CAPITULO 2

TIPOS DE REDES NEURONALES

2.1.- Introducción

Existen numerosos tipos de redes neuronales, que pueden ser clasificadas y diferenciadas de variadas formas. De éstas, las más clásicas se refieren a su estructura, tipo de aprendizaje y operación, y a su aplicación.

En lo referente a su estructura, la atención se concentra en las capas de la red, neuronas, conexiones y funciones de activación. La clasificación según el aprendizaje y operación básicamente se concentra en el algoritmo de aprendizaje que se utilice, tipo de supervisión que posea y en como opera la red en su evocación.Por último, su aplicación puede ser enfocada de variadas formas: puede referirse a si la red es hetero-asociativa (caso en que el vector de entrada es asociado con un vector de salida de distinto tipo) o auto-asociativa (caso en que la entrada es asociada a un vector igual en la salida). Por otro lado, la aplicación también puede ser enfocada hacia la función que le asigne el diseñador a la red (predicción, clasificación, asociación, filtraje, conceptualización y optimización).

Se ha incluido en este capítulo una descripción completa de diferentes algoritmos de aprendizaje, con el fin de complementar y ayudar al diseñador en la creación de alguna aplicación. Los algoritmos aquí presentados están enfocados a ser aplicados no sólo al control automático, sino que también a otros campos como son el reconocimiento de patrones, filtros y otros.

2.2 Tipos de redes

A continuación presentaremos un conjunto de redes neuronales que están clasificadas según el tipo de aprendizaje y evocación. Dicha clasificación está basada en la presentada por Simpson en su libro "Artificial Neural Systems" [3], en la que distinguen cuatro tipos de redes:

(i) Redes con aprendizaje no supervisado y evocación retroalimentada.
(ii) Redes con aprendizaje no supervisado y evocación hacia adelante.
(iii) Redes con aprendizaje supervisado y evocación retroalimentada.
(iv) Redes con aprendizaje supervisado y evocación hacia adelante.

A continuación se describirán las redes más representativas en cada una de ellas, comenzando con las redes supervisadas y de evocación hacia adelante, por servir algunas de éstas de base para la comprensión de las demás.

2.3.- Redes neuronales con aprendizaje supervisado y evocación hacia adelante.

2.3.1.- Adaline (Adaptive Linear Element)

- **Descripción**

Este tipo de red es implementada en base a un tipo de neurona denominada Adaline. El esquema básico de un Adaline corresponde a una neurona de McCulloch-Pitts, seguida de una función de activación lineal, de la forma F(Zj) = Zj. Sus entradas pueden tomar cualquier valor, y a ellas se les aplica una ponderación determinada de acuerdo a la ley del error cuadrático mínimo. Este proceso de ponderación incluye además, un peso denominado "polarización" el cual está siempre conectado a una entrada fija +1.

En la figura 7 se presenta el esquema básico de este tipo de neurona:

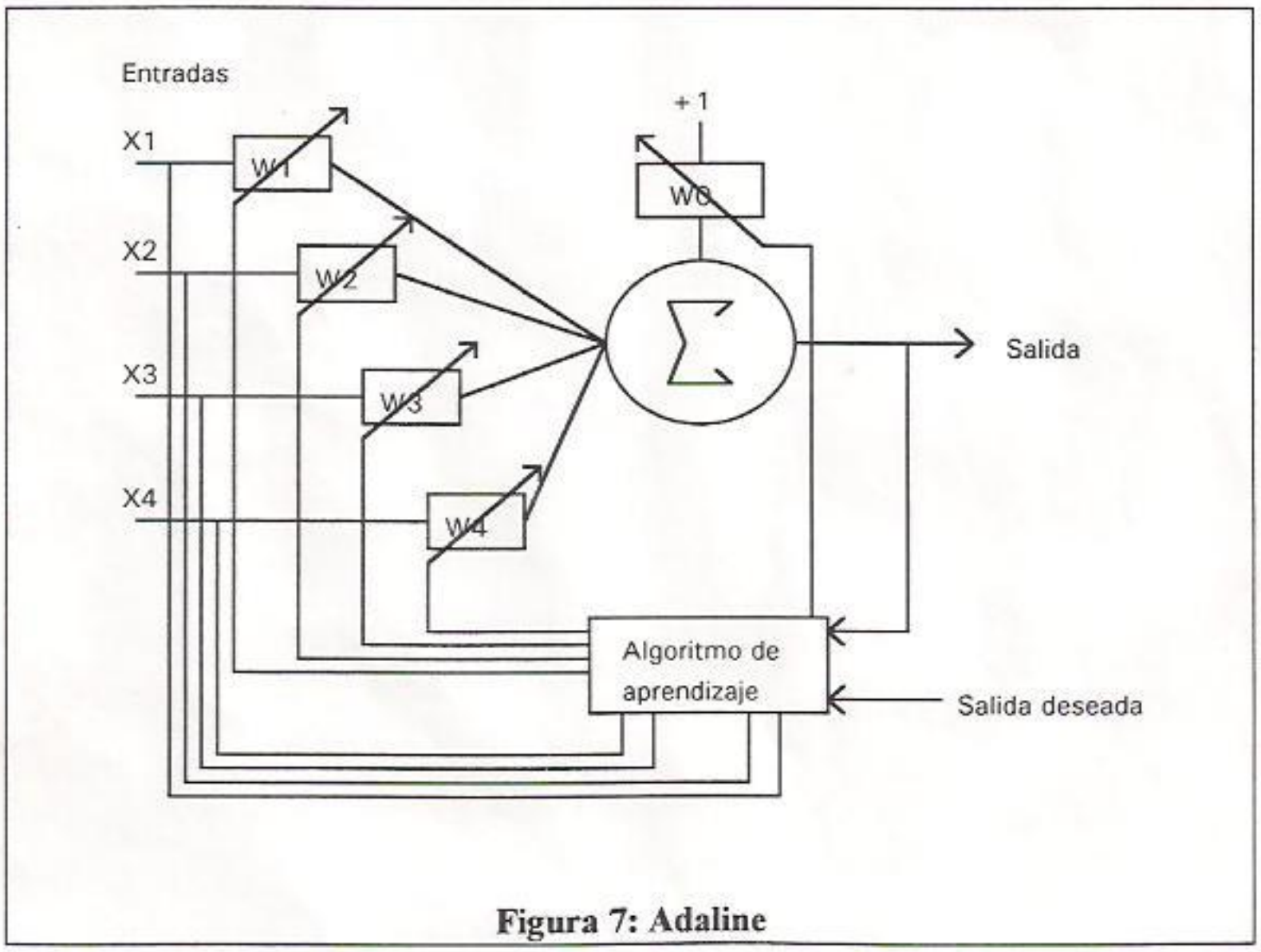

Figura 7: Adaline

Si bien el proceso adaptivo ha sido simplificado y se muestra un caso que sólo presenta una salida, es claro que un sistema con muchas salidas paralelas es implementable directamente por varias de las unidades mostradas. La figura 8 presenta una red de Adaline.

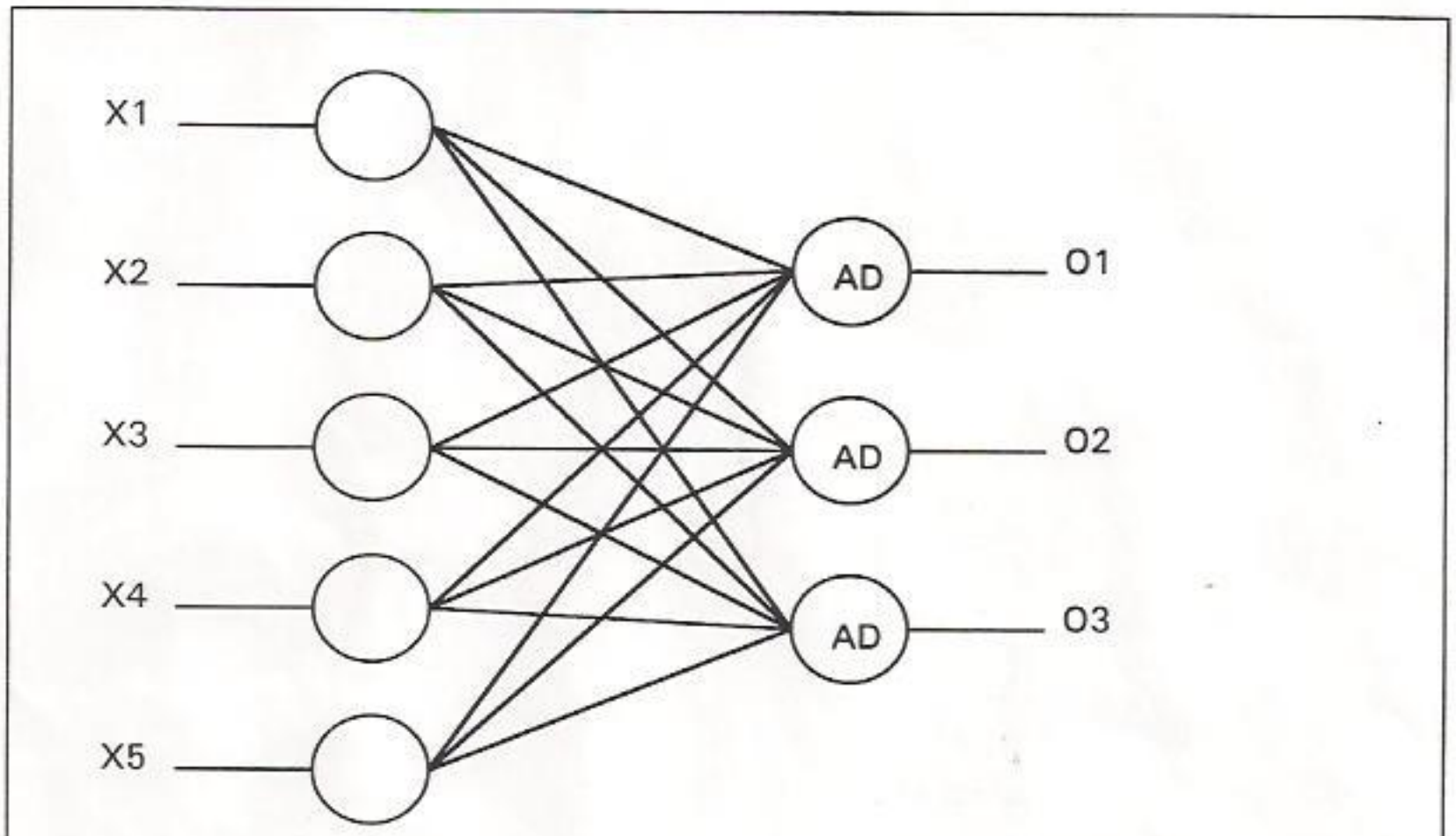

Figura 8: Red de Adaline

Dado que un Adaline es un modelo lineal, si se interconectan capas de Adaline no se tendrá una mayor eficiencia computacional debido a que una combinación lineal de estas unidades puede ser llevada a cabo por una sola unidad lineal [Apéndice 1]. Por lo tanto, cuando se hable de una red de Adaline se entenderá que corresponde a una red de una sola capa.

