UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

— ARAGON —

Sistemas NEURO - FUZZY

T E S I S

QUE PARA OBTENER EL TITULO DE

Ingeniero en Computación

PRESENTAN:

Alfonso Magaña Ramírez

José Luis Ortega Vallejo

Asesor: M. en I. DAVID GONZALEZ MAXINES

SAN JUAN DE ARAGON, MEXICO 1 9 9 6

TESIS CON
FALLA DE ORIGEN

Con mucho cariño a mi madre ELOISA, por haberme dado todo su apoyo en cada momento de mi vida y por sobre todo en los momentos más difíciles de mi carrera.

A mi esposa ISABEL, por ser en todo momento mi ayuda y mi continuo estímulo a la superación.

A mi hijo ALFONSO, por ser mi aliciente final para la terminación del presente trabajo.

ALFONSO

Con mucho cariño para mis padres ROSA MARIA Y TOMAS, que tanto me apoyaron para la terminación de mi carrera.

También a mis hermanas por su apoyo.

JOSE LUIS

A nuestro profesor, M. en I. DAVID GONZALEZ MAXINES por el asesoramiento, paciencia y recomendaciones en el desarrollo de la presente tesis.

INDICE

Introducción 1

1 Conceptos de una Red Neuronal Artificial

Introducción 5
1.1 Terminología 5
1.1.1 Vectores de Entrada y Salida 6
1.1.2 Conexiones 7
1.1.3 Elementos de Proceso 8
1.1.3.1 Combinación Lineal 9
1.1.3.2 Conexiones de Variación Media 10
1.1.3.3 Conexiones Mínimas/Máximas 11
1.2 Funciones de los Elementos de Proceso (PE) 12
1.2.1 Función Lineal 13
1.2.2 Función Escalón 13
1.2.3 Función Rampa 14
1.2.4 Función Sigmoide 15
1.2.5 Función Gausiana 15
1.3 Como Aprenden las Redes Neuronales 16
1.3.1 Aprendizaje Supervisado Versus No Supervisado 16
1.3.2 Aprendizaje Fuera de Linea Versus En Linea 17
1.4 Correlaciones de Hebbian 17
1.4.1 Valores Ilimitados de los PE y Cargas 18
1.4.2 Valores PE Limitados y Cargas Ilimitadas 18
1.4.3 Valores Ilimitados de los PE y Cargas Limitadas 19
1.5 Componente Principal del Aprendizaje 20
1.6 La Diferencial del Aprendizaje Hebbian 21
1.6.1 Diferencial Básica del Aprendizaje Hebbian 21
1.6.2 Aprendizaje Conducido por el Reforzamiento 22
1.6.3 Correlación Covariante 22
1.7 El Aprendizaje Competido 23
1.8 Aprendizaje Máximo-Mínimo 25
1.9 Corrección de Errores en el Aprendizaje 25
1.9.1 Corrección de Errores en el Aprendizaje en dos Capas 25
1.9.2 Corrección de Errores en el Aprendizaje en Múltiples Capas 27
1.10 Refuerzo del Aprendizaje 32
1.11 Aprendizaje Stochastic 33

1.12 Sistemas de Hardwired 34
1.13 Resumen de Procedimientos de Aprendizaje 34
1.14 El Recordar en Redes Neuronales 35
1.14.1 Recuerdo sin Realimentación 36
1.14.2 Recuerdo con Realimentación 37
1.14.3 Respuestas de Interpolación Versus Respuestas Proximas/Contiguas 38
1.15 Sistemas Dinámicos Neuronales 39
1.15.1 Declaración del Espacio Neuronal 40
1.15.2 Señales que Declaran el Espacio como Hipercubico 40
1.15.3 Activaciones Neuronales en el Términos de Memoria Corta 42

2 Arquitectura y Modelos de Redes Neuronales Artificiales

Introducción 43
2.1 Terminología 43
2.1.1 Capas 43
2.1.2 Conexiones Intracapas Versus Intercapas 44
2.1.3. Redes con Realimentación Versus Redes sin Realimentación 44
2.2 Entrada en Estrella, Salida en Estrella y el Adaline 44
2.3 Redes de Capas Simples: Autoasociación, Optimización y Contraste en el Incremento 45
2.3.1 Terminación de un Vector 46
2.3.2 Alejamiento del Ruido 46
2.3.3 Optimización Neuronal 47
2.3.4 Contraste en el Incremento 47
2.4 Redes de Dos Capas: Heteroasociación y Clasificación 48
2.4.1 Diseños Iguales a los que no Tienen Realimentación 49
2.4.2 Diseños Iguales a los Realimentados 49
2.4.3 Diseños de Clasificación sin Realimentación 49
2.5 Redes de Capas Múltiples, Heteroasociación y Funciones de Aproximación 49
2.6 Redes de Conexiones Aleatorias 51
2.7 El Adaline y Madaline 52
2.7.1 El Programa Adaline 59
2.7.2 Ejemplo del Adaline 68
2.7.3 El Programa Madaline 69
2.7.4 Ejemplo del Madaline 78
2.7.5 Otro Ejemplo del Madaline 79

3 Redes Neuronales y Sistemas Fuzzy Logic

Introducción 82
3.1 Estimadores de Funciones Fuzzy y Neuronales 84

3.2 Representación Fuzzy VS Neuronal de la Estructura de Conocimientos 86
3.3 Proyecciones FAMS 88
3.4 La Multiplicación Fuzzy Vector-Matriz: La Composición Máxima-Mínima 89
3.5 Fuzzy FAMS Hebb 91
3.6 El Teorema del FAM Bidireccional para la Codificación de la Correlación-Mínima 93
3.6.1 Codificación Correlación-Producto 95
3.7 Recordar Salidas y la Defuzzificacion 97
3.8 La Arquitectura del Sistema FAM 98
3.9 Sistemas de Control Fuzzy y Neuronales 99
3.9.1 Parte Superior del Camión 99
3.9.2 Sistema Controlador Fuzzy del Camión 100
3.9.3 Sistema Neuronal del Camión 105
3.9.4 Comparación de los Sistemas Fuzzy y Neuronal 106
3.9.5 Análisis de Sensibilidad 107
3.9.6 Fuzzy Adaptable del Camión 109
3.9.7 Controlador Fuzzy del Camión y Remolque 112
3.9.8 Sistema Controlador BP del Camión y Remolque 115
3.9.9 Sistema Controlador AFAM del Camión y Remolque 116

Conclusiones 119

A Neuronas Biológicas

Introducción I
A.1 Mente y Cerebro II
A.2 Microcircuitos en el Sistema Nervioso VI
A.2.1 Dendritas y Sinapsis VII
A.2.2 Sinapsis Dendrodendritica VIII
A.3 Funciones Sensoriales IX
A.4 Distribución de Microcircuitos X
A.5 Otra Interacción Local XI

PROLOGO

Dentro de la tecnología de vanguardia son variados los temas que podemos desarrollar, sin embargo en esta ocasión nos dedicaremos al estudio de los sistemas Neuro-Fuzzy, haciendo una combinación de las redes neuronales artificiales y la tecnología fuzzy logic.

Sistemas de este tipo los podemos encontrar en el abrir o cerrar de una puerta a través de huellas digitales o inclusive por tonos de voz. Aun cuando estos sistemas nos parezcan a la vista muy simples, el intentar comprender su funcionamiento pudiera resultar complicado, de acuerdo al nivel de conocimientos básicos que poseamos sobre el tema. Es por ello que resulta de gran importancia que dentro de las universidades, en los estudiantes se despierte esta inquietud para que tengan un mayor interés y preferencia hacia el estudio de estos temas.

Teniendo presente la observación arriba citada, el presente trabajo tiene como objetivo interesar al lector en el diseño de sistemas de redes neuronales artificiales complementados con la tecnología fuzzy logic, presentando aquí los conceptos básicos para facilitar su comprensión y posterior desarrollo.

Bajo esta idea, la investigación se encuentra dividida en introducción general, tres capítulos y un apéndice. En síntesis: La introducción general nos presenta el marco de composición de las redes neuronales artificiales con sus entradas y salidas. El capítulo uno se refiere a los conceptos generales de la red neuronal artificial presentando una descripción más detallada. El capitulo dos nos presenta lo relacionado a la arquitectura y modelos, además de los diferentes tipos de conexiones. En el capítulo tres podremos encontrar la conexión de redes nauronales artificiales a través de los sistemas fuzzy logic, en este capitulo se podra apreciar con claridad la aplicación y ventajas de la aplicación de dichos sistemas.

INTRODUCCION

Las redes neuronales artificiales son modelos de computadoras que se encuentran inspiradas en la estructura y conducta de verdaderas neuronas biológicas similares al cerebro, pueden reconocer modelos y reorganizar datos. Una red neuronal artificial se compone de objetos llamados unidades, que representan los cuerpos de las neuronas, las unidades son unidas por conexiones, que actúan como los axóns y las dendritas de las neuronas biológicas, estas conexiones multiplican la salida de una unidad por un factor de carga, las conexiones entonces pasan el valor de la salida de la carga a otra unidad, que suma los valores pasados por todas las otras nuevas conexiones. Si el valor total de la entrada excede unos valores mínimos, la unidad es cargada.

Las modificaciones en los modelo de carga constituyen el aprendizaje. En las verdaderas neuronas el aprendizaje se piensa que ocurre en la sinapsis; cuando las fuerzas de conexión entre sinapsis cambian. En las redes neuronales artificiales el aprendizaje ocurre cuando los factores de carga en las conexiones cambian[1].

Algunas de las operaciones que una red neuronal realiza incluyen.

* Clasificación. Un modelo de entrada es proporcionado a la red y la red produce una clasificación representativa como salida.

* Combinación de modelos. Un modelo de entrada es proporcionado a la red y la red produce el modelo correspondiente de salida.

* Conclusión de un modelo. Un modelo de entrada incompleto es proporcionado y la red producirá un modelo de salida que tiene la porción faltante del modelo de entrada.

* Eliminación de Ruido. Un modelo de entrada con un ruido de corrupción es proporcionado y la red quitara uno (o todo) el ruido y producirá una versión limpia del modelo de entrada como salida.

1(Vea "Coma las Redes Neuronales Aprenden de Experiencias", por Geoffrey E. Hinton, página 104).

* Optimización. Un modelo de entrada que contiene los valores iniciales para un problema de Optimización es proporcionado y la red producirá un conjunto de variables que representa una solución al problema.

* Control. Un modelo de entrada que representa el estado común de un controlador y la respuesta deseada para el controlador, la salida sera una serie apropiada de mandatos que creen la respuesta deseada.

La figura I.1 ilustra una típica red neuronal. Cada capa en una red neuronal consiste de una colección de elementos de proceso (PE).

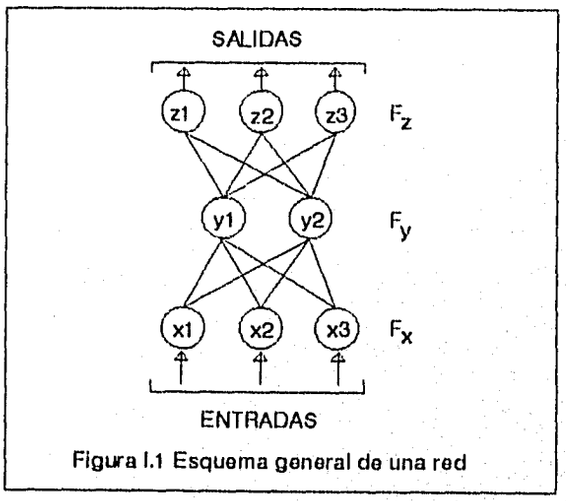


Figura I.1 Esquema general de una red

Cada colección de valores en una red neuronal son tomados de todas sus conexiones de entrada, realiza una operación matemática predefinida (típicamente un producto punto, seguido por una función **PE**) y produce un solo valor de salida. La red neuronal de la figura I.1 tiene tres capas; $\mathbf{F_x}$ que consiste del **PE** { x_1, x_2, x_3}; $\mathbf{F_y}$ que consiste del **PE** { y_1, y_2}; y $\mathbf{F_z}$ que consiste del **PE** { z_1, z_2, z_3} (desde el ultimo al primero, respectivamente), los **PE** son conectados por las cargas de conexión. Como un ejemplo, en la figura I.1 la conexión del $\mathbf{F_x}$ x_1 al $\mathbf{F_y}$ y_2 es la carga de conexión w_{12} (la conexión de x_1 a y_2), la carga de conexión guarda la información. El valor de la carga de conexión a menudo es determinado por un procedimiento de aprendizaje en la red neuronal (aunque a veces son predefinidas y escritas a mano en la red). La red puede aprender ha determinar el ajuste de las cargas, al hacer las operaciones de actualización para cada **PE** y puede recordar la información. Hay varias características importantes ilustradas por la red de la figura I.1 y estas se aplican a todas las redes neuronales:

* Cada **PE** actúa independientemente de todos los demás, cada PE de salida cuenta solamente con sus entradas constantemente disponibles de las conexiones determinadas.

* Cada **PE** cuenta solamente con la información local, la información que es proporcionada por la conexión contigua es todo un **PE** que necesita ser procesado; no necesite saber el estado de alguno otro **PE** donde no tenga una conexión explícita.

* El gran número de conexiones proporciona una gran cantidad de redundancia y facilitan una representación distribuida.

Las primeras dos características anteriores permiten que la red neuronal opere eficientemente en paralelo. Estas características proporcionan redes neuronales inherentes tolerables a fallas y cualidades generales que es muy difícil de obtener en típicos sistemas computacionales. Además de estas características, por arreglo propio de las redes neuronales, la introducción de un proceso no lineal de los elementos (agregando una función **PE** no lineal), y la utilización apropiada de los métodos de aprendizaje, las redes pueden aprender arbitrariamente en diseños no lineales, este es un atributo poderoso. Hay tres situaciones primarias donde las redes neuronales son provechosas:

1.- Situaciones donde solamente se requieren de algunas decisiones de una masiva cantidad de datos (habla y procesamiento de imágenes)

2.- Situaciones donde diseños no lineales tiene que ser automáticamente adquiridos (valoraciones de préstamos y control)

3.- Situaciones donde una óptima solución cercana de un problema de optimización combinatorial es requerido muy rápidamente (fijar el horario de las lineas aéreas y la ruta de mensajes de telecomunicación)

Los fundamentos de redes neuronales consisten del entendimiento de la nomenclatura y una comprensión firme de los rudimentarios conