diferentes teorías matemáticas. Wang, Cruz y Mulligan [23] ofrecen una corrección al modelo de KOHKO, que es muy similar al de HOPFIELD. Por otra parte, Murakami y Aibara [20] proponen una mejoría al modelo de KOHONEN, que no se trató aquí. Sin embargo, no existe hasta el momento un algoritmo satisfactorio que llegue

algoritmo de Retropropagación y otros [22],[32] garantizan la convergencia hacia la solución, pero esta puede llevar mucho tiempo. El algoritmo Alg 4 está construído siguiendo casi las mismas ideas del método de retropropagación, sólo que en sentido bidireccional, que con este esquema también garantiza la convergencia hacia la solución. Como ejemplo sólo basta ver el comportamiento no-lineal de este algoritmo: una asociación la aprendió (con las mismas condiciones iniciales, es decir, la matriz de pesos) en 7 segundos, dos asociaciones en 15 segundos, tres en 24 segundos, cuatro en 93 segundos y cinco asociaciones en aproximadamente 3 días no terminó su proceso de aprendizaje. La simulación se hizo en una computadora GAMA BABY con procesador 80286 a 10 MHz. Esto es sólo una pequeña muestra de lo difícil que es encontrar un algoritmo satisfactorio debido al tipo de sistema tratado: un sistema no-lineal dinámico multivariable con muchos elementos.

CAPITULO 4
ANALISIS DE LOS RESULTADOS

4.1 LA CONDUCTA DINAMICA DE UNA RED NEURONAL ARTIFICIAL

El estudio de algoritmos de aprendizaje deja un cierto vacío cuando se trata de captar la conducta global de la red neuronal. Es cierto que en las redes neuronales artificiales, una vez que ésta ha aprendido algo, se aplica y ya. Pero si se trata de extrapolar conocimiento hacia la comprensión de la dinámica de una red neuronal biológica, entonces, se quisiera poder conocer más sobre el funcionamiento de la red neuronal artificial. Con esto en mente, se hicieron estudios sobre las variaciones del error a cada cambio en los pesos para los algoritmos adaptivos; la graficación de la variación global de los pesos y, la correlación cruzada de la actividad de algunas de las neuronas artificiales. Después de estos estudios, desafortunadamente, no es mucho lo que se puede agregar sobre el conocimiento de la dinámica distribuída de la red. Sin embargo, la visualización y comprensión de los mecanismos subyacentes de lo que funciona en paralelo y distribuído -como el sistema nervioso- es un problema contemporáneo no resuelto, de ahí que sean válidos estos intentos.

Así pues, la pregunta es: ¿Qué sucede dentro de la red? Unas maneras de verlo, ya que se tienen muy pocas pistas de cuál puede ser la más reveladora, es hacer gráficas del error, de los pesos y, de correlaciones cruzadas.

4.2 GRAFICAS DE ERROR

La manera usual de estudiar la evolución del proceso de aprendizaje es graficar la variación del error a cada cambio en los pesos. Las siguientes figuras muestran el valor del error contra el número de iteraciones (número de cambios en los pesos). Es decir, con la matriz inicial se tiene un error que, de acuerdo al algoritmo utilizado, se va minimizando; además se indica el tiempo durante este proceso en horas, minutos y segundos.

En la **Figs. 4.1** y **4.2** se muestran las gráficas de error para los algoritmos adaptivos de KOSKO: ASH y ADH, respectivamente, para la red de la Fig. 2.2 con las cinco asociaciones de la Fig. 3.1. ASH hace 38 cambios en los pesos, partiendo de un error del 31.01% hasta llegar al 1.10%, en 6 minutos y 52 segundos, que lo hace ser más rápido que ADH con 49 cambios en los pesos en 9 minutos con 45 segundos que concluyó con error del 0.68%. La simulación se hizo en una computadora GAMA BABY con procesador 80286 a 10 MHz. Estos valores de error se obtienen empleando la fórmula (2-40), misma utilizada para el algoritmo de Retropropagación [32] (Cap. 2).

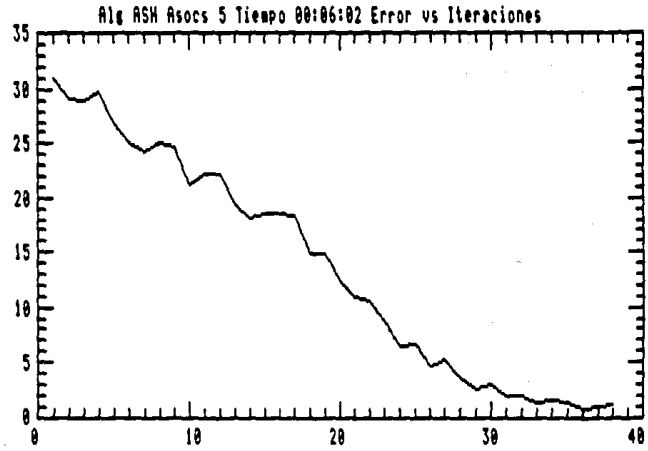


Fig. 4.1 Gráfica de variación del error para la red con 5 asociaciones con el algoritmo de aprendizaje ASH.

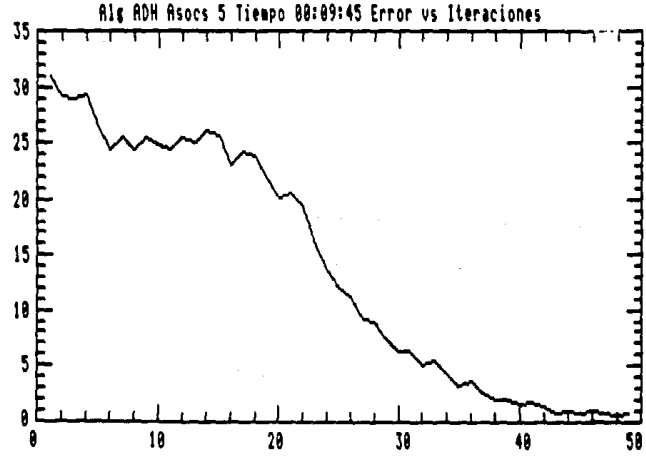


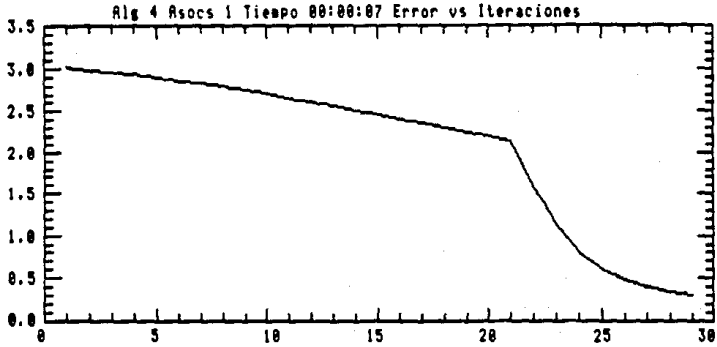
Fig. 4.2 Gráfica de variación del error para la red con 5 asociaciones con el algoritmo de aprendizaje ADH.

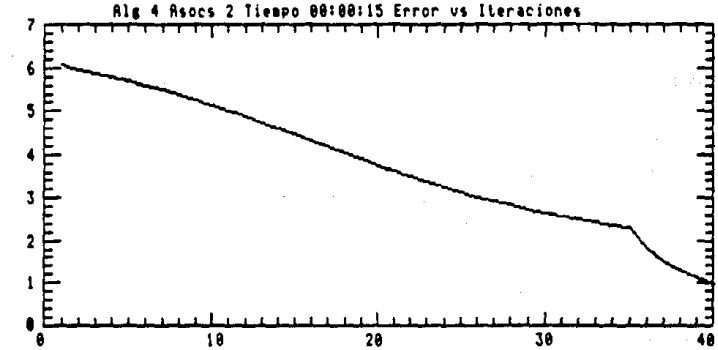
En la **Fig. 4.3** (a) y (b) se muestran las gráficas de error para diferentes número de asociaciones utilizando el algoritmo **Alg 4**. En el título de cada gráfica se indica el número de asociaciones en la red, las de la Fig. 3.1. Así por ejemplo, **Asocs** 3 indica que la red aprende 3 asociaciones, las asociaciones 1,2 y 3 de la Fig. 3.1.

En la primera gráfica de la Fig. 4.3(a) (Asocs 1) la red aprendió una asociación haciendo **29** iteraciones en 7 segundos, con error inicial de 3.01% y error final de 0.30%. Se observa aquí la pendiente pronunciada a partir de la iteración 21. Los valores de error se obtienen empleando la fórmula (2-46), misma propuesta por Hinton [32].

En la segunda gráfica de la Fig. 4.3(a) la aprendió dos asociaciones (Asocs 2) en **40** iteraciones y 15 segundos, con error inicial y final de 6.07% y 0.97%, respectivamente.

En la tercera gráfica de la Fig. 4.3(a) la red aprendió tres asociaciones (Asocs 3) haciendo **61** iteraciones en 24 segundos, con error inicial de 9.57% y error final de 2.22%.





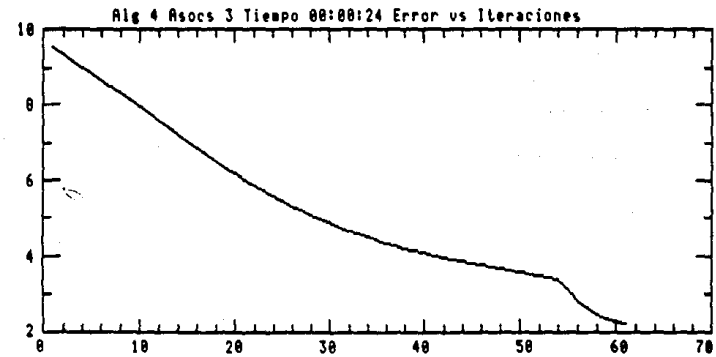
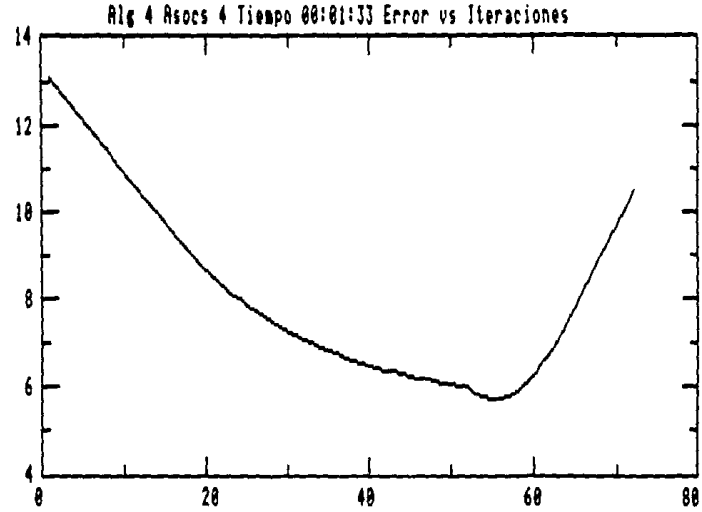


Fig. 4.3(a) Gráficas de variación del error para redes con 1,2,3,4 y 5 asociaciones utilizando el algoritmo Alg 4.

En la primera gráfica de la Fig. 4.3(b), la red aprendió cuatro asociaciones (Asocs 4) en 72 iteraciones durante 1 minuto y 33 segundos, con error inicial y final de 13.09% y 10.44%. Se observa que el error empezó a aumentar a partir de la iteración 56, en donde la red aún no ha concluido el proceso de aprendizaje con error del 5.72%. En cambio, en la iteración 72 con error del 10.44%, la red ha aprendido todas las asociaciones. Esto se debe a que en la fórmula de error (2-46) uno de los sumandos tenía error casi cero y el otro un valor cercano al error total; es decir, el error en una ruta, como se indica en el Cap. 2, es muy grande. A partir de aquí el algoritmo Alg 4 modificó los pesos haciendo que los errores se hicieran casi iguales, hasta terminar el proceso de aprendizaje en la iteración 72.

Hasta aquí se observa que a medida que van aumentando el número de asociaciones en la red, aumentan también los errores: inicial y final. La red con 5 asociaciones a aprender (Asocs 5) no concluyó el proceso de aprendizaje en las más de 10,000 iteraciones que toman más de 72 horas de corrida, con error inicial de 16.49%. Esta es una muestra clara de lo que se definió en la introducción, la MAB es un sistema dinámico no-lineal, por lo que no se puede esperar que para la red con cinco asociaciones se concluya el proceso de aprendizaje en "pocas iteraciones", extrapolando del tiempo para redes más pequeñas, por lo que su comportamiento es no-lineal.

Para el ejemplo considerado, los algoritmos adaptivos de KOSKO son más rápidos en su proceso de aprendizaje. ASH es el más preciso en la recuperación que los otros algoritmos adaptivos. El segundo más preciso es Alg 4 y por último ADH.



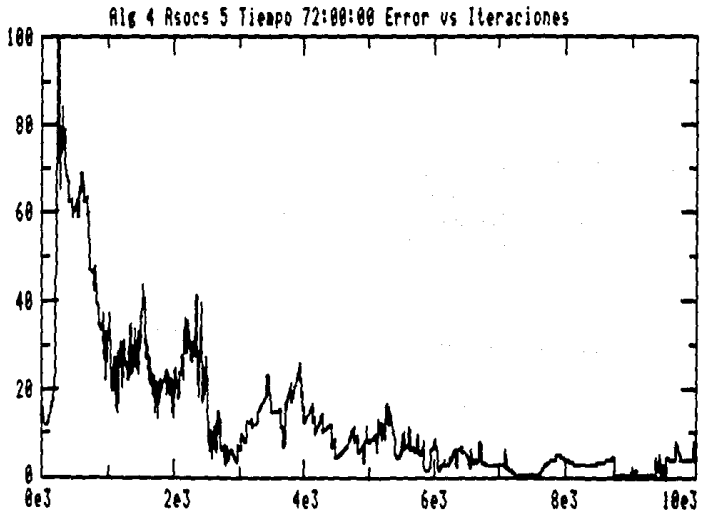


Fig. 4.3(b) GrÁficas de variación del error para redes con 1,2,3,4 y 5 asociaciones utilizando el algoritmo Alg 4.

4.3 GRAFICAS DE PESOS

Una forma de ver las conexiones en la red es graficando una superficie en 3 dimensiones utilizando los valores de la matriz de pesos como alturas desde un plano XY, que son los renglones y columnas. Esto se hace con el fin de visualizar el comportamiento global de los procesadores de la red.

4.3.1 ALGORITMOS DETERMINISTAS

En la **Fig. 4.4** (a) y (b) se muestran las matrices de pesos para los algoritmos deterministas para redes con cinco asociaciones, las empleadas en las recuperaciones anteriores, además se indican los valores máximo y mínimo de los pesos en la matriz. Recuérdese que las matrices de pesos para este tipo de algoritmos se obtienen en forma *única*, es decir, no existe matriz de pesos iniciales y finales como ocurre en los algoritmo adaptivos.

En la Fig. 4.4(a) con inspección visual, las gráficas para KOSKO y PAO parecen ser iguales ya que, con base a los algoritmos, son sólo proporcionales. KOHONEN obtiene una superficie más suave que las anteriores. Los valores de pesos en KOSKO son los más grandes que los de KOHONEN y PAO; este último tiene los pesos más pequeños.

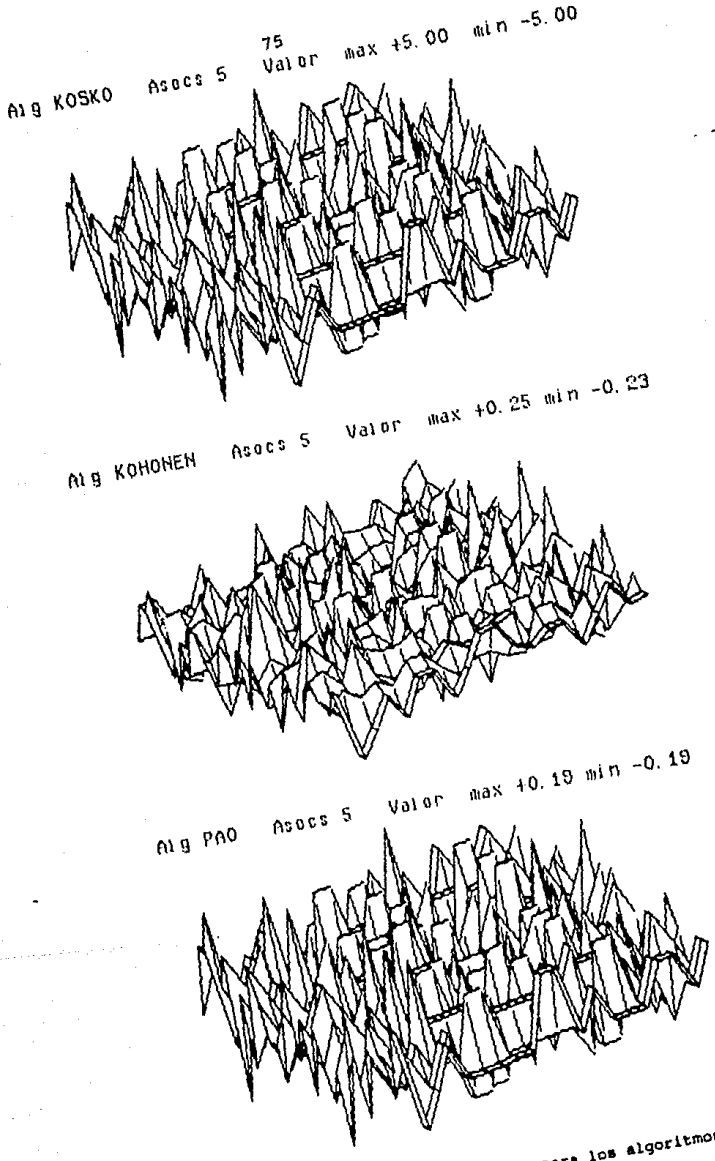
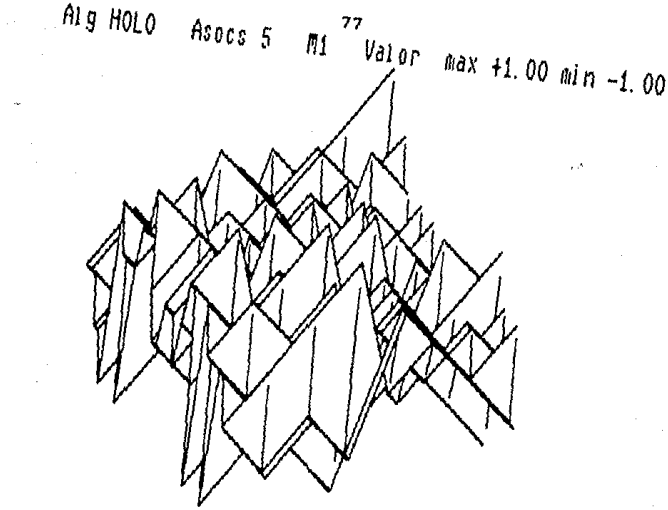


Fig. 4.4(a) Matrices de pesos para los algoritmos deterministas: KOSKO, KOHONEN y PAO.

En la Fig. 4.4(b) las matrices de HOLO resultan estar entre +1.00 y -1.00. Recuérdese que son dos matrices diferentes que produce HOLO, como se indica en el Cap. 2.

Los pesos de las conexiones entre los procesadores que se encuentran como a la mitad de cada capa, parecen tener cierta simetría entre ellos, para las matrices de KOSKO y PAO, como se muestra en la figura anterior. Lo mismo se puede decir de la matriz de KOHONEN para todos los pesos de las conexiones, donde la superficie es más suave. Esto no ocurre para los procesadores en los extremos de las capas, y en general para toda la superficie de KOSKO o PAO. En la segunda superficie de HOLO se observa más claramente lo anterior.



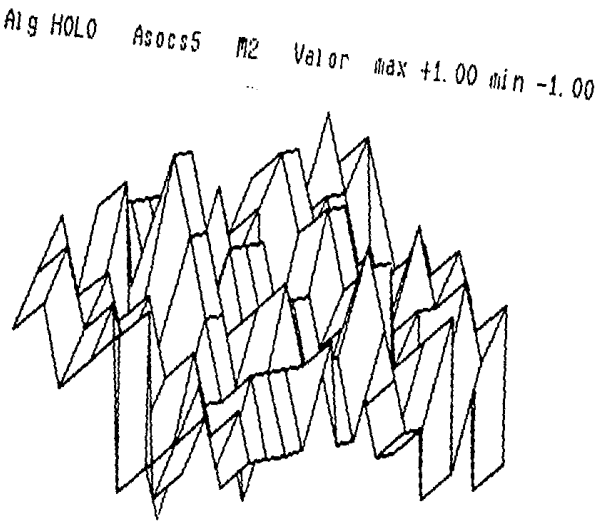


Fig. 4.4(b) Matrices de pesos para el algoritmo determinista holográfico: HOLO.

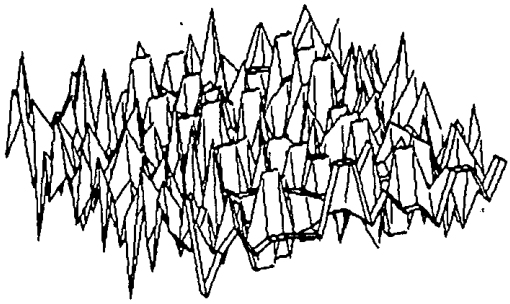
4.3.2 CORRECCION DE HOPFIELD

En la **Fig. 4.5** se muestran las matrices de pesos para los algoritmos deterministas con corrección de HOPFIELD.

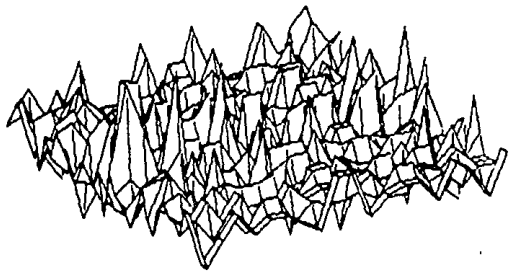
Nuevamente KOSKO tiene los valores más grandes. Las superficies de KOSKO y PAO que antes de la corrección eran proporcionales ahora no lo son, aunque siguen teniendo superficies muy similares, comparándolas con las superficies de la Fig. 4.4(a) sin corrección.

KOHONEN obtiene una superficie menos suave que sin la corrección. Para todas las superficies la corrección de HOPFIELD hace aparecer más "picos".

Algs KOSKO y HOPFIELD Asocs 5 Valor max +4.05 min -4.05



Algs KOHONEN y HOPFIELD Asocs 5 Valor max +0.17 min -0.14



Algs PAO y HOPFIELD Asocs 5 Valor max +0.12 min -0.12

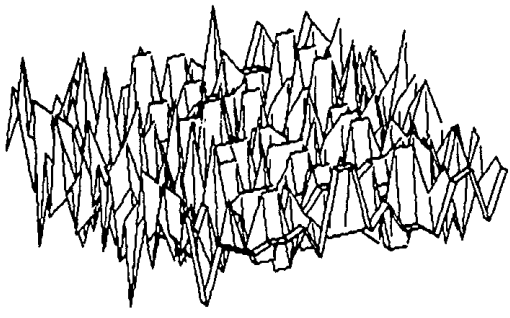


Fig. 4.5 Matrices de pesos finales para los algoritmos deterministas: KOSKO, KOHONEN y PAO, después de aplicar la corrección de HOPFIELD.

4.3.3 ALGORITMOS ADAPTIVOS

La **Fig. 4.6** muestra la matriz inicial de pesos aleatorios utilizada en el proceso de aprendizaje para los algoritmos adaptivos estudiados: Aprendizaje por Señal de Hebb (*ASH*), el Aprendizaje Diferencial de Hebb (*ADH*) y el algoritmo *Alg 4*. Se escogió así de acuerdo al generador de números aleatorios de la computadora entre los valores de ±0.5, como lo proponen varios autores [15],[17],[22],[32].

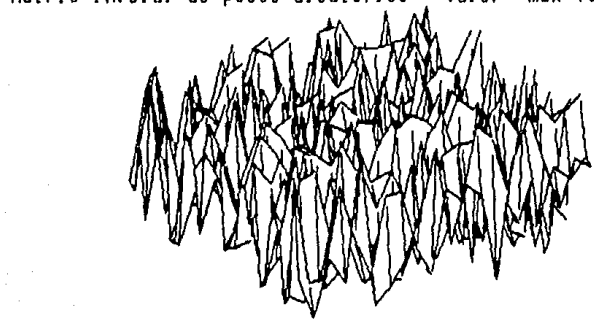


Fig. 4.6 Matriz inicial de pesos aleatorios para los algoritmos adaptivos.

En la **Fig. 4.7** se muestran las matrices de pesos finales, es decir, la que se generó después del proceso de aprendizaje, para los algoritmos adaptivos de KOSKO: ASH y ADH, para redes con las 5 asociaciones de la Fig. 3.1. Ambas superficies son similares, sin embargo la matriz ADH tiene valores mayores que la matriz ASH. También se observa que las matrices finales son diferentes a la matriz inicial de la Fig. 4.6.

Los valores de los pesos de las conexiones se muestran en las **Tablas 4.1 y 4.2**, donde la matriz es de 35x20, y además se muestra la superficie de la matriz. La Tabla 4.1 corresponde a la matriz inicial de pesos aleatorios y la Tabla 4.2 corresponde a la matriz final de pesos para la red con 5 asociaciones utilizando el algoritmo ASH.

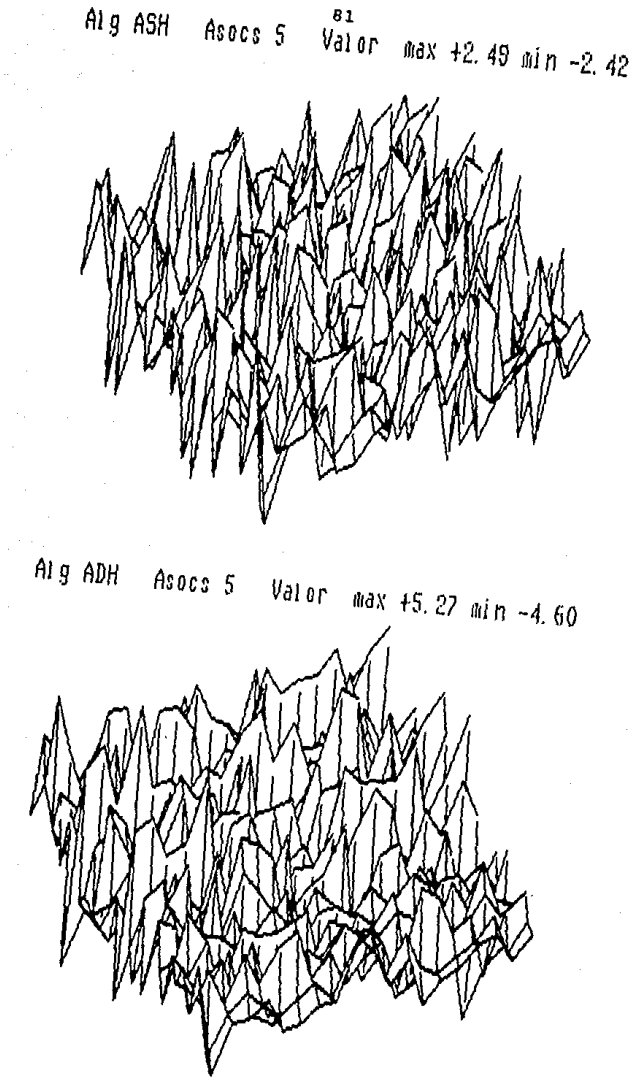


Fig. 4.7 Matrices de pesos finales para los algoritmos adaptivos: ASH y ADH.

-0.29	0.24	0.26	-0.31	-0.48	-0.06	-0.13	-0.31	0.10	-0.47	-0.31	-0.26	0.01	0.45	0.04	-0.36	-0.13	0.22	0.22	0.25
-0.32	0.35	0.07	-0.19	0.19	-0.19	-0.32	-0.18	0.36	0.36	0.41	-0.35	-0.34	-0.06	-0.34	0.20	-0.45	0.15	-0.10	0.42
-0.08	0.43	-0.25	-0.00	-0.38	0.06	0.47	-0.49	-0.27	-0.33	-0.29	0.25	0.22	0.48	0.30	-0.14	0.47	-0.09	-0.42	0.22
-0.46	0.08	-0.33	-0.19	0.04	0.42	0.13	0.45	0.49	0.12	-0.20	0.27	-0.49	-0.32	0.09	0.03	-0.21	0.14	0.31	0.19
0.37	-0.42	-0.07	0.38	-0.15	-0.04	0.12	0.46	0.40	-0.18	-0.37	0.49	0.49	-0.17	0.14	0.16	-0.39	-0.30	0.36	-0.12
0.24	-0.27	0.32	0.20	-0.18	0.26	0.01	-0.50	0.15	-0.04	-0.38	0.26	-0.21	0.38	-0.05	-0.29	0.25	0.50	0.20	0.00
-0.00	-0.15	-0.36	0.22	-0.36	0.32	0.32	-0.16	-0.02	0.19	-0.09	0.48	-0.16	0.39	0.37	0.37	-0.50	0.18	0.38	-0.12
0.24	-0.22	0.26	0.20	0.42	0.47	-0.48	0.11	0.04	-0.48	-0.33	-0.05	-0.20	0.32	-0.00	-0.48	0.44	-0.26	-0.35	-0.48
-0.48	-0.26	-0.33	0.13	0.00	-0.38	0.31	0.10	-0.15	-0.24	0.36	-0.09	-0.33	0.02	0.20	-0.14	-0.27	0.23	0.09	0.20
-0.49	0.49	-0.36	0.00	-0.17	0.45	-0.15	0.22	-0.12	-0.20	0.43	0.14	0.16	-0.21	-0.17	0.25	-0.29	-0.02	-0.11	-0.10
-0.48	-0.19	0.33	0.12	0.01	-0.11	-0.37	0.23	0.22	0.27	-0.27	-0.00	-0.05	0.10	0.46	-0.27	-0.35	-0.02	0.27	0.45
0.23	0.02	-0.10	0.14	-0.16	-0.25	-0.20	0.43	-0.43	0.46	0.30	0.45	-0.29	-0.41	0.30	0.40	0.26	-0.09	-0.29	-0.44
0.21	0.35	0.24	0.46	-0.15	0.26	0.42	-0.11	0.38	-0.39	-0.01	0.42	-0.23	0.07	0.38	0.00	-0.48	-0.14	0.18	0.49
0.05	0.16	0.48	-0.22	-0.49	0.15	-0.48	0.12	-0.09	-0.46	0.32	-0.12	-0.29	0.14	0.18	0.38	-0.26	0.14	-0.02	-0.05
-0.33	-0.03	-0.40	-0.31	0.03	0.01	-0.03	-0.00	-0.03	-0.16	-0.20	-0.38	0.27	0.17	-0.30	0.40	-0.05	-0.21	-0.16	0.20
0.19	-0.01	0.47	-0.02	0.40	-0.16	0.09	-0.21	0.16	0.12	-0.49	0.06	0.20	-0.12	0.47	-0.40	-0.17	-0.32	-0.03	-0.17
0.38	0.13	-0.20	-0.35	0.18	-0.41	0.33	-0.25	-0.00	-0.00	0.03	-0.32	0.17	-0.01	0.32	0.42	0.37	0.02	-0.38	0.23
0.13	-0.19	-0.09	-0.00	0.38	0.35	-0.09	0.33	0.41	0.44	-0.18	0.46	-0.03	-0.39	0.32	-0.20	0.04	-0.46	-0.09	-0.34
0.26	-0.12	0.11	0.12	-0.32	0.03	-0.21	-0.05	-0.07	-0.20	-0.10	-0.31	0.26	0.12	0.20	-0.17	0.12	-0.32	0.43	0.06
-0.33	-0.17	-0.26	0.00	-0.38	0.17	-0.27	0.47	-0.16	-0.19	0.41	-0.35	-0.31	-0.20	-0.25	0.11	-0.19	-0.40	-0.39	0.29
-0.38	0.02	-0.48	-0.19	0.22	0.02	0.48	-0.07	-0.13	0.14	-0.44	-0.44	-0.15	0.42	-0.35	0.47	-0.36	0.14	0.32	-0.41
-0.20	-0.26	0.45	0.34	0.16	0.09	0.35	0.47	0.37	-0.09	-0.17	0.44	-0.20	-0.07	0.41	-0.38	-0.49	-0.16	-0.19	0.44
0.13	0.38	-0.15	-0.26	-0.28	-0.22	0.37	0.31	-0.37	0.41	-0.05	0.01	0.21	0.41	-0.35	-0.43	0.26	-0.22	0.17	-0.33
0.13	0.42	0.31	-0.47	-0.03	0.44	0.17	-0.16	-0.21	-0.20	0.34	0.24	0.12	-0.06	-0.49	-0.47	-0.41	0.45	-0.32	0.46
-0.00	0.04	0.27	0.09	-0.24	0.46	-0.35	0.41	0.39	-0.41	-0.12	-0.20	0.10	-0.11	0.11	-0.46	0.10	0.27	-0.31	-0.25
0.31	0.17	-0.46	0.37	0.20	0.20	0.34	-0.01	0.25	0.38	-0.38	0.03	0.26	0.49	0.20	-0.44	0.09	-0.18	0.37	-0.14
-0.42	-0.38	-0.20	0.00	-0.20	-0.10	0.46	-0.06	-0.40	0.36	0.43	0.37	0.24	0.47	0.38	-0.00	-0.14	-0.26	0.15	0.38
-0.39	-0.04	0.42	0.34	-0.49	-0.40	0.37	0.09	0.13	-0.11	-0.46	-0.25	0.09	-0.01	0.44	-0.10	-0.42	0.26	0.25	0.10
-0.03	0.26	-0.04	-0.31	0.19	0.15	0.19	-0.04	0.16	-0.05	-0.36	-0.42	-0.22	-0.01	-0.39	0.07	-0.46	-0.29	0.25	0.32
0.10	-0.06	0.41	-0.45	0.00	0.15	-0.24	-0.27	-0.32	-0.17	0.34	0.49	0.16	-0.30	0.00	0.42	0.25	-0.46	-0.20	-0.34
-0.04	0.42	-0.20	-0.38	0.47	0.27	-0.31	-0.45	-0.46	-0.19	-0.39	0.17	0.30	0.29	0.45	-0.30	-0.37	0.43	0.30	0.42
-0.42	0.30	0.34	-0.07	-0.00	-0.30	-0.19	-0.02	-0.17	0.39	-0.04	0.10	-0.07	-0.29	0.27	-0.21	0.27	-0.46	-0.00	0.17
-0.36	-0.41	-0.33	0.09	0.02	-0.30	-0.07	-0.13	0.20	-0.30	0.29	0.34	-0.39	0.46	-0.05	-0.22	0.05	-0.37	0.40	0.49
0.37	-0.20	-0.20	0.26	-0.09	-0.32	0.15	0.15	-0.21	-0.39	0.11	0.01	0.46	0.38	0.03	0.45	-0.22	0.11	0.01	0.13
-0.22	-0.47	-0.14	-0.30	0.46	-0.40	-0.38	-0.24	0.04	0.40	-0.21	0.13	0.05	-0.40	0.21	0.49	0.17	-0.13	0.16	0.00

Matriz inicial de pesos aleatorios Valor max +0.50 min -0.50

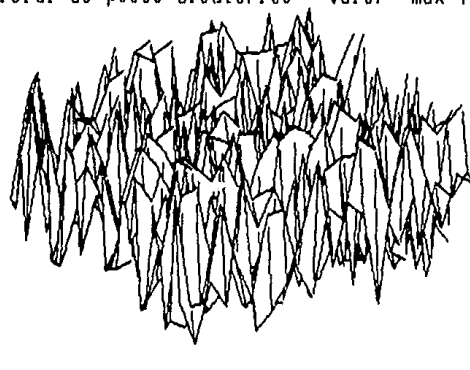


Tabla 4.1 Valores de la matriz inicial de pesos aleatorios

-0.61	0.67	-0.27	0.35	-1.01	-0.27	-0.34	-0.84	1.17	-0.79	-0.63	0.81	0.22	0.24	-0.17	0.81	-1.20	0.33	1.29	-0.82
1.17	0.67	-0.57	0.34	1.47	-2.22	-2.35	-0.82	1.33	-0.07	-0.82	-1.10	1.69	-0.17	-0.45	1.35	0.38	0.34	0.99	1.37
1.87	-0.74	-0.46	0.11	-0.59	-0.67	-0.84	-0.78	-0.59	0.74	0.78	-0.87	0.75	-0.85	-0.23	-0.57	0.79	0.55	-0.74	0.54
1.57	-0.15	-1.50	0.88	0.81	-1.07	-1.36	-0.72	1.15	0.25	-0.17	-1.01	1.08	0.11	0.57	0.54	1.07	-0.10	0.95	1.47
0.58	-0.55	-1.14	1.55	-1.22	0.28	0.44	-0.41	0.97	-1.03	-0.16	1.02	0.17	0.15	0.46	0.86	-0.92	-0.75	0.09	-0.45
1.31	0.48	0.11	2.35	-0.59	-0.27	-0.52	-0.51	-0.17	1.01	0.77	-0.86	0.52	-0.15	-0.50	1.20	0.57	-0.76	-0.04	0.49
-2.03	0.86	0.83	-0.85	-1.11	1.01	1.01	1.01	-0.66	0.89	-0.20	1.76	-1.65	-0.84	-0.86	-0.16	-1.78	0.50	-0.24	-1.49
0.77	-0.86	0.58	-0.73	0.74	-0.69	-1.55	0.43	0.27	0.05	0.20	0.16	0.87	-0.75	-1.07	-0.57	0.75	0.91	-0.14	-0.69
-1.97	-0.58	0.51	-0.40	-1.20	1.75	2.34	0.74	-1.32	0.19	0.79	0.66	-2.36	0.13	0.31	-1.21	-1.02	0.82	-1.98	-0.55
0.58	1.24	-0.57	2.11	-0.38	-0.86	-0.48	0.81	-0.44	0.79	1.58	-0.18	0.69	-0.74	-0.78	1.74	0.83	-1.38	-0.43	0.22
-0.85	-0.51	-0.75	-0.41	0.65	-0.88	-0.26	-1.05	0.97	-1.22	-1.74	-1.25	-0.16	2.13	2.49	-1.34	0.82	-0.25	1.02	1.62
-0.38	-1.24	-0.58	-1.35	-0.48	0.82	0.79	0.11	-0.64	-0.87	-0.23	0.24	-1.56	0.66	1.37	-1.55	0.47	0.66	-0.58	-0.25
0.18	0.56	-0.51	-0.61	1.82	-0.17	-0.81	-0.86	1.58	-2.42	-2.84	-0.22	0.28	1.56	1.87	-0.45	0.16	0.18	1.46	1.15
-1.82	-0.59	0.61	-2.25	-0.28	0.68	0.85	0.33	0.23	-1.53	-0.75	0.28	-0.82	0.67	0.71	-1.19	-0.58	1.42	0.38	-0.37
0.18	-0.35	-1.66	-0.84	0.67	0.12	0.96	-1.28	0.72	-1.65	-1.77	-1.55	0.16	2.29	1.73	-0.59	1.12	-0.42	0.59	1.45
1.47	0.52	0.84	0.38	1.89	-0.91	-0.66	-0.44	0.85	-0.52	-1.15	-1.97	0.95	1.85	1.66	-0.61	1.86	-1.39	-0.16	1.86
-0.26	0.66	1.21	-0.83	-0.25	0.76	1.50	1.24	-2.11	1.28	1.31	-0.43	-1.88	-0.76	-0.43	0.21	0.48	-1.05	-2.41	0.34
0.56	-0.31	-1.37	-0.55	0.94	0.46	0.82	-0.95	1.16	-1.05	-1.47	-0.71	-0.16	1.66	2.35	-1.27	1.21	-0.67	0.66	0.83
-0.91	0.95	2.16	-0.89	-0.21	0.67	0.43	1.98	-1.56	0.47	0.65	0.12	-0.58	-1.16	-1.88	0.35	-0.31	-0.85	-1.84	-0.37
0.42	0.98	-0.15	-0.15	1.73	-1.11	-1.55	0.58	0.27	-1.34	-0.76	-1.86	0.97	0.44	0.41	0.43	1.36	-1.01	0.84	1.78
-0.85	0.66	-0.88	0.24	-0.18	1.89	1.55	-0.39	-0.34	-0.39	-0.97	-0.65	-1.22	1.49	0.72	0.56	-0.15	-1.83	0.31	-0.20
-1.77	-0.58	1.89	-0.19	-1.12	2.12	2.56	1.31	-0.88	0.36	0.26	1.19	-2.23	0.84	0.52	-1.45	-1.24	-0.37	-1.56	-0.31
0.82	0.59	-0.90	-1.33	0.97	-0.65	-0.86	-0.44	0.91	-1.62	-2.86	-0.63	0.64	1.96	1.14	-0.96	0.98	0.18	1.45	0.31
-1.98	0.63	1.48	-1.56	-0.78	1.93	1.66	1.01	-0.85	-0.31	0.23	1.52	-1.57	-0.69	-0.92	-1.88	-1.69	0.77	-0.94	-0.82
-0.61	0.68	-0.85	0.52	-0.56	1.53	0.72	0.89	0.18	-0.94	-0.65	-0.69	-0.97	0.96	1.18	-0.57	0.31	-0.98	-0.52	-0.84
1.38	0.92	-0.67	2.48	0.87	-0.33	-0.17	-0.22	-0.87	1.45	0.77	-0.29	0.79	-0.84	-0.25	1.95	0.61	-1.46	0.85	0.18
-0.65	-0.19	0.87	-1.09	0.79	-0.58	0.16	1.61	-0.93	0.15	0.22	-0.16	0.56	0.15	0.84	-0.64	0.39	0.17	-0.38	0.91
-1.46	1.15	0.63	0.23	-0.28	0.15	0.98	0.38	0.45	-1.18	-1.55	0.87	-0.44	0.12	0.97	0.25	-0.74	-0.38	0.57	-0.22
0.29	-0.17	0.47	-0.95	0.72	0.56	0.48	0.49	-0.91	0.27	-0.84	-1.49	-0.63	0.20	-0.18	-1.18	0.61	-0.48	-0.02	1.39
1.17	0.69	0.20	1.58	-0.15	-0.38	-0.77	-0.48	-0.64	0.98	1.41	0.17	0.69	-0.83	-0.45	1.91	0.57	-1.74	-0.52	-0.82
0.49	-0.22	0.12	-0.81	0.79	-0.80	-1.38	-0.15	-0.25	0.34	0.16	0.38	1.57	-0.78	-0.62	-0.19	-0.58	1.60	0.51	0.21
0.11	-0.36	0.66	-0.58	0.24	-1.45	-1.26	0.58	0.84	0.92	0.49	0.59	1.00	-1.36	-0.88	-0.16	0.84	0.71	0.21	-0.84
-0.89	0.23	-0.65	0.52	-0.58	0.69	1.88	-0.45	0.87	-0.91	-0.24	0.13	-1.46	1.53	1.84	-0.33	0.26	-1.54	0.19	0.78
1.44	-1.45	-0.49	0.37	-0.38	-0.85	-0.58	-0.84	-0.53	0.88	1.18	-0.31	0.99	-0.15	-0.58	0.82	0.18	0.75	-0.51	0.45
0.85	-1.64	-0.35	-0.19	0.25	-1.91	-0.91	-0.45	-0.28	1.47	0.84	-0.19	0.58	-1.81	-0.32	0.84	0.49	0.51	-0.16	0.48

Alg ASH Asocs 5 Valor max +2.49 min -2.42

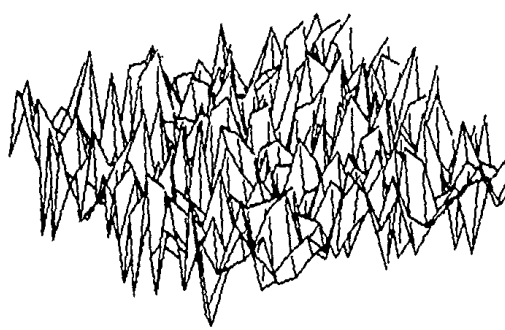


Tabla 4.2 Valores de la matriz final de pesos para la red con 5 asociaciones con el algoritmo ASH.

En la **Fig. 4.8** (a) y (b) aparecen las matrices finales para redes con 1,2,3,4 y 5 asociaciones utilizando el algoritmo Alg 4, descritas en la Fig. 4.3 con sus gráficas de variación de error, partiendo de la matriz inicial de la Fig. 4.6. Asocs indica el número de asociaciones que debe aprender la red. Se observa que a medida que aumenta el número de asociaciones que la red deberá aprender, la superficie aparentemente tiende a ser más suave, además de que aumentan los valores máximo y mínimo de la matriz.

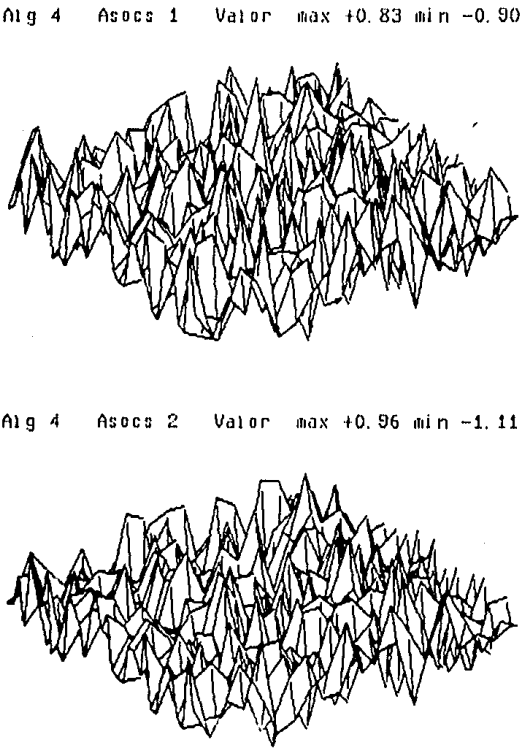
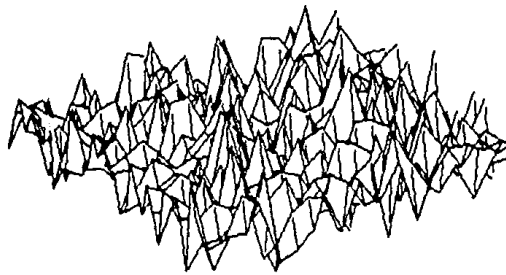
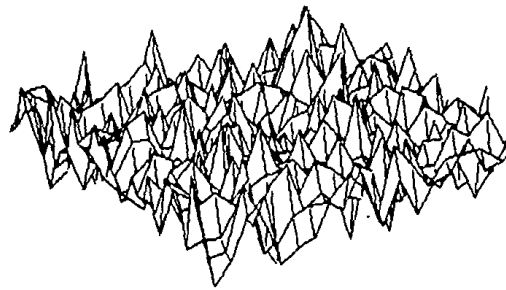


Fig. 4.8(a) Matrices de pesos finales de la red tipo MAB con 1 y 2 asociaciones para el algoritmo Alg 4.

Alg 4 Asocs 3 Valor max +1.16 min -1.35



Alg 4 Asocs 4 Valor max +1.28 min -1.93



Alg 4 Asocs 5 Iteracion 9800 Valor max +67.68 min -73.74

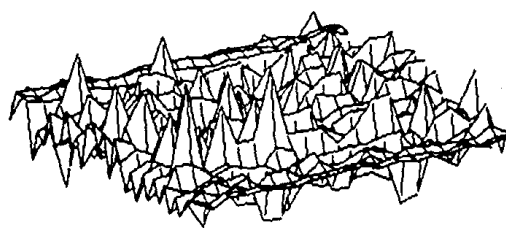


Fig. 4.8(b) Matrices de pesos finales de la red tipo MAB con 3,4 y 5 asociaciones para el algoritmo Alg 4.

4.3.4 CORRELACION CRUZADA Y PESOS

En un intento por detectar las correlaciones entre los procesadores de la red, pero ahora durante el proceso de aprendizaje, se graficó la evolución de las matrices, desde la matriz inicial de pesos de la Fig. 4.6 hasta la matriz final de pesos, para el caso de la red con 5 asociaciones con el algoritmo ADH. Esta evolución se muestra en la **Fig. 4.9** (a),(b) y (c).

Se acarreó la matriz inicial de pesos de la Fig. 4.6 para comparación. Al inicio el error es de 31.01%. Este valor de error y los siguientes se obtienen empleando la fórmula (2-40), misma utilizada para el algoritmo de Retropropagación [32] (Cap. 2).

En la segunda gráfica correspondiente a la iteración 3 se observa que los valores máximo y mínimo aumentaron desde la matriz inicial de +0.50 y -0.50 a +0.65 y -0.65, además ya se muestran cambios en la superficie. Aquí la red ya aprendió la asociación 1 de la Fig. 3.1. El error es de 28.90%.

En la iteración 8 la superficie se hace más suave y los valores máximo y mínimo de la matriz aumentan, además la red aprendió ya las asociaciones 1 y 2 de la Fig. 3.1. El error disminuye hasta 24.47%.

Para la iteración 17, se observa que algunos procesadores tienden a aumentar su peso en la conexión, ya que los valores extremos siguen aumentando. La red para esta iteración ya aprendió las tres primeras asociaciones de la Fig. 3.1. El error es ya del 24.21%.

Para la iteración 23, los pesos de las conexiones siguen aumentando y el error disminuyendo hasta el 16.07%.

Y por último, en la iteración 49, los valores de los pesos aumentaron aún más. La red en ésta última iteración termina su proceso de aprendizaje para las 5 asociaciones de la Fig. 3.1, con error de 0.68%.

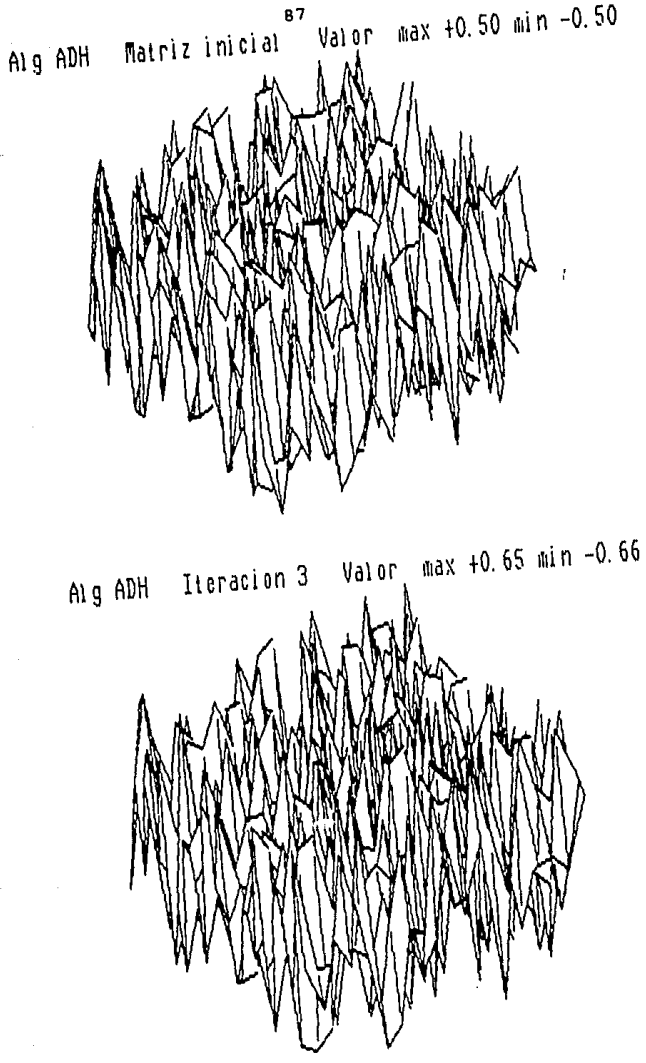


Fig. 4.9(a) Evolución durante el proceso de aprendizaje de las conexiones de la red con las 5 asociaciones de la Fig. 3.1 con el algoritmo adaptivo ADH. Nótese los cambios de escala.

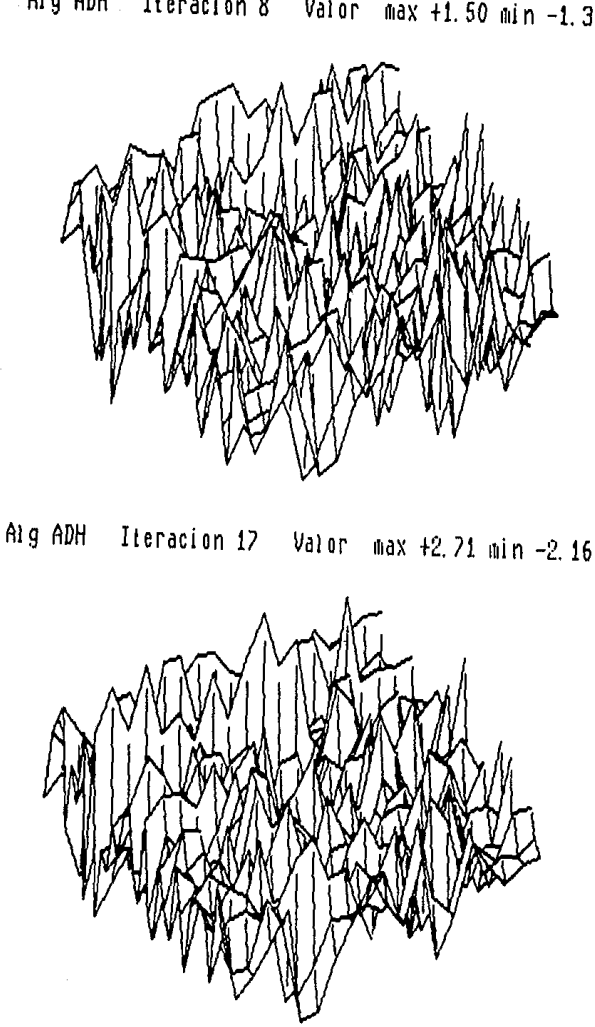


Fig. 4.9(b) Evolución durante el proceso de aprendizaje de las conexiones de la red con las 5 asociaciones de la Fig. 3.1 con el algoritmo adaptivo ADH. Nótese los cambios de escala.

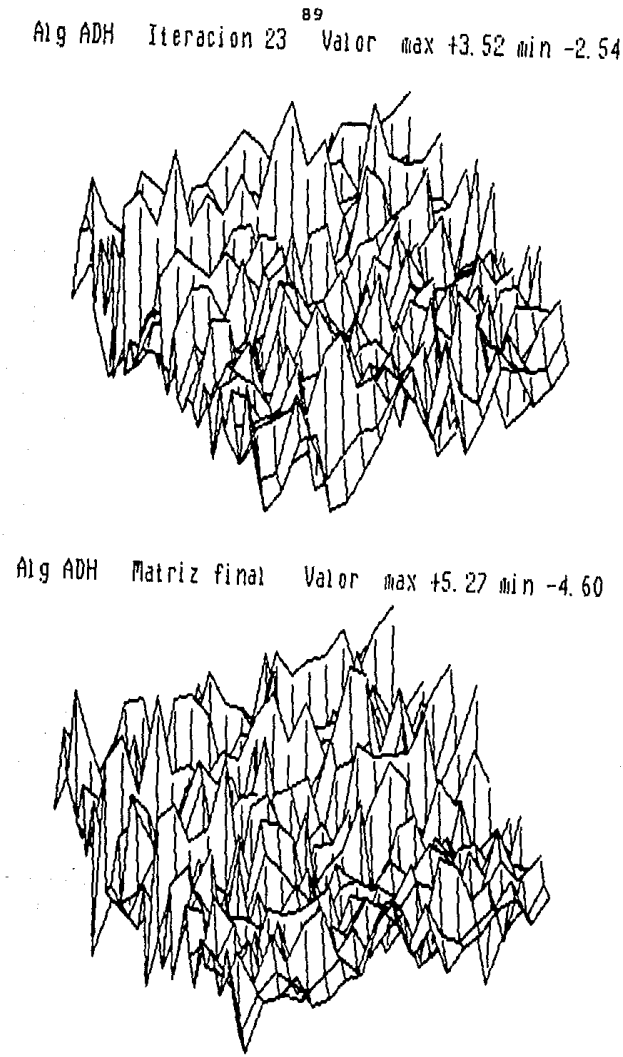


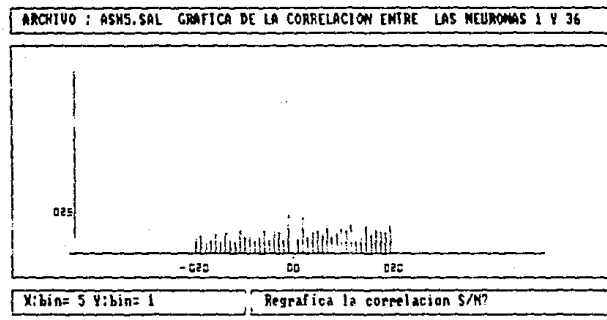
Fig. 4.9(c) Evolución durante el proceso de aprendizaje de las conexiones de la red con las 5 asociaciones de la Fig. 3.1 con el algoritmo adaptivo ADH. Nótese los cambios de escala.

4.3.5 HISTOGRAMAS DE CORRELACION CRUZADA

Se construyeron histogramas de correlación cruzada [1] de pares de trenes de impulsos de los procesadores. Estos trenes de impulsos se generan durante el proceso de aprendizaje de la red utilizando los algoritmos adaptivos.

En la **Fig. 4.10** se muestran las correlaciones cruzadas entre dos procesadores, escogidos al azar, indicados en la parte superior derecha de cada figura, para la red con las 5 asociaciones de la Fig. 3.1 con el algoritmo ASH. En los histogramas de correlación cruzada para los procesadores (neuronas) 1 y 36 de las capas A y B, respectivamente, se observa un "pozo" en el origen. Esto indica la posibilidad de inhibición mutua entre ambos procesadores, que es lo mismo afirmar que el procesador 1 de la capa A inhibe al procesador 1 (o 36, debido a que hay 35 procesadores en la capa A) de la capa B y viceversa; ya que en otro caso el "pozo" aparecería corrido a la izquierda o derecha. Esto se corrobora viendo que el peso bidireccional de la conexión en la matriz aleatoria (ver tabla 4.1) para los procesadores 1 y 36 (de la red) antes del proceso de aprendizaje es de -0.29, y después de concluir este proceso es de -0.61, lo que demuestra la interpretación del histograma: que se inhiben mutuamente.

Para los procesadores 17 y 38 de la red el histograma es muy ambiguo y no es posible decidir qué tipo de conexión se tiene. Si se ven los valores de los pesos en la matriz de la tabla 4.1, se encuentra que el peso inicial fue de -0.28 y el peso final de 1.21. Hubo una transición de inhibición mutua a excitación mutua entre los procesadores 17 y 38, pero no es posible detectarla con el histograma de correlación cruzada.



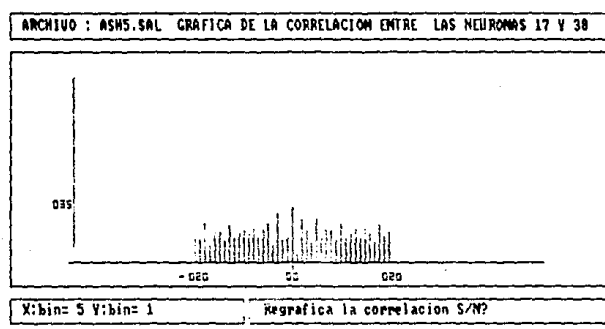
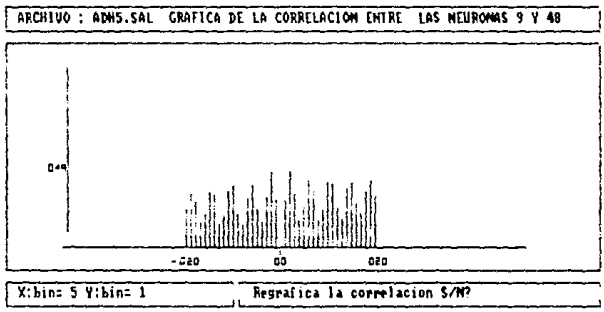


Fig. 4.10 Histogramas de correlación cruzada de trenes de impulsos de 2 procesadores para la red con 5 asociaciones utilizando el algoritmo adaptivo ASH.

En la **Fig. 4.11** se muestran unos histogramas de correlación cruzada para la red con 5 asociaciones utilizando el algoritmo ADH.

En el primer par de histogramas se muestra la correlación cruzada para los procesadores 9 y 48 de la red. Se observa que hay un "pozo" en el origen que indica inhibición mutua. Por otra parte, los pesos inicial y final fueron de -0.33 y -4.60, respectivamente.

Caso contrario con respecto a la correlación anterior, son los histogramas entre el segundo par de procesadores, 2 y 36 de la red, en el que se presenta otro caso de ambigüedad y en el cual los pesos cambiaron de -0.32 a 2.69. En el histograma se ve un pozo a la derecha del origen pero no es claro que significa.



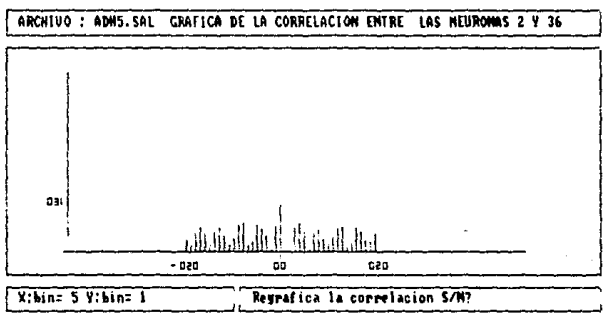


Fig. 4.11 Histogramas de correlación cruzada de trenes de impulsos de 2 procesadores para la red con 5 asociaciones utilizando el algoritmo adaptivo ADH.

CAPITULO 5
DISCUSION Y CONCLUSIONES

Una comparación de algoritmos como la realizada en este trabajo no ha sido publicada hasta la fecha. Kosko ha presentado algunos ejemplos, pero no comparaciones. La mayoría de los trabajos de otros autores son teóricos y no consideran ejemplos. Pareciera que algunos de estos trabajos teóricos no resistirían la prueba de ser utilizados en ejemplos directos y, así, queda en duda su posible utilización en una aplicación capaz de ser implementada en hardware. Sin embargo, hay que reconocer que con la explosión de información en el campo de las redes neuronales artificiales es muy difícil en este momento estandarizar la nomenclatura y comparar todos los métodos existentes.

El poderío del método holográfico para recuperar asociaciones es el resultado más interesante de este trabajo. Se ha demostrado que las recuperaciones en modo bipolar son más precisas que en modo binario y que el algoritmo HOLO es el más eficiente. El resto de los algoritmos estudiados son ineficientes para aprender asociaciones. No se sabe si la eficiencia del algoritmo HOLO se extiende a cualquier número de procesadores o si, como otros algoritmos de aprendizaje, tiene problemas de escalamiento. Tampoco se conoce su capacidad para tolerar ruido en cualquier condición. Mayor conocimiento sobre el algoritmo HOLO requerirá de más ejemplos y la revisión y extensión de sus bases teóricas.

No obstante, sería de mucho interés utilizar el método holográfico en aplicaciones como el reconocimiento de caracteres y manuscrito y, en general, de imágenes borrosas como las que se obtienen en criminología, ya que su rapidez y precisión de

recuperación lo hacen un método ideal para implementar en hardware.

Por otra parte, este estudio no se hizo únicamente para explorar aplicaciones tecnológicas de las redes neuronales, en particular, la arquitectura de la memoria asociativa bidireccional, sino también se pretendió utilizar dicha red o MAB como modelo de procesos cognoscitivos en los que existe adaptación y/o cierta forma de aprendizaje. Un experimento de este tipo en una preparación biológica es muy difícil, de ahí que la MAB se presta como una preparación artificial adecuada. El acceso a las activaciones de los procesadores y a las matrices de pesos permite estudiar la conducta de la red de una manera distribuida y puede tratar de observarse el proceso de adaptación o aprendizaje. Con este fin se estudiaron las gráficas de error que resultaron no útiles para esto, pues son un índice global de comportamiento demasiado comprimido y, por ello, no dicen nada acerca de la dinámica distribuída de la red.

Por otro lado, las gráficas globales de pesos si proporcionan cierta intuición visual de lo que hacen los algoritmos adaptivos durante el proceso de aprendizaje, pero no es suficiente. Se necesitan herramientas computacionales y gráficas más poderosas y versátiles para poder concluir algo con certeza y robustez.

En el caso de los histogramas de correlación cruzada, que es una herramienta más fina, se necesita estudiar el mapa completo de histogramas -para todos los pares, que son 35*20=700- y eso hace muy difícil observar o encontrar donde podrían localizarse cambios significativos durante el proceso de aprendizaje de las asociaciones. Un sistema experto podría ser útil en este renglón, pero no se ha probado todavía.

Se hicieron tales ejercicios por su novedad y actualidad. En el estudio de la dinámica del cerebro existe una problemática similar, sólo que amplificada. La resolución espacial y temporal de las técnicas disponibles no es suficiente para determinar los mecanismos que intervienen durante el aprendizaje o la percepción. Por ejemplo, la encefalografía tiene buena resolución

temporal, pero pésima resolución espacial; la tomografía, en sus diferentes versiones, tiene buena resolución espacial, pero su resolución temporal da la historia de la dinámica del cerebro de media hora en media hora. Si se toma en cuenta que la comunicación monosináptica entre neuronas es de 1 milisegundo, entonces se ve que se pierde el detalle. Se hacen intentos por mejorar esto, pero por ahora la tecnología adecuada todavía no está a la mano.

Una aspiración de este trabajo ha sido desarrollar un ambiente de práctica sobre dinámica compleja en paralelo. En una red neuronal artificial se tiene acceso y control de procesos síncronos, asíncronos y concurrentes. Se espera que tal práctica pueda conducir a inventar mejores métodos de análisis de la dinámica distribuída del sistema nervioso y de los sistemas complejos en general.

Se debe reconocer que este es un trabajo interdisciplinario, aunque muy modesto por la escasez de recursos con que contó. La dinámica de la red neuronal estudiada, es decir, la memoria asociativa bidireccional es también un caso muy interesante de interacción entre procesadores o neuronas artificiales. Estos procesadores podrían ser análogos de partículas y, así, lo que se puede estudiar es un caso especial del problema de muchos cuerpos. En este caso, las posibilidades de aprendizaje en la red pueden considerarse como el estudio de un cambio de fase, de no-aprendizaje a aprendizaje completo. Podría hablarse de un flujo de entrenamiento que produce una propiedad colectiva de la red: el aprendizaje de las asociaciones y aunado a ello la capacidad de reconocimiento aún en condiciones de alta incertidumbre. Estos son conceptos que en la actualidad se encuentran en desarrollo, pero que ya han demostrado su importancia en la Física Teórica contemporánea.

APENDICE
Manual de usuario del paquete MAB

El paquete MAB está programado en Turbo Pascal Versión 5.0. Consta de diez algoritmos de aprendizaje: cuatro deterministas y seis adaptivos, que están sobre cuatro programas ejecutables que se corren desde el programa MAB (MAB.EXE), también ejecutable. Estos programas son KSK.EXE, KHNN.EXE, JJG.EXE y HLGR.EXE. El paquete está separado en seis subdirectorios: MODELOS, PAT, MAT, DATOS, ERROR y CORRELAC.

En el subdirectorio MODELOS se encuentran los programas ejecutables *.EXE y las utilerías *.TPU correspondientes a cada uno de los algoritmos. En los subdirectorios PAT y MAT se encuentran los archivos de las asociaciones (pares de patrones) y matrices (matrices de pesos), respectivamente. Los archivos de patrones llevan como extensión *.PAT y los de matrices *.MAT. Para los patrones ruidosos la extensión del archivo es *.PRD. En el subdirectorio DATOS se encuentran las utilerías para gráficas y datos sobre la corrida de los programas ejecutables. En el subdirectorio ERROR se encuentran los archivos de gráficas de variación del error, con extensión *.ERR. También en este subdirectorio se encuentra el paquete GRAPHPC para grafica. los datos de los archivos. Esto se hace corriendo desde el sistema DOS, :\>GRAPHPC -a 1 1 [NOMBRE].ERR.

En el subdirectorio CORRELAC, se encuentra el paquete para graficar los histogramas de correlación cruzada de neuronas [4]. Los archivos que contienen los trenes de impulsos durante el proceso de aprendizaje (archivos de salida para CORRELAC) llevan extensión *.SAL. Esta es una visión general del paquete.

Para una corrida sobre el paquete MAB, se debe teclear MAB, de la forma:

:\>MAB

y presionar *Intro* que hará mostrar una presentación del paquete mencionando al autor y director de la tesis, como en la **Fig. A.1.**

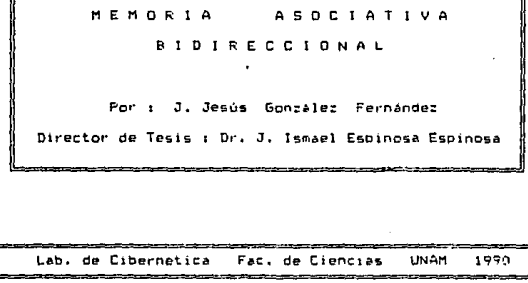


Fig. A.1 Presentación del paquete.

Para continuar se presiona cualquier tecla. En primer lugar aparecerá un menú de algoritmos, como se muestra en la **Fig. A.2,** que señala la barra que, con las flechas, se puede mover para elegir el algoritmo presionando la tecla *Intro*. Este proceso de elección es el mismo para todos los menús del paquete. En caso de algún error en la elección, presionando la tecla *Esc* se regresa al paso anterior.

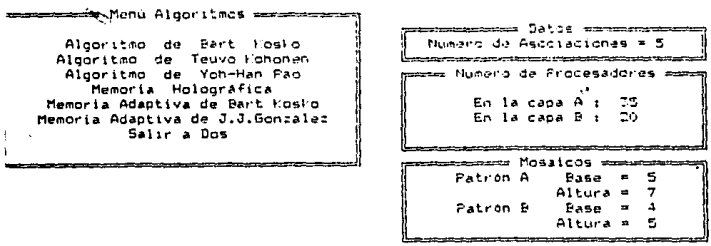


Fig. A.2 Menú de Algoritmos.

En el Menú de Algoritmos se pueden leer, en este orden, los cuatro algoritmos deterministas, los dos últimos son algoritmos adaptivos. Para los algoritmos adaptivos de Kosko: Aprendizaje por Señal de Hebb (ASH) y Aprendizaje Diferencial de Hebb (ADH) se debe escoger la opción Memoria Adaptiva de Bart Kosko y aparecerán los dos algoritmos anteriores, como lo muestra la **Fig. A.3.**

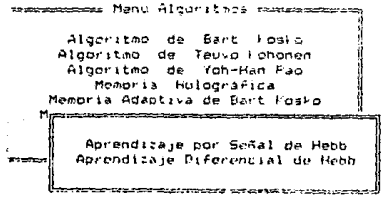


Fig. A.3 Menú de Algoritmos, Opción Memoria Adaptiva de Bart Kosko.

El mismo caso se presenta para los cuatro algoritmos adaptivos propuestos por el autor de la tesis. En este caso aparecerán cuatro opciones, como se muestra en la siguiente figura.

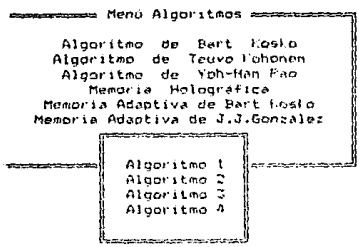


Fig. A.4 Menú de Algoritmos, Opción Memoria Adaptiva de J.J. González.

Una última opción es el abandono del paquete. El caso se muestra en la **Fig. A.5**.

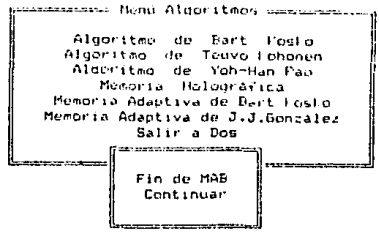


Fig. A.5 Abandono del paquete MAB.

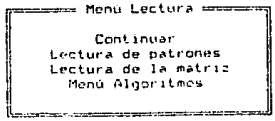
Una vez que se ha escogido el algoritmo a emplear, se deben dar los datos sobre el número de asociaciones y procesadores en cada capa de la red. En la parte inferior izquierda de la pantalla se indican los valores máximos permitidos para esta versión. En caso de necesitar el usuario otra versión con más asociaciones y procesadores, se puede comunicar a la siguiente dirección:

Laboratorio de Cibernética
3er. piso, Departamento de Física
Facultad de Ciencias U.N.A.M.
Tel. 550-5215 Ext. 3922
Cd. Universitaria

Además se deben de indicar los valores de las bases y alturas de los mosaicos, que se muestran como en las evoluciones del Cap. 3. Así, al multiplicar la base por la altura del mosaico debe ser igual al número de proecesadores en la capa, de otra manera no se podrá continuar con la corrida, ver **Fig. A.2**.

Hecho lo anterior, el paquete hará la corrida del algoritmo escogido.

El segundo es el menú de Lectura, ver **Fig. A.6**, independientemente del algoritmo escogido.



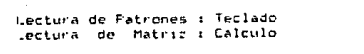


Fig. A.6 Menú de Lectura.

Se tiene la opción de leer los patrones de archivo o construirlos (**Fig. A.7**), lo mismo para la matriz de pesos, ya sea leerla de un archivo o calcularla (**Fig. A.8**). Por default, la opción para Lectura de patrones es construírlos: *Teclado;* y la opción para la matriz de pesos es calcularla: *Cálculo*. La opción continuar indica el siguiente paso, según lo elegido para lectura. Si la opción para lectura de patrones es *Archivo*, entonces se debe teclear el nombre del archivo sin extensión, ya construído bajo cierto formato y que con las teclas F3, las flechas e Intro, se puede elegir. Si la opción para lectura de patrones fué *Teclado*, aparecerán los mosaicos para la construcción de las asociaciones, que en un principio estarán todos los procesadores en estado no-excitado y que con la tecla Intro se cambiará el estado del procesador. Para moverse sobre el mosaico se utilizan las flechas. Una vez que el patrón se haya construído (mosaico), con la tecla *Esc* se procede a construir los mosaicos siguientes.

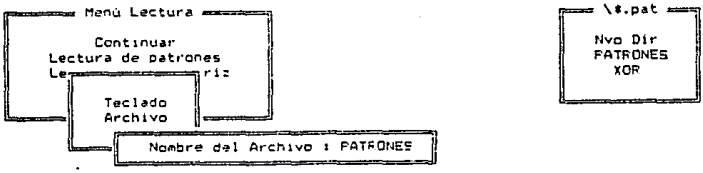


Fig. A.7 Lectura de patrones.

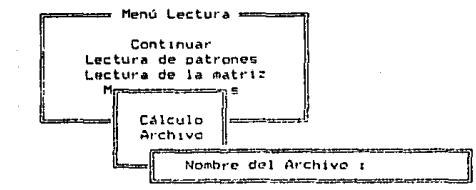
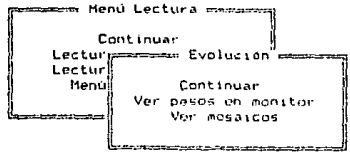


Fig. A.8 Lectura de la matriz de pesos.

En el caso de haber elegido un algoritmo adaptivo en el Menú de Algoritmos, aparecerán otra serie de menús para el proceso de aprendizaje, antes de llegar a la recuperación de patrones.

El siguiente menú es el de la Evolución, es decir, del proceso antes mencionado. En este menú se elige entre tres opciones (**Fig. A.9**): Ver pesos en monitor muestra los valores de pesos en la pantalla, Ver mosaicos muestra la evolución de los procesadores si se activan o no y por último Continuar sólo muestra una pequeña ventana que indica el valor del error y de la iteración, esta última es dada por default.



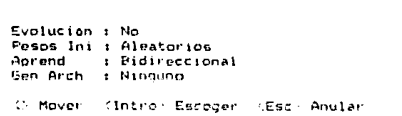
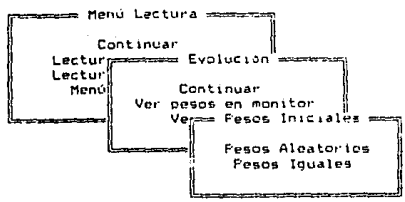


Fig. A.9 Menú Evolución.

En la **Fig. A.10** se muestra el menú de Pesos Iniciales en donde se escoge la opción de generar valores de Pesos Aleatorios o que todos sean Pesos Iguales, para comenzar con el proceso de aprendizaje. Para la primera opción, dada por default, los valores de los pesos no son mayores de 0.5 ni menores de -0.5.



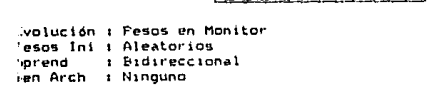


Fig. A.10 Pesos Iniciales.

Existe la forma de guardar información acerca del proceso de aprendizaje. En la **Fig. A.11** aparecen cuatro opciones que indican la generación de archivos de salida. Salida para Correlac crea un archivo de salida (*.SAL) para el programa CORRELAC en donde se observan los estados activados y no-activados de los procesadores durante el aprendizaje, asimismo la opción Gráfica de Error genera un archivo de salida (*.ERR) para el paquete GRAPHPC. La opción Ambos crea los dos archivos para los dos paquetes. Los archivos de salida se almacenan en los subdirectorios que le correspondan. Por default, se da la primera opción: Ninguno, que no generará archivos de salida.

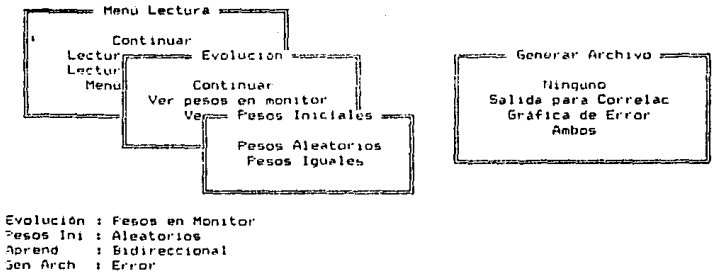


Fig. A.11 Generar Archivo.

Si se ha elegido en el menú Evolución la opción Ver pesos en monitor, aparecerá una ventana con los valores de pesos según la opción del menú de Pesos Iniciales (**Fig. A.12**). El valor del peso superior izquierdo corresponde a la primera columna y primer renglón de la matriz, y el valor del peso inferior derecho corresponde al renglón número 18 y columna 7 de la matriz. En la parte inferior aparece otra ventana en la que se indican el valor del error para la iteración dada.

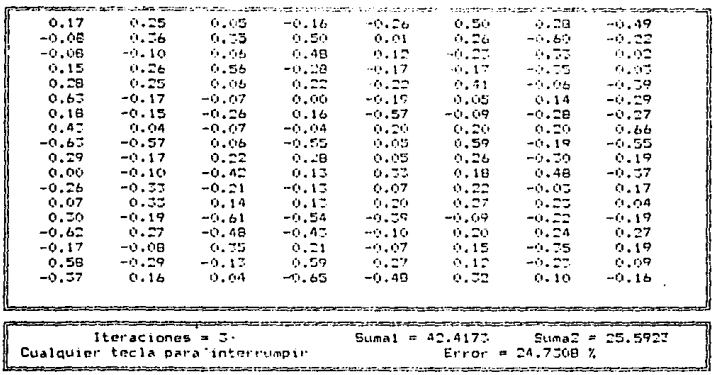


Fig. A.12 Ventana de pesos durante el proceso de Aprendizaje.

En caso de presionar Cualquier tecla para interrumpir el proceso de aprendizaje, aparecerá otro pequeño menú con las opciones: Salir, Continuar, Paso y Cambiar las coordenadas de la matriz de pesos (**Fig. A.13**). La opción Salir aborta el proceso de aprendizaje, Continuar lo habilita, Paso indica a cada cuántos pasos mostrará los valores de los pesos (por default, Paso=1), y por último Cambiar coor de matriz M hace cambiar las coordenadas de la parte superior izquierda de la matriz.

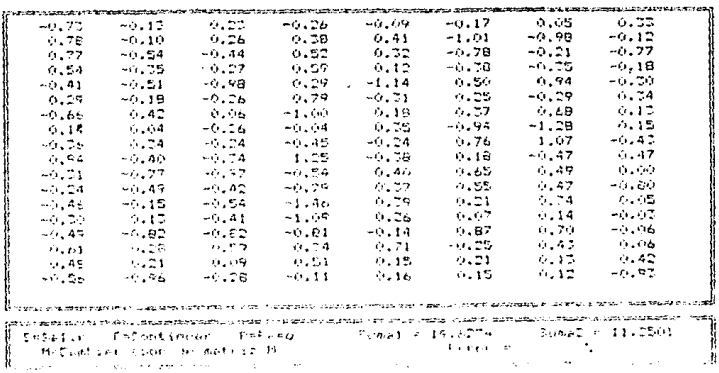


Fig. A.13 Opciones durante el proceso de Aprendizaje.

El menú Opciones ofrece varios cambios para la etapa de recuperación de patrones (**Fig. A.14**). La opción Vectores ofrece dos facetas: binario y bipolar, que indican el modo de recuperación. Con la tecla *Intro* se cambia la opción y *Esc* la acepta; binario está por default. La tercera opción escoge la Función de Activación para los procesadores de las capas. Por default, ambas capas tienen la función: On-Off. La opción Corrección de Hopfield aparece para todos los algoritmos excepto para el algoritmo determinista HOLO. Esta oopción habilita o no la aplicación de la corrección, en caso afirmativo se dará el valor del parámetro ϵ, de acuerdo a lo mencionado en el Cap. 2, siguiendo *Intro* y *Esc* para aceptar. La opción Info da información general sobre la corrida. La opción Guardar en Archivo ofrece dos casos a elegir: una es guardar las asociaciones y la otra almacenar la matriz de pesos. La elección es con *Intro* y *Esc* para anular. En ambos casos dar el nombre sin extensión y se almacenará en el subdirectorio que le corresponda.

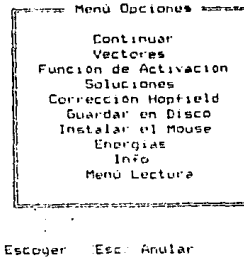


Fig. A.14 Menú de Opciones.

La opción Continuar presentará enseguida un menú con tres opciones a elegir una de ellas: Capa A, Capa B y Ambos, que indica en qué capa se va a presentar el patrón ruidoso a la red para la recuperación, ver Fig. A.15. Ambos indica que se llenarán los dos mosaicos como patrón ruidoso.

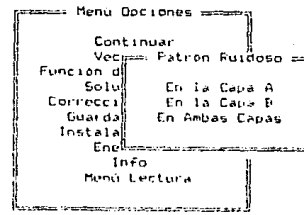


Fig. A.15 Menú para el patrón ruidoso.

Una vez hecho lo anterior aparecerá otro menú de opciones para el patrón ruidoso a considerar, como se muestra en la Fig. A.16. Teclado indica que habrá que construir el patrón ruidoso, Archivo puede leer el patrón ruidoso de un archivo en disco, que con la tecla F3 presentanra el irectorio; y por último, Tec y Sal hace las dos opciones anteriores, es decir, se construye el patrón ruidoso para después almacenarlo en un archivo.

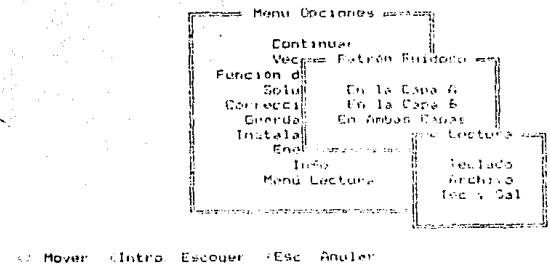


Fig. A.16 Lectura del patrón ruidoso.

Hasta entonces aparecerán los mosaicos, en donde se mostrará la recuperación de acuerdo a las opciones, ver **Fig. A.17**. Si se escogió la opción Teclado entonces con la tecla *Intro* se cambiará el estado activado o no-activado del procesador que indique el cuadro guía y con la tecla *Esc* se procede a la recuperación. Con la tecla F3 aparecerán las asociaciones en la parte superior derecha, que con las flechas se pueden mostrar, y presionando *Return* se puede presentar otro patrón ruidoso; y se se presionó la tecla *Esc* entonces aparecerá el Menú de Opciones.

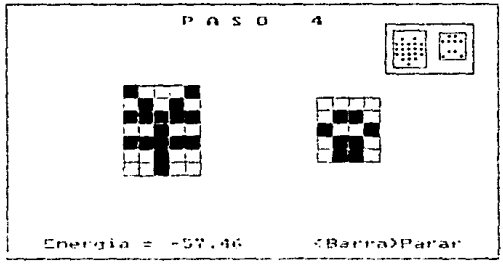


Fig. A.17 Recuperación de patrones.

REFERENCIAS

ARTICULOS

[1] M. Abeles y M. H. Goldstein, "Multispike Train Analysis," Proc. IEEE, vol. 65, no. 5, pp. 762-773, May 1977.

[2] I. Espinosa E., "Las Redes Neuronales Artificiales," ICYT, vol. 12, no. 163, pp. 37-43, Abr. 1990.

[3] I. Espinosa E. y J. J. González, "Estudio sobre Filtro de Ruido y Memoria Asociativa Bidireccional utilizando un simulador de Redes Neuronales," Memorias del XXXII Congreso Nacional de Física, vol. 3, no. 2, May/Ago. 1989.

[4] I. Espinosa E. y M. Alcántara G., "Neurored: Simulador de Redes Neuronales y Analizador de Conectividad Funcional," Memorias del XV Congreso Nacional de Ingeniería, Sep. 1989.

[5] J. J. González e I. Espinosa E., "Red Neuronal Artificial Tipo Memoria Asociativa Bidireccional," Rev. Mex. Ing. Biomed., Nov. 1990.

[6] R. Hecht-Nielsen, "Neurocomputing: picking the human brain," IEEE Spectrum, pp. 36-41, Mar. 1988.

[7] F. Heredia L.; J. J. González; I. Espinosa E. y R. Lara V., "Estudios sobre Memoria Asociativa Bidireccional utilizando dos simuladores diferentes de Redes Neuronales," Memorias de la Quinta Conferencia Internacional: Las Computadoras en las

Instituciones de Educación e Investigación CA-UNAM-UNISYS, Nov. 1989.

[8] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Nat. Acad. Sci., U.S., vol. 79, pp. 2554-2558, Apr. 1982.

[9] J.J. Hopfield, "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons," Proc. Nat. Acad. Sci., U.S., vol. 81, pp. 3088-3092, May 1984.

[10] J.J. Hopfield, D. I. Feinstein y R. G. Palmer, "'Unlearning' has a Stabilizing Effect in Collective Memories", Nature, vol. 304, no.14, pp. 158-159, July 1983.

[11] T. Kohonen, "Correlation Matriz Memories", IEEE Trans. Comput., vol. C-21, no. 4, pp. 353-359, Apr. 1972.

[12] T. Kohonen, "An Adaptive Associative Memory Principle," IEEE Trans. Comput., vol C-23, no. 4, pp. 444-445, Apr. 1974.

[13] T. Kohonen y M. Ruohonen, "Representation of Associated Data by Matrix Operators", IEEE Trans. Comput., vol. C-22, no. 7, July 1973.

[14] B. Kosko, "Constructing an Associative Memory," Byte, vol. 12, no. 10, pp. 137-144, Sept. 1987.

[15] B. Kosko, "Adaptive Bidirectional Associative Memories," Appl. Opt., vol. 26, no. 23, pp. 4947-4960, Dec. 1, 1987.

[16] B. Kosko, "Bidirectional Associative Memories," IEEE Trans. Syst., Man, Cybern., vol. SMC-18, no. 1, pp. 49-60, Jan./Feb. 1988.

[17] B. Kosko, "Unsupervised Learning in Noise," IEEE Trans. Neural Networks, vol. 1, no. 1, pp. 44-57, Mar. 1990.

[18] W. A. Little, "The Evolution of Non-Newtonian Views of Brain Function," Concepts in Neuroscience, vol. 1, no. 1, pp. 149-164, 1990.

[19] W. S. Mc Culloch y W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bull. Math. Biophys., vol. 5, pp. 115-133, 1943.

[20] K. Murakami y T. Aibara, "An Improvement on the Moore-Penrose Generalized Inverse Associative Memory", IEEE Trans. Syst., Man, Cybern., vol. SMC-17, no. 4, pp. 699-707, July/Aug. 1987.

[21] G. S. Stiles y D.-L. Denq, "A Quantitative Comparison of the Performance of Three Discrete Distributed Associative Memory Models," IEEE Trans. Syst., Man, Cybern., vol SMC-36, no. 3, pp. 257-263, Mar. 1987.

[22] G. Tesauro y Y. H. Subutai A., "Asymptotic Convergence of Backpropagation," Neural Networks, vol. 1, no. 3, pp. 382-391, 1989.

[23] Y.-F. Wang, J. B. Cruz y J. H. Mulligan, "Two Coding Strategies for Bidirecctional Associative Memory," IEEE Trans. Neural Networks, vol. 1, no. 1, pp. 81-92, Mar. 1990.

LIBROS

[24] N. Ahmed y K. R. Rao, Orthogonal Transform for Digital Signal Processing. Berlín and New York: Springer Verlag, 1970.

[25] A. Albert, Regression and Moore-Penrose Pseudoinverse. New York: Academic, 1972.

[26] G. Arfken, Mathematical Methods for Physicists. New York and London: Academic Press, 1971.

[27] L. Elsgoltz, Ecuaciones Diferenciales y Cálculo Variacional. Moscú: MIR, 1983.

[28] T. Geszti, Physical Models of Neural Networks. World Scientific, 1990.

[29] D. H. Hubel, C. F. Stevens, El cerebro, Libros de Investigación y Ciencia. Barcelona: Labor, 1981.

[30] M. L. Minsky y S. A. Papert, Perceptrons. MIT Press, 1988.

[31] K. Ogata, Modern Control Engineering. Prentice Hall, 1970.

[32] Y.-H. Pao, Adaptive Pattern Recognition and Neural Networks. Addison Wesley, 1989.

[33] R. Tapia, Las células de la mente. SEP-FCE, 1987.

[34] L. Viana, Memoria Natural y Artificial. SEP-FCE, 1990.

MANUALES

[35] PAINTBRUSH User's Guide. Microsoft Corporation, 1987.

[36] PC-MATLAB User's Guide V. 1.51. The MathWorks Inc., 1984.

[37] TURBO PASCAL Reference Guide V. 5.0. Borland Int., 1988.