- **Algoritmo de aprendizaje**

No obstante que el sistema físico básico es continuo en el tiempo, resulta más fácil estudiar el sistema como un proceso en tiempo discreto $\{t_k,\ k = 1,2,...\}$.

El propósito de este dispositivo es producir un valor dado Tk en su salida Ok cuando un conjunto de valores Xi , i = 1, 2, ..., n es presentado en la entrada. El problema es determinar los coeficientes Wi,k , i = 0, 1, 2, ..., n, de tal forma que la transferencia entrada-salida sea válida para un gran número de señales arbitrariamente elegidas. Luego, el ajuste de los pesos para cada neurona puede ser expresado por la recursión

$$O_k = \sum_{i=1}^{n} X_{i,k} \bullet W_{i,k} + W_{0,k} \qquad (2.1)$$

$$W_{i,k+1} = W_{i,k} + \alpha_k (T_k - O_k) X_{i,k} \qquad (2.2)$$

donde α_k es un escalar suficientemente pequeño. Puede ser demostrado que si los patrones son linealmente independientes, Wi,k converge a límites únicos. Frecuentemente, sin embargo, las correcciones permanecerán oscilatorias. Para prevenir esto, α debe ser decrementado mediante una aproximación estocástica [Apéndice 2] de la que resulta :

$$\alpha_k = 1 / \| X \|^2 \qquad (2.3)$$

Así, la ecuación (2.2) quedaría como

$$W_{i,k+1} = W_{i,k} + \frac{\beta (T_k - O_k) X_{i,k}}{\|X_k\|^2} \qquad (2.4)$$

donde $X_k = \{ X_1, X_2, ..., X_n \}$ es el vector de entrada en el instante k. En (2.3) Y (2.4), $\| \cdot \|$ denota norma.

En este caso se ha introducido la constante β, conocida con el nombre de coeficiente de aprendizaje. La modificación de dicha constante influenciaría el tamaño de los pasos del vector de los pesos mientras se realiza la adaptación. Un primer criterio es que esta constante debe ser positiva y menor que 1. Muchos autores sugieren que un rango típico de valores es $0.1 < \beta < 1$, dependiendo del problema.

- **Entrenamiento**

El entrenamiento que posee esta red es del tipo supervisado, por lo que cada entrada que se presente debe ir acompañada de la salida que se desea. Las entradas son ponderadas y luego sumadas, lo que produce una salida que se compara con la deseada, originando un error. Este error es el que se utilizará en el ajuste de los pesos según el algoritmo de aprendizaje.

Inicialmente todos los pesos deben poseer valores aleatorios. Si todos los pesos iniciales son iguales el procedimiento de adaptación podría sumar o restar siempre la misma cantidad, y así caer en un mínimo local en el espacio de los pesos. El espacio de los pesos es un espacio Euclidiano de dimensión igual al número total de pesos; cada punto en este espacio es representado como un vector con los valores de los pesos.

2.3.2.- Madaline (Multiple Adaline)

- **Descripción**

Este tipo de red está formada por varias Adaline en paralelo, interconectadas en forma total con la capa de entrada, y seguidas cada una de ellas por una función de umbral del tipo signo. Posee una sola neurona de salida, denominada Madaline, la que realiza la función de entregar el valor que corresponda a la mayoría de sus entradas (ver figura 9).

Como este tipo de red tiene una única salida binaria, ésta puede ser usada sólo para discriminar entre dos clases. Si se desea discriminar entre más de dos clases, varias redes de Madaline independientes pueden ser usadas, una para cada par de clases.

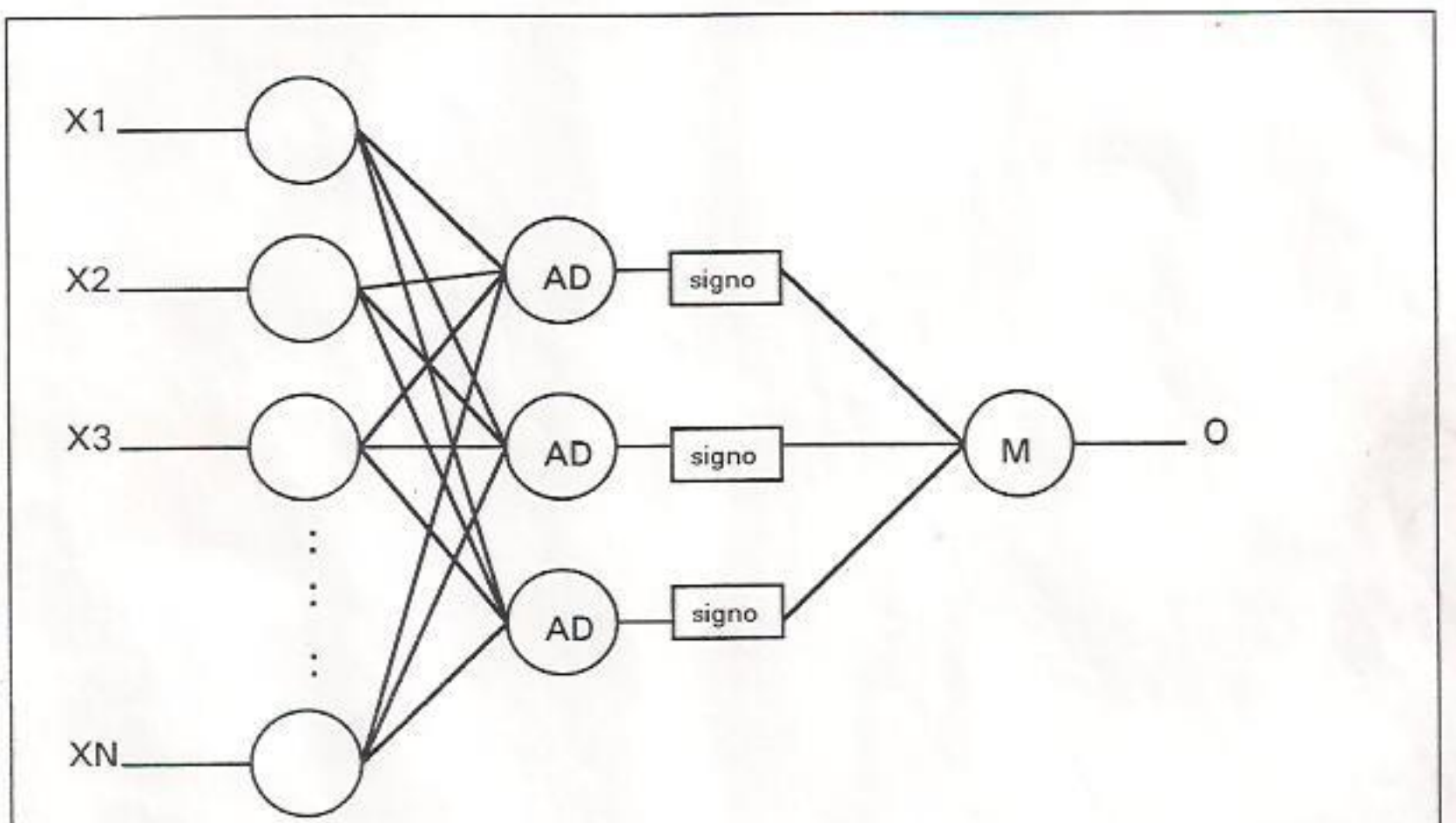

Figura 9: Madaline

Si se denota por Y_k la salida de la neurona k de la capa de las Adaline, entonces la salida de la red, O, corresponde a:

$$O = \begin{cases} -1, \text{ si } S < 0. \\ +1, \text{ si } S \geq 0. \end{cases} \qquad (2.5)$$

donde

$$S = \sum_{k=0}^{m} Y_k \qquad (2.6)$$

$$Y_k = \sum_{i=1}^{n} X_i \bullet W_{i,k} + W_{0,k} \qquad (2.7)$$

- **Algoritmo de aprendizaje**

Cuando un patrón de entrada y su salida deseada son presentados a la red, ésta genera una salida que puede coincidir con la deseada, en cuyo caso no se realiza el algoritmo de aprendizaje. En caso contrario, se lleva a cabo el algoritmo de aprendizaje. El problema en este último caso es determinar el error para adaptar los pesos de las Adaline, ya que no se especifica cual es su salida deseada. Este problema, para este caso en particular, se resuelve fácilmente: como se desea que la salida de la red sea igual a la deseada, dicho valor debe corresponder a la mayoría de las salidas de las Adaline, por lo que la salida deseada de esta última será la misma que la de la red.

Por lo tanto, el algoritmo que modifica los pesos de las neuronas Adaline queda determinado a partir de la recursión descrita por (2.4), de la que, introduciendo la notación según (2.7) y denotando la salida deseada de la red por T, se obtiene:

$$W_{i,k+1} = W_{i,k} + \frac{\beta\,(T - Y_k)\,X_{i,k}}{\|X\|^2} \qquad (2.8)$$

Luego, como las entradas toman sólo los valores +1 y -1, resulta que:

$$\|X\|^2 = X_1^2 + X_2^2 + X_3^2 + \ldots + X_N^2 = N \qquad (2.9)$$

en que N es el número de entradas a la red. Por tanto, la adaptación de los pesos queda determinada por la recursión:

$$W_{i,k+1} = W_{i,k} + \frac{\beta\,(T - Y_k)\,X_{i,k}}{N} \qquad (2.10)$$

- **Entrenamiento**

Durante el entrenamiento el patrón de entrada y la salida deseada son presentados a la red (entrenamiento supervisado). El aprendizaje es realizado sólo por las neuronas Adaline, las que usan un algoritmo de aprendizaje similar al utilizado para una red Adaline.

En general, este tipo de red es usado con entradas de tipo binario, con valores de +1 y -1, utilizándose a menudo, en sus aplicaciones, una codificación especial para sus entradas. Por ejemplo: si X es el valor de una variable de entrada a la red, entonces una codificación binaria y linealmente independiente de ella podría ser:

Valor de X	Codif. 1	Codif. 2
Entrada número	1 2 3 4 5	1 2 3 4 5
$X < 29$	1 -1 -1 -1 -1	1 1 1 1 1
$29 \leq X < 30$	-1 1 -1 -1 -1	1 1 1 1 -1
$30 \leq X < 31$	-1 -1 1 -1 -1	1 1 1 -1 -1
$31 \leq X < 32$	-1 -1 -1 1 -1	1 1 -1 -1 -1
$32 \leq X$	-1 -1 -1 -1 1	1 -1 -1 -1 -1

2.3.3.- Perceptrón

- **Descripción**

Este sistema adaptivo, sugerido por Rosenblatt, es una combinación de diferentes unidades, de las cuales la primera capa (S) simplemente incluye los sensores del ambiente. Las señales producidas por los sensores son combinadas en la segunda capa de unidades llamadas "**Elementos Asociativos**" (A). Esta capa está conectada en forma total o aleatoria a la primera y opera decodificando los patrones de entrada de modo de detectar sus características específicas: una respuesta activa se obtendrá si y sólo si se da la combinación binaria del valor de la señal de entrada. Según esto, los "**Elementos Asociativos**" van a actuar como un Adaline con pesos fijos y con su salida conectada a una función de transferencia del tipo umbral, que entrega valores lógicos 0 y +1. La tercera capa, de los elementos de respuesta (R), constituye el propio sistema de aprendizaje: las conexiones de A a R son realizadas a través de conexiones variables, similares a las del Adaline.

La figura 10 presenta el esquema básico de una red Perceptrón:

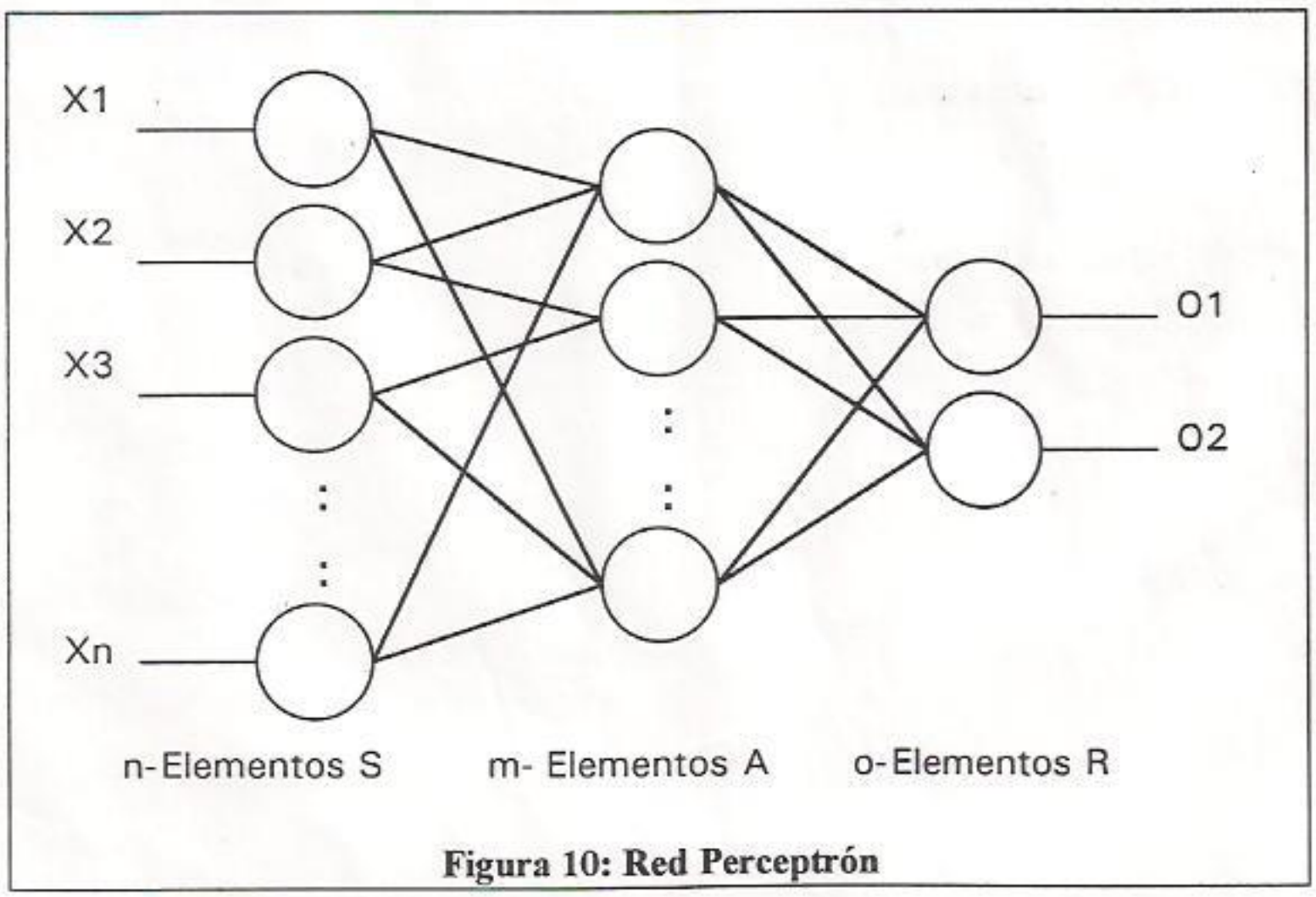

Figura 10: Red Perceptrón

La diferencia más notable entre las funciones adaptivas del Perceptron y el Adaline es que cada elemento R es un disparador de umbral; éste posee sólo dos estados de salida, que dependen de la entrada y de una función discriminante. Si se denota las señales producidas por los elementos A por Yi, los pesos de las respectivas conexiones de entrada adaptiva por Wi,k, y la salida de una unidad R por Ok, se asume que

$$O_k = \begin{cases} 0, \text{ si } S_k < \delta. \\ 1, \text{ si } S_k \geq \delta. \end{cases} \qquad (2.11)$$

donde δ es el umbral de discriminación y

$$S_k = \sum_{i=1}^{m} Y_i \bullet W_{i,k} \qquad (2.12)$$

- **Algoritmo de aprendizaje**

Asumiendo que un conjunto dado de señales de entrenamiento es {Yi,k}, por lo cual la señal Yi,k producida por los elementos de A es en principio restringida a valores binarios, Yi,k $\in$ {0,1}. Si la respuesta binaria esperada es Tk , una de las reglas de entrenamiento sugeridas cambiará sólo aquellos pesos de entrada para los cuales Yi,k = 1 (activa). Los valores Wi,k son incrementados o decrementados por un valor constante λ de acuerdo a la regla

$$W_{i,k+1} = W_{i,k} + \lambda (T_k - O) Y_{i,k} \qquad (2.13)$$

Algunos autores sugieren utilizar simplemente $\lambda=1$ y el siguiente procedimiento:

(i) Aplicar un patrón de entrada y calcular la salida O.

(ii)

a.- Si la salida es correcta volver a (i).

b.- Si la salida es incorrecta, y es cero, sumar cada entrada a su peso correspondiente.

c.- Si la salida es incorrecta, y es uno, restar cada entrada a su peso correspondiente.

(iii) Volver a (i).

- **Entrenamiento**

Las células asociativas (elementos A) son inicializadas con pesos fijos, cuyos valores pueden ser +1 ó -1. Su conexión con las células de respuesta (elementos R) es en forma aleatoria y dependerá del diseñador.

El proceso de aprendizaje es supervisado en los elementos R y puede tomar diferentes formas. La red puede ser utilizada tanto para casos discretos binarios como para continuos, pero en este último no se entrará en detalle debido a que existe otra red similar que opera en mejor forma y que se presentará a continuación.

El proceso de entrenamiento es prácticamente igual que en el caso del Adaline, pero utiliza el algoritmo de aprendizaje presentado recién.

2.3.4.- Retropropagación (Backpropagation)

- **Descripción**

Una red de perceptrones es capaz de entrenar sus unidades de salida para aprender a clasificar los patrones de entrada linealmente separables. Para el caso de patrones no linealmente separables el proceso se hace más complejo y sólo puede ser solucionado con redes multicapas. Sin embargo, si la salida posee error, existirá el problema de cómo determinar cual elemento de proceso o interconexión ajustar. Una red de retropropagación es capaz de resolver esto asignando a todos los elementos de proceso una cierta "responsabilidad" por la respuesta errónea. Esto se traduce en un error que es propagado desde la capa de salida a las neuronas de las capas previas de modo que éstas adapten sus pesos para minimizar dicho error y alcanzar el valor deseado.

Técnicamente, retropropagación es una ley de aprendizaje específica. Este término es usualmente usado para referirse a una arquitectura de red jerárquica que usa el algoritmo de retropropagación para ajustar los pesos de interconexión de cada neurona de la red, basados en el error presente en la salida.

Esta red es una extensión de la estructura del Madaline con la diferencia de que a la salida de cada neurona existe una función no lineal (ver figura 11) y todos sus pesos son adaptables. Dicha función no lineal debe ser continua, diferenciable, monotónicamente creciente y asintótica a un valor. En este trabajo utilizaremos específicamente dos tipos de estas funciones: la función sigmoidal y la tangente hiperbólica.

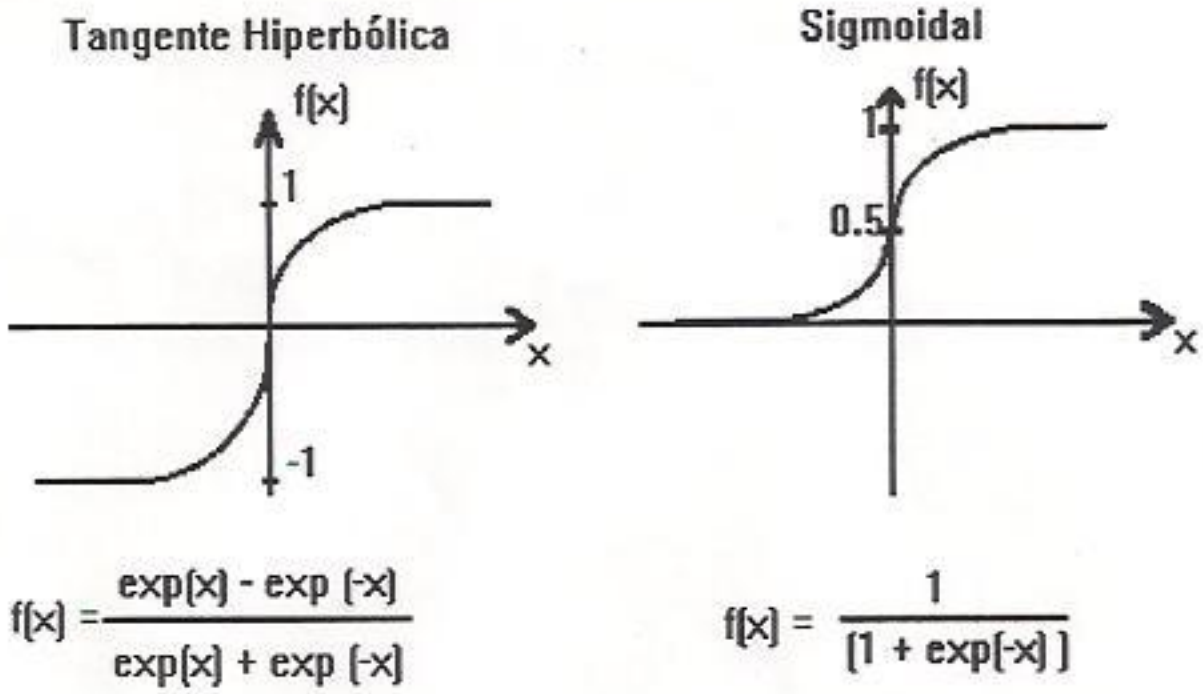

Figura 11: Funciones de activación usadas en redes retropropagación

La estructura de esta red (ver figura 12) consta siempre de una capa de entrada, una de salida y al menos una capa oculta. Cada capa está totalmente conectada a la siguiente y no existe interconexión entre las neuronas de una misma capa. Los patrones de entrada y salida no tienen que ser necesariamente valores binarios, éstos pueden tomar cualquier valor que sea capaz de representar la red. Muchas veces, cuando se utiliza una de las funciones de activación previamente mencionadas, la salida es generalmente escalada a los valores máximos y mínimos que logran dichas funciones.

Teóricamente no está limitado el número de capas ocultas, pero típicamente no son más de dos. Existen trabajos en que se han utilizado hasta tres capas ocultas, de modo de solucionar problemas complejos de clasificación de patrones. El número de neuronas en esta capa dependerá del diseñador de la red. Demasiadas neuronas en las capas ocultas harán que la red tenga dificultades al procesar nuevos tipos de patrones de entrada, es decir, hará generalizaciones. Pocas neuronas no permitirán a la red hacer suficientes representaciones internas de modo de generar sus mapas de entrada y salida.

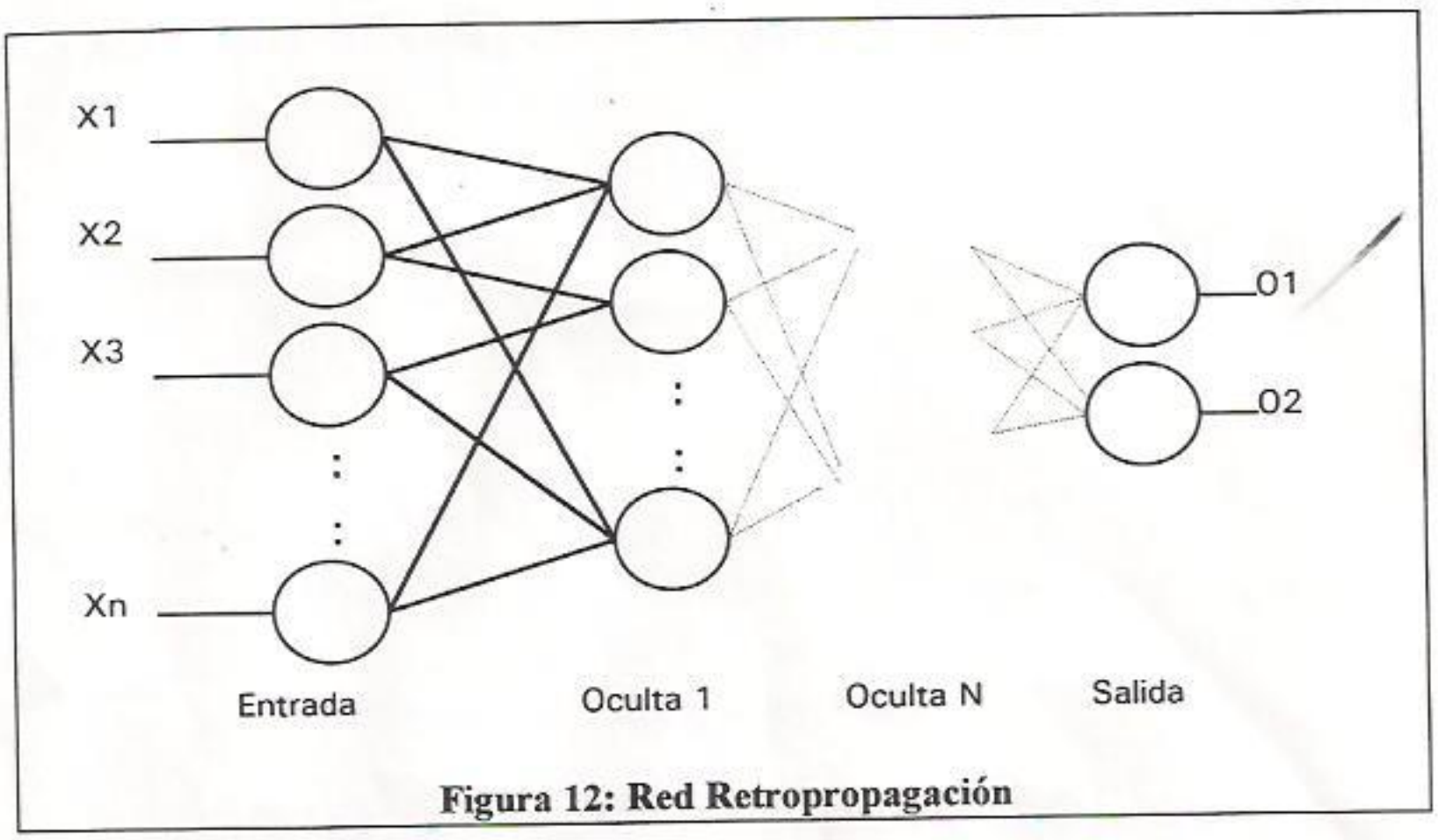

Figura 12: Red Retropropagación

Todas las neuronas dentro de la red son idénticas al Adaline (figura 7) pero, como se mencionó anteriormente, con una función no lineal a la salida .

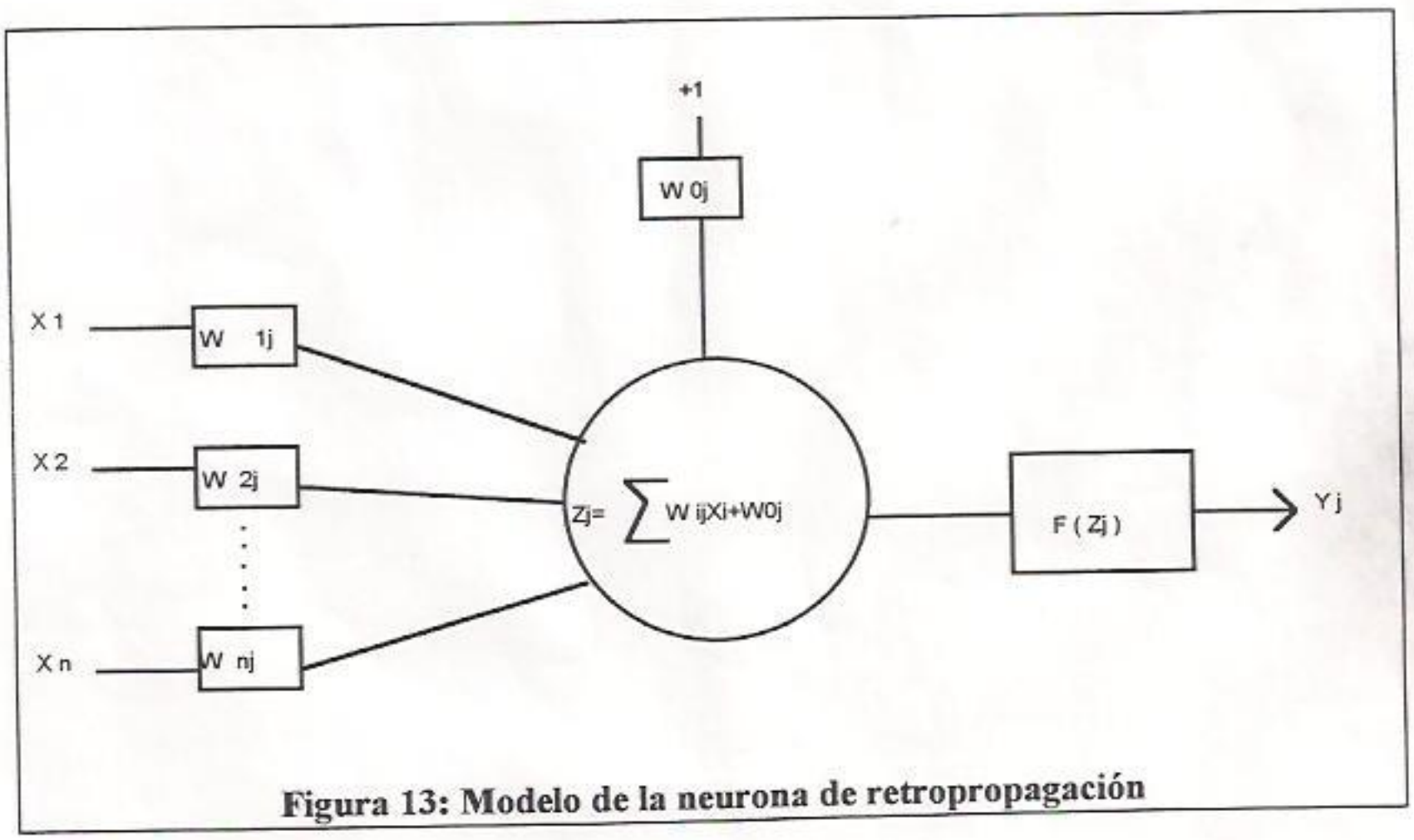

Figura 13: Modelo de la neurona de retropropagación

- **Algoritmo de aprendizaje**

Como se mencionó anteriormente, el proceso de ajuste de los pesos se lleva a cabo primero en la capa de salida y posteriormente en las capas internas o escondidas, procediendo desde la más cercana a la salida a la más cercana a la entrada. El ajuste se realiza en base al algoritmo del descenso del gradiente o regla delta generalizada [Apéndice 3].

Ajuste de los pesos de la capa de salida

Se considera a continuación el ajuste de un solo peso desde la neurona 'p' en la última capa escondida j a la neurona q en la capa de salida, como se muestra a continuación en la figura 14:

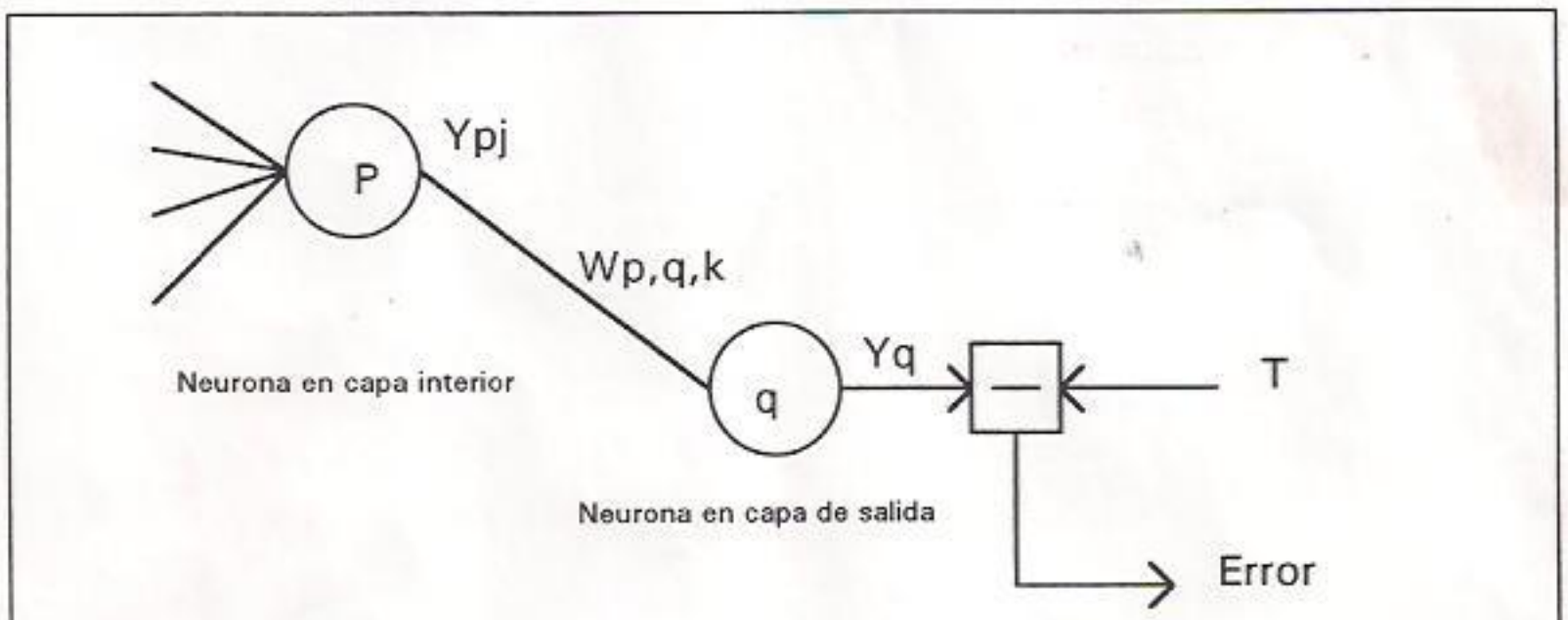

Figura 14: Ajuste pesos de la capa de salida

(i) La salida de la neurona q se resta de la salida deseada correspondiente, generando una señal de error que se multiplica por la derivada de la función no lineal evaluada para la neurona q.

$$\delta = f'(x_q)\,(T - Y_q) \qquad (2.14)$$

donde x_q es la entrada a la red.

(ii) Luego δ se multiplica por la salida de la neurona p y este producto se multiplica por un coeficiente de entrenamiento η (típicamente entre 0.01 y 1.0), y el resultado se suma al peso.

De otra forma, teniendo en cuenta el tiempo discreto n, n={0,1,2,...} se tiene

$$W_{p\,q,\,k}\,(n+1) = W_{p\,q,\,k}\,(n) + \eta\,\delta\,Y_{p,\,j} \qquad (2.15)$$

Ajuste de los pesos de las capas internas

Puesto que no existen vectores de salida deseados para las neuronas de las capas internas no es posible aplicar el algoritmo anterior.

Las capas internas se entrenan propagando el error hacia atrás, a través de la red, capa por capa ajustando los pesos de cada una de ellas. Se utilizan las mismas ecuaciones anteriores, calculando δ de otra forma. Se considera la figura 15:

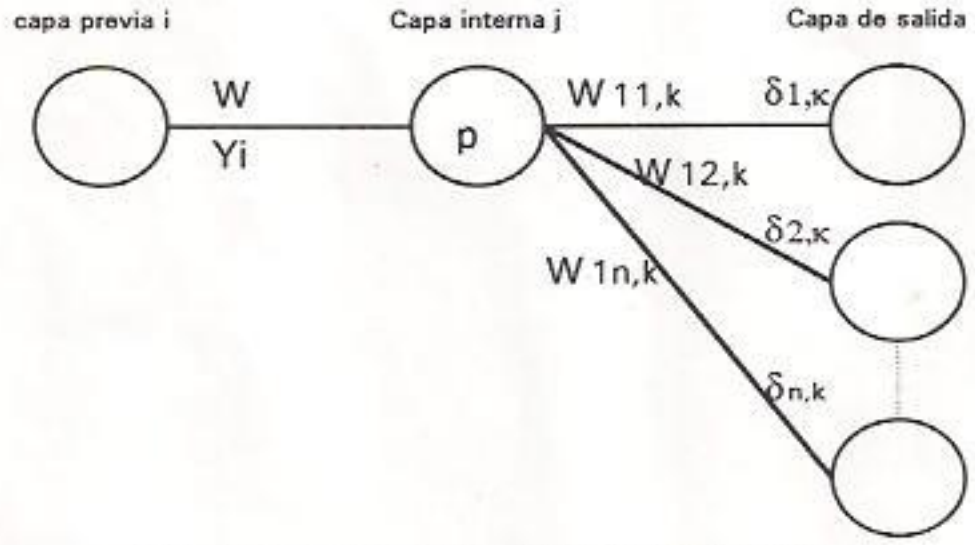

Figura 15: Ajuste pesos capas internas

El valor de δ correspondiente a la neurona p de la capa interna j se calcula de la siguiente forma:

$$\delta_{p,j} = f'(x_{p,j}) \sum_{q} \delta_{q,k}\, W_{pq,\,k} \qquad (2.16)$$

Con este valor de δ es posible aplicar las ecuaciones del entrenamiento de la capa de salida para las capas internas o escondidas.

- **Entrenamiento**

Lo esencial de este procedimiento es que los términos de error requeridos para adaptar los pesos son retropropagados desde los nodos en la capa de salida hacia los nodos de las capas internas, de ahí que se le conozca como retropropagación.

La aplicación de la regla delta implica dos fases:

(i) Durante la primera fase la entrada es presentada y propagada hacia adelante a través de la red para computar el valor de salida O para cada unidad. Esta salida es luego comparada con el objetivo, resultando en una señal de error δ para cada unidad.

(ii) La segunda fase involucra un paso hacia atrás en la red, similar al paso hacia adelante, durante el cual la señal de error es entregada a cada unidad en la red realizándose los cambios apropiados en los pesos.

Métodos para acelerar el entrenamiento

(i) Método del momentum: consiste en sumar, al ajuste de los pesos, un término proporcional a la cantidad del cambio producida en el ajuste previo del misno.

$$\Delta W_{pq,k}(n+1) = \alpha \Delta W_{pq,k}(n) + \eta \delta Y_{p,j} \qquad (2.17)$$

$$W_{p\,q,\,k}(n+1) = W_{p\,q,\,k}(n) + \Delta W_{pq,k}(n+1) \qquad (2.18)$$

Un valor típico de α, llamado coeficiente de momentum, se ajusta comúnmente en 0.9. El factor η fue definido en (2.15).

(ii) Método del suavizamiento: similar al anterior, pero basado en un suavizamiento exponencial que ha demostrado ser superior en algunas aplicaciones.

$$\Delta W_{pq,k}(n+1) = \alpha \Delta W_{pq,k}(n) + (1 - \alpha) \delta Y_{p,j} \qquad (2.19)$$

$$W_{p\,q,\,k}(n+1) = W_{p\,q,\,k}(n) + \eta \Delta W_{pq,k}(n+1) \qquad (2.20)$$

α es el coeficiente de suavizamiento en el rango de 0.0 a 1.0. Si α es cero, el suavizamiento es mínimo; el ajuste de los pesos corresponde al nuevo valor calculado. Si α es 1.0 el nuevo ajuste se ignora, repitiéndose el ajuste previo.

El factor η nuevamente corresponde al coeficiente de entrenamiento, que ajusta el tamaño del cambio promedio de los pesos.

(iii) Método estocástico: al igual que en el caso de la red Adaline, el método resulta ser aplicable fácilmente en este tipo de red y con excelentes resultados.

$$W_{p\,q,\,k}(n+1) = W_{p\,q,\,k}(n) + \eta \delta Y_{p,\,j} / \| Y_j \|^2 \qquad (2.21)$$

donde δ, al igual que en los casos anteriores queda determinado por:

$\delta = f'(x_q)(T - Y_q)$ para el caso de las neuronas de la salida.

$\delta_{p,j} = f'(x_{p,j}) \sum \delta_{q,k} W_{pq,\,k}$ para las neuronas ocultas.

2.4.- Redes con aprendizaje no supervisado y evocación hacia adelante.

2.4.1.-Propagación por conteo hacia adelante (Counterpropagation)

- **Descripción**

Esta red basa su funcionamiento en la habilidad de aprender un mapa matemático por adaptación, en respuesta a ejemplos de la función de mapa, definida como:

$$\phi : IR^n \rightarrow IR^n \qquad \text{donde } Y = \phi(X)$$

De esta función se genera un conjunto de ejemplos (X1,Y1), (X2,Y2),..., donde X es una variable independiente escogida de acuerdo a la densidad fija de probabilidad ρ . Con estos ejemplos, se define estadísticamente la relación deseada de entrada/salida, aprendiendo a través de ellos el mapeo matemático de la función.

Estas redes que siguen el procedimiento anterior son llamadas redes neuronales de mapeo, y dentro de esta clasificación se encuentra la red Propagación por Conteo [4] [5]. Esta arquitectura propuesta por Robert Hecht-Nielsen (1987), tiene capacidades que la convierten en una red muy atractiva para una amplia gama de aplicaciones.

Su arquitectura está formada por tres capas:

(i) Una capa de entrada, que contiene n unidades de dispersión que simplemente multiplexan las señales de entrada X1, X2,...,Xn.

(ii) Una capa intermedia llamada de Kohonen con N elementos de proceso que tienen señales de salida Z1, Z2, ..., ZN.

(iii) Una capa de salida llamada de Grossberg con m elementos de proceso que tienen señales Y'1, Y'2, ..., Y'm.

Las salidas de la capa de Grossberg corresponden a las más importantes, debido a que representan aproximaciones a las componentes Y1, Y2, ..., Ym de $Y = \phi(X)$. Durante el entrenamiento, estas componentes "correctas" son entregadas a las unidades de Grossberg y a las unidades de entrada de la capa de entrada (ver figura 16).

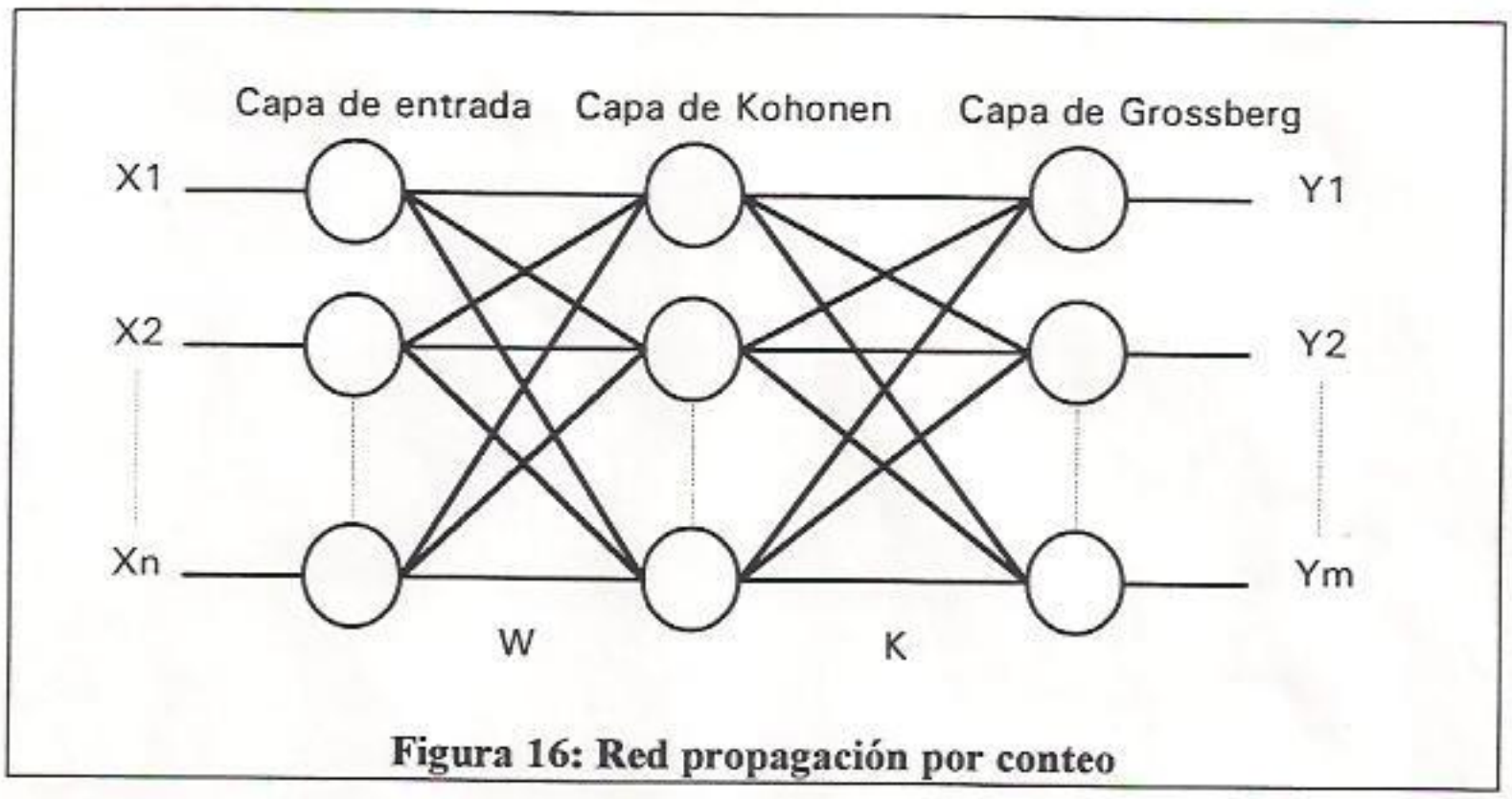

Figura 16: Red propagación por conteo

- **Entrenamientos y algoritmos de aprendizaje**

Inicialización de los pesos

Los pesos de las capas de Kohonen y Grossberg inicialmente son asignados en forma aleatoria con valores convenientemente entre 0 y 1. Es recomendable que los pesos de la capa de Kohonen, al igual que el vector de entrada, sean normalizados, de modo que su producto entregue un valor de la similitud, medida en términos de los cosenos direccionales [Apéndice 4].

El entrenamiento de la red se realiza en dos fases y por dos algoritmos diferentes.

Fase 1 del entrenamiento

Durante esta fase se aplica una secuencia de vectores de entrenamiento X y un algoritmo conocido como Aprendizaje de Kohonen, para ajustar los pesos Wi. El aprendizaje de Kohonen es un algoritmo auto-organizativo que opera en modo no supervisado, esto es, sólo vectores de entrada (sin el vector de salida objetivo) son aplicados durante el entrenamiento y los pesos W son ajustados de manera tal que una neurona o grupo de ellas esté activa en la segunda capa. Lo que interesa de esta fase del entrenamiento es que se asegure la separación de los vectores disímiles.

El algoritmo de aprendizaje utilizado para el ajuste de los pesos queda determinado por:

(1) Aplicar un vector de entrada X.

(2) Encontrar el producto punto del vector X con cada conjunto de pesos conectados a cada neurona. Si Wi es el conjunto de pesos asociados con la neurona i de la capa 2, entonces para cada i, (i=1,...N) calcular el producto punto X·Wi.

(3) Encontrar la neurona con el producto punto más alto y rotular esta neurona con la letra C.

(4) Ajustar el vector de pesos asociado con la neurona C de acuerdo a la siguiente fórmula:

$$Wc\,(t+1) = Wc(t) + \alpha\,(X - Wc(t))\,Zc \qquad (2.22)$$

$$Zc = \begin{cases} 1, & \text{si c es el menor índice para el cual se cumple:} \\ & \|Wc(t) - X\| \le \|\,Wj(t) - X\|\ \text{, para todo j.} \\ 0, & \text{en otro caso} \end{cases} \qquad (2.23)$$

(5) Repetir los pasos 1 al 4 tantas veces como sea necesario para entrenar el sistema.

Al comienzo del entrenamiento, usualmente α parte con un valor entre 0.5 y 0.8. Al progresar el entrenamiento, este valor decrece a 0.1 o menos.

Una vez que la capa de Kohonen se ha estabilizado (en otras palabras, los vectores se han congelado, después de alcanzar la equiprobabilidad), la capa de Grossberg comienza a aprender las salidas correctas para cada vector Wi de la capa de Kohonen. Esto se realiza en la segunda fase del entrenamiento.

Fase 2 del entrenamiento

Durante esta fase, se entrenan los pesos que conectan las capas 2 y 3 mediante el aprendizaje "outstar" de Grossberg. Para esto, se entrega un vector de entrada X y un vector de salida Y (objetivo), por lo que este método corresponde a un entrenamiento supervisado. El ajuste de los vectores Ki se realiza de la siguiente forma:

(1) Entrar un vector X y un correspondiente vector objetivo Y.

(2) Calcular el producto punto de cada vector de pesos Wi con el vector X. Encontrar el mayor de estos productos y rotular la neurona que corresponda en la capa de Kohonen con la letra C. Colocar su señal de salida en uno y la señal de salida de las otras neurona de esta capa en cero.

(3) Ajustar cada uno de los pesos Kcj, entre la neurona C de la capa de Kohonen y todas las neuronas de la capa de Grossberg, de acuerdo a la siguiente regla:

$$K_{cj}(t+1) = K_{cj}(t) + (d\,Y_j - c\,K_{cj}(t))\cdot Zc \quad , j=1,\ldots,m \qquad (2.24)$$

donde c y d son constantes menores que 1 y que son reducidas durante el entrenamiento y Z es el nivel de señal de salida de la unidad C.

(4) Repetir los pasos (1) al (4) tantas veces como sea necesario para entrenar el sistema.

Operación normal de la red

Una vez que el entrenamiento ha terminado se establece el modo normal de operación.

En este modo los vectores de entrada son presentados a la primera capa, para luego ser multiplicados por los pesos de las neuronas de Kohonen. Una competencia se lleva a cabo, ganando aquellas unidades cuyos pesos Wi estén más cerca del vector de entrada X.

El valor de la salida de la neurona i de la capa de Kohonen, queda determinada por:

$$Z_i = \sum_{j=0}^{n} W_{ij} \cdot X_j \qquad (2.25)$$

La competencia puede establecer una única unidad ganadora o a un grupo de ellas. Las unidades ganadoras son puestas en uno y las no vencedoras en cero. La capa de Grossberg se encargará de dar la salida deseada para el vector X de entrada dada por:

$$Y_m = \sum_{i=0}^{N} K_{im} \cdot Z_i \qquad (2.26)$$

2.5.- Redes con aprendizaje no supervisado y evocación retroalimentada.

2.5.1 Hopfield

- **Descripción**

La red de Hopfield es normalmente usada con entradas binarias. Esta red es la más apropiada cuando las representaciones binarias exactas son posibles, como en las imágenes en blanco y negro donde los pixeles son los valores de entrada. Esta red es menos apropiada cuando los valores de entrada son continuos, debido a que el problema fundamental de representación debe ser direccionado para convertir las cantidades análogas en valores binarios.

Una versión de la red original [5] será la que se describirá a continuación. Esta red, mostrada en la figura 17, posee N nodos que contienen funciones de activación del tipo signo con entradas y salidas que toman valores +1 ó -1. La salida de cada neurona es realimentada a todos los otros nodos, excepto a sí misma, con pesos denotados por Wij.

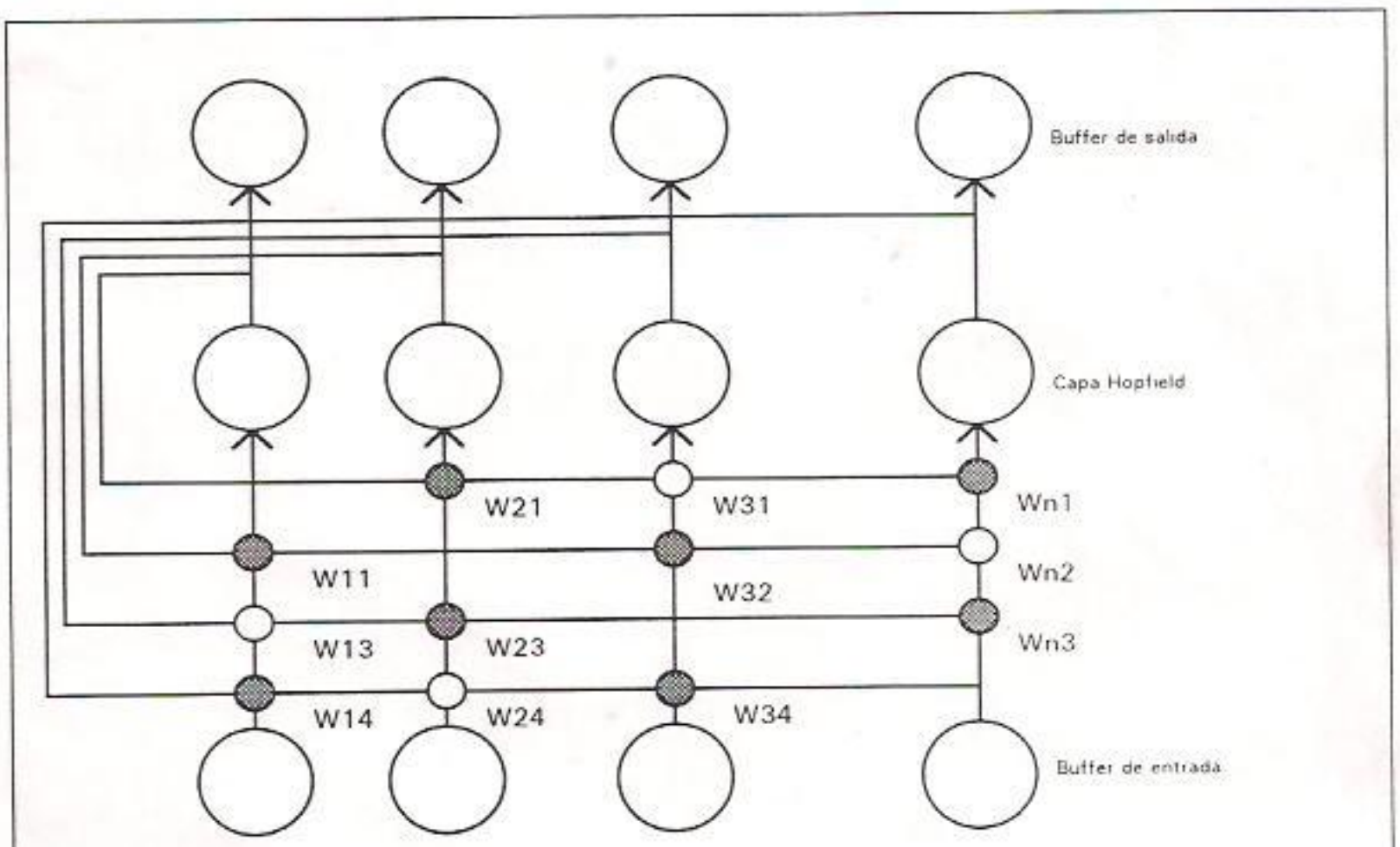

Figura 17: Red de Hopfield

Las neuronas dentro de la red están basadas en el esquema de la Adaline, con la diferencia de que utiliza en su salida una función del tipo signo en vez de una lineal. La figura 18 presenta un elemento de proceso de la red de Hopfield:

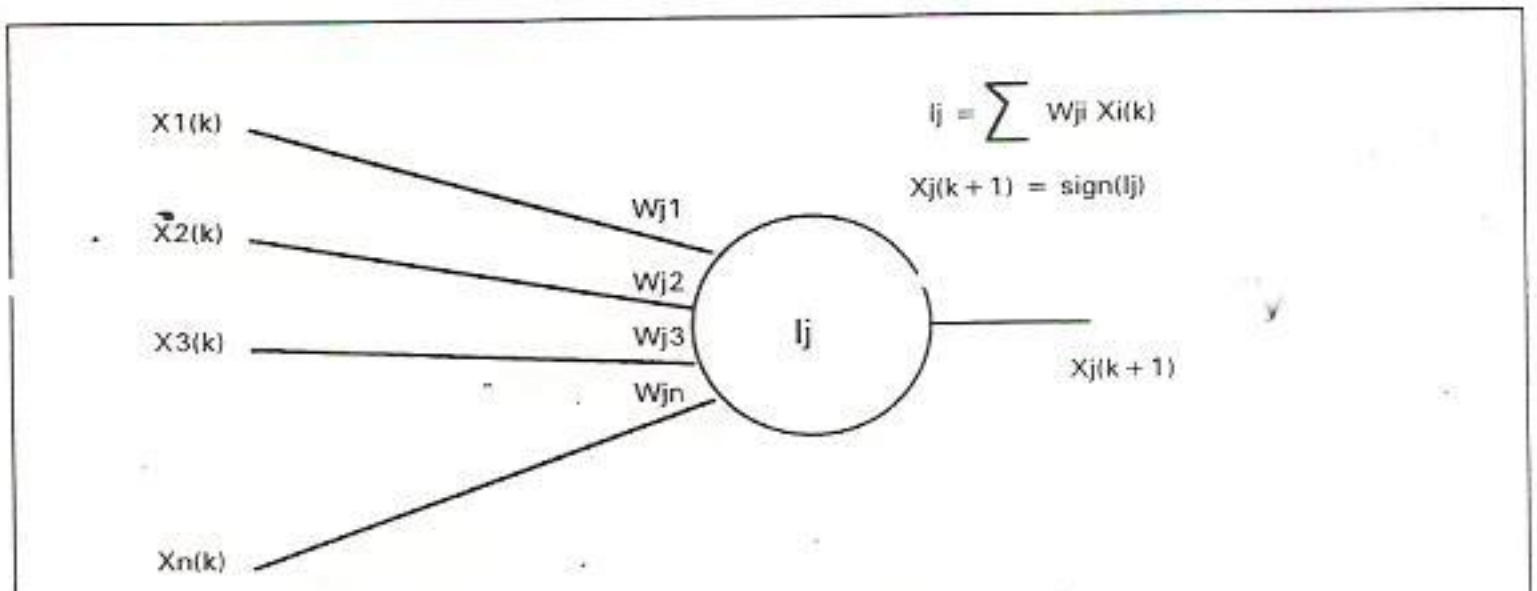

Figura 18: elemento de proceso de una red de Hopfield

- **Entrenamiento y algoritmo de aprendizaje**

Básicamente, la asignación de los pesos corresponde al aprendizaje de la red [5]. Estos son asignados con valores que dependerán de las clases de los patrones de entrada.

Si Wij son los pesos de conexión del nodo i al nodo j, y si Xi,s es el elemento i del patrón de entrada de la clase s, entonces

$$W_{ij} = \begin{cases} \sum_{s=0}^{m-1} X_{i,s}X_{j,s} & , \ i \neq j \\ 0 & , \ i=j,\ 0 \leq i,\ j \leq m-1 \end{cases} \qquad (2.27)$$

donde m es el número total de clases de los patrones.

A pesar de la simplicidad del algoritmo, éste presenta dos graves limitaciones cuando es usado como una memoria asociativa. La primera limitación es en cuanto al número de patrones que pueden ser almacenados en forma precisa. Si demasiados patrones son almacenados, la red puede converger a valores falsos. Hopfield demostró [5] que el número apropiado de patrones que es capaz de almacenar tiene que ser menor que 0.15 N, donde N es el número de nodos de la red. La segunda limitación se refiere a que un ejemplar puede hacer inestable la red si su forma contiene demasiados bits en común con otros patrones. Se dice que el sistema es inestable cuando el patrón de salida converge a algún otro ejemplar que no es el correcto

- **Operación de la red**

La operación de esta red es distinta a las anteriores. Esta consiste básicamente en una iteración que consta de dos pasos:

Paso 1

Cuando un patrón es impuesto a la red es forzado para que aparezca en la salida. Esto se podría describir por

$$\mu i\,(0) = Xi \;,\; 0 \le i \le N\text{-}1 \qquad (2.28)$$

En esta fórmula $\mu i(0)$ es la salida del nodo i al tiempo 0 y Xi es el elemento i del patrón de entrada, el cual puede ser +1 ó -1.

Paso 2

Este paso consiste en una iteración, en tiempo discreto, que se detiene cuando la salida permanece estable ante nuevas repeticiones. Dichas iteraciones se realizan en base a la recursión

$$\mu_i\,(t+1) = f_H\left(\sum_{i=0}^{N-1} W_{ij}\,\mu_i\,(t)\right) \qquad (2.29)$$

donde fH es la función no lineal del tipo signo.

2.5.2.- Hamming

- **Descripción**

La red de Hamming, usada como red neuronal, es un clasificador de mínimo error para vectores binarios, donde el error es definido usando la distancia de Hamming. En un clasificador de error mínimo, las clases son definidas por el significado de los vectores de ejemplo; los vectores de entrada son asignados a la clase para la cual la distancia entre el vector de ejemplo y el vector de entrada son mínimos.

La distancia de Hamming es la vía de medida de la distancia entre dos vectores binarios y es definida como el número de bits en un vector de entrada que no sean los mismos que los bits de uno de los vectores de ejemplo. Consideremos, por ejemplo, el siguiente conjunto de vectores con que la red ha sido entrenada:

C1 = [1 1 1 1 1 1]
C2 = [1 1 1 -1 -1 -1]
C3 = [-1 -1 -1 1 1 1]
C4 = [-1 -1 -1 -1 -1 -1]

y sea el vector de entrada

Cx = [1 -1 1 -1 -1 1]

Este vector de entrada Cx es comparado con cada uno de los vectores Cn, con n = 1, 2, 3, 4 , resultando ser C2 el vector con la menor distancia según Hamming (posee sólo dos bits distintos).

Una diferencia importante que posee esta red con la de Hopfield, mencionada en [5] es que, desde el punto de vista de la precisión y capacidad, la red de Hamming clasifica en mejor forma los vectores de entrada cuando los errores en los bits son independientes y aleatorios.

Por otro lado, la capacidad de almacenamiento que posee la red de Hamming es notablemente mayor. Por ejemplo: una red de Hopfield con 100 nodos puede almacenar en forma óptima alrededor de 10 patrones de entrada y, en cambio, una red de Hamming puede almacenar cerca de 62 patrones bajo las mismas condiciones.

Estructuralmente, como se muestra en la figura 19, la red de Hamming está compuesta por dos redes: una primera que calcula las N distancias mínimas de Hamming para los M patrones de ejemplo, y una segunda red que selecciona el nodo con salida máxima.

La primera red está compuesta por dos capas: la primera es una capa de entrada que distribuye todas las entradas a la capa siguiente, llamada capa de formas. Esta última está formada por neuronas del tipo Adaline más una función del tipo signo en su salida. Las entradas a la red son del tipo binario, tomando los valores +1 ó -1. La segunda red tiene la misma estructura que una red de Hopfield pero la asignación de los pesos es distinta y sus entradas no son binarias.

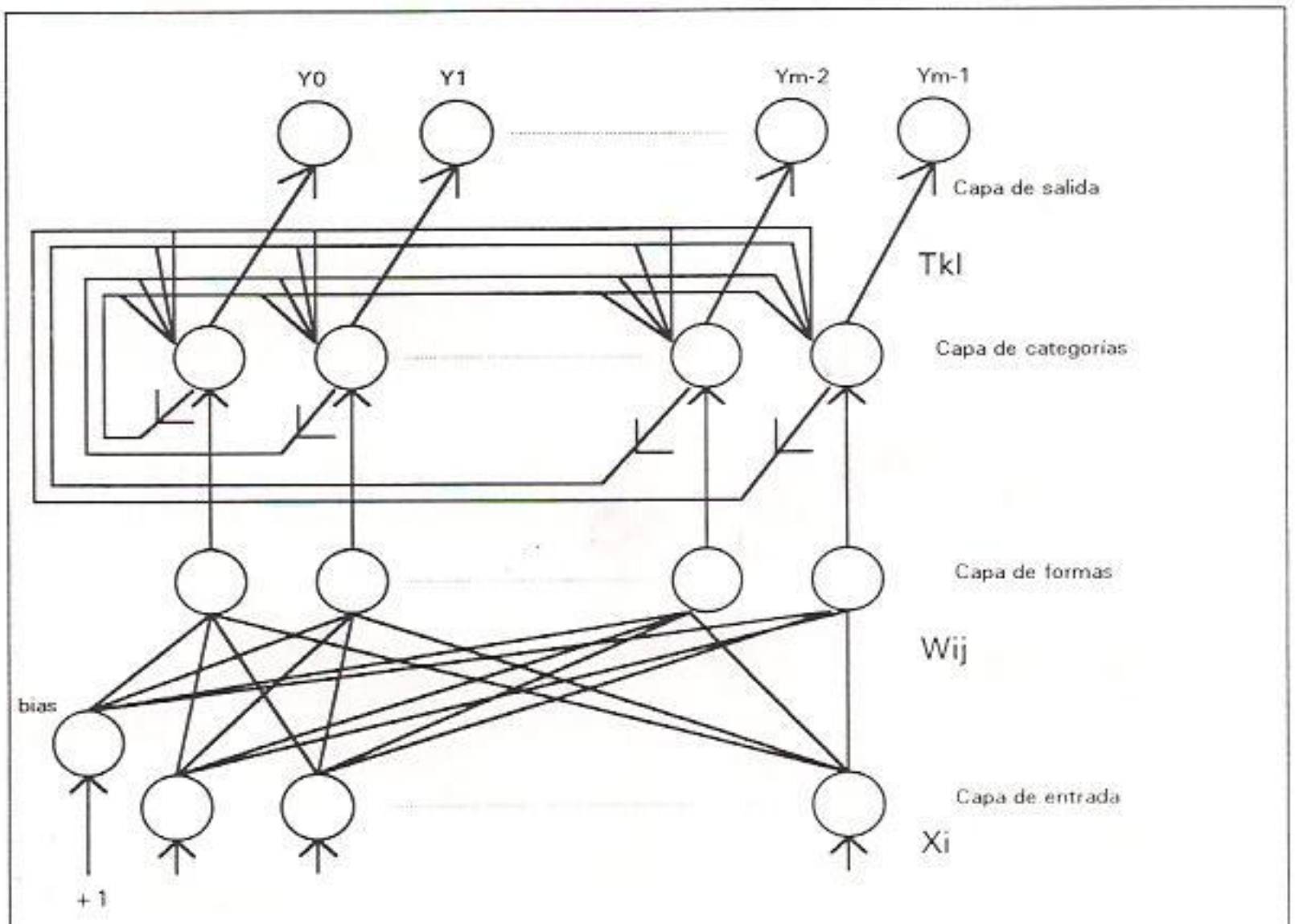

Figura 19: Red de Hamming

- **Entrenamiento y algoritmo de aprendizaje**

Al igual que en la red de Hopfield, el aprendizaje no es realizado mediante sucesivas adaptaciones de sus pesos, sino que por una asignación dependiente de los patrones de ejemplo.

Si los pesos de la capa de formas son denotados por Wij, y los patrones de ejemplo de la clase s por X^s, entonces la asignación de los pesos estaría dada por:

$$Wij = \begin{cases} 0.5 \cdot X_{ij} , & i=1,...,N \; , \; j= 1,...,M \\ 0.5 \cdot N , & i=1,...,N \; , \; j= 0 \end{cases} \qquad (2.30)$$

Para el caso de la capa de categorías, si se denota por t_{kl} los pesos se tiene que la asignación queda determinada por:

$$t_{kl} = \begin{cases} 1 , & k = l \\ -\varepsilon , & k \neq l, \;\; \varepsilon < 1/M \end{cases} \qquad (2.31)$$

- **Operación de la red**

Básicamente, la operación de la red consta de dos pasos:

Paso 1

Consiste en que un patrón desconocido es presentado en la entrada y procesado por las neuronas de la capa de forma entregando como resultado:

$$\mu_j (0) = f_T \left(\sum_{i=1}^{N-1} W_{ij} X_i + 0.5 \cdot N \right) , \quad 0 \leq j \leq M-1 \qquad (2.32)$$

En esta fórmula $\mu i(0)$ es la salida del nodo i al tiempo 0, Xi es el elemento i del patrón de entrada, el cual puede ser +1 ó -1 y Wij es el peso de conexión entre la capa de entrada y la capa de forma. f_T es una función de tipo no lineal definida como:

$$f_T(x) = \begin{cases} x, & \text{si } x > 0 \\ \\ 0, & \text{si } x \leq 0 \end{cases} \qquad (2.33)$$

Como queda demostrado en el Apéndice 5, el valor de $\mu i(0)$ corresponderá a la cantidad de bits en común que posea el vector de entrada con el patrón de la clase j.

Paso 2

Este paso es muy similar al caso de la red de Hopfield, e itera hasta que la red converge a un valor. Esta iteración queda determinada por la recurrencia:

$$\mu_j(t+1) = f_T \left(\mu_j(t) - \varepsilon \sum_{k \neq j} \mu_k(t) \right), \quad 0 \leq j, \quad k \leq M-1 \qquad (2.34)$$

Una diferencia clara que se aprecia con la red de Hopfield es que las entradas a esta segunda red ya no corresponden a entradas binarias.

2.5.3.- Teoría de la Resonancia Adaptiva Binaria (ART1)

- **Descripción**

La red ART1, introducida por Carpenter y Grossberg [5], consiste en dos capas: F1 y F2 (figura 20), clasificadoras de cercanía, las que almacenan un número arbitrario de patrones espaciales binarios $Ak=\{a_1^k,a_2^k,...,a_n^k\}$, k = 1, 2,..., m, usando un aprendizaje competitivo en línea.

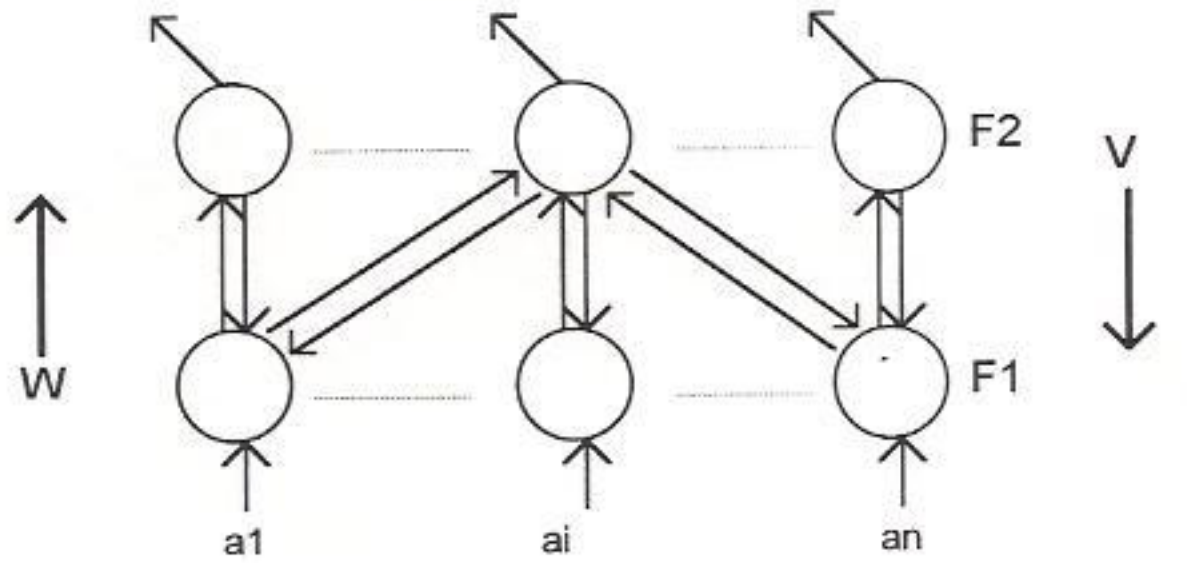

Figura 20: ART1

La capa F1 está constituida por un número de neuronas igual al de componentes del patrón de entrada y está totalmente interconectada con la capa F2 por medio de dos tipos de conexiones: las conexiones que conducen de F1 a F2 denominadas W_{ij}, y las conexiones que conducen de F2 a F1 denominadas V_{ij}. La función de umbral utilizada por las neuronas de F2 corresponde a una sigmoidal.

Las componentes de las neuronas obedecen a ecuaciones diferenciales (ecuaciones de membrana) no lineales que describen las conductancias, permeabilidades etc., de las membranas neuronales biológicas. Para el caso de la transmisión sináptica, el estado de la sinapsis puede ser influenciada solamente por el estado de las componentes que estén en contacto fisico con ella.

- **Operación de la red**

Como esta red opera en línea, la descripción de su entrenamiento y aprendizaje quedará determinada por la operación de la red. El método que se describirá a continuación corresponde al método rápido de aprendizaje y consiste, básicamente, en los siguientes pasos:

Paso 1. Inicialización de los pesos.

$$W_{ij}(0) = 1 / (1+N) \tag{2.35}$$

$$V_{ij}(0) = 1 \quad , \; 0 \le i \le N\text{-}1, \quad 0 \le j \le M\text{-}1 \tag{2.36}$$

$$\text{Asignar } \rho, \quad 0 \le \rho \le 1 \tag{2.37}$$

donde N es el número de entradas de la capa F1 y M es el número de elementos en la capa F2 . W_{ij} son las conexiones de F1 a F2 y V_{ij} de F2 a F1, entre el nodo de entrada i y el nodo de salida j al tiempo 0. El valor de ρ es el umbral de vigilancia que indica qué tan próxima una entrada debe ser para ser almacenada.

Paso 2. Aplicar una nueva entrada.

Se presenta un patrón de entrada $Ak=\{a_1^k, a_2^k, \ldots, a_n^k\}$, k = 1, 2,..., m a F1, quedando su salida puesta con los valores de la entrada:

$$F1 = \{ a_1^k, a_2^k, \ldots, a_n^k \} \tag{2.38}$$

Paso 3. Cómputo de las salidas de F2.

Cada elemento de proceso de F1 activado con un valor a_i, envía dicha señal a través de las conexiones W_{ij} a F2, la que computa su salida dada por:

$$\mu_j = \sum_{i=0}^{N-1} W_{ij}(t) \cdot X_i(t) \tag{2.39}$$

$X_i(t)$ es el elemento i de la entrada, el cual puede tomar los valores 0 ó 1.

Paso 4. Seleccionar el ganador de la competencia.

Cada elemento de proceso de F2 compite con los otros quedando sólo un elemento activo, seleccionado según el criterio:

$$\mu_j^* = \max_j \{ \mu_j \} \qquad (2.40)$$

En caso de empate, se resuelve en favor del elemento de proceso con menor índice.

Paso 5. Propagación hacia atrás.

El elemento ganador en F2 propaga su señal hacia F1 por medio de las conexiones V_{ij}, generando un nuevo conjunto de activaciones en F1, al que llamaremos:

$$F1' = \{ a_1', a_2', \ldots, a_n' \} \qquad (2.41)$$

donde

$$a_i' = \mu_j^* \cdot V_{ij} \qquad (2.42)$$

Este paso no es necesario de realizar, pero sirve para aclarar el siguiente paso.

Paso 6. Prueba de cercanía.

La entrada $Ak=\{a_1^k, a_2^k, \ldots, a_n^k\}$ es comparada con el vector generado por el elemento ganador en F2, $F1' = \{ a_1', a_2', \ldots, a_n' \}$, por medio del siguiente criterio:

$$\| X \| = \sum_{i=0}^{N-1} X_i(t) \qquad (2.43)$$

$$\| V \cdot X \| = \sum_{i=0}^{N-1} V_{ij}(t) \cdot X_i(t) \qquad (2.44)$$

¿ es $\| V \cdot X \| / \| X \| > \rho$?

(i) Si la diferencia relativa entre estas dos activaciones excede el valor de vigilancia ρ (si es verdadera la interrogación anterior), entonces el elemento de proceso ganador en F2 representa el patrón de la clase A_k. Los pesos del elemento ganador son adaptados según el criterio siguiente:

$$V_{ij}(t+1) = V_{ij}(t) \cdot X_i(t) \qquad (2.45)$$

$$W_{ij}(t+1) = V_{ij}(t) \cdot X_i(t) \,/\, \left[0.5 + \sum_{i=0}^{N-1} V_{ij}(t) \cdot X_i(t)\right] \qquad (2.46)$$

(ii) En caso contrario, si la diferencia entre las activaciones es menor que el valor de vigilancia, el elemento ganador no representa la clase ganadora, por lo que dicho elemento es deshabilitado (forzado a ser 0) y se vuelve al paso 3, determinando un nuevo ganador.

Paso 7. Término del aprendizaje del patrón.

En este paso, el aprendizaje del patrón ha concluido y la red está lista para recibir al siguiente. Previo a esto, se debe habilitar todos los nodos que fueron deshabilitados en el paso anterior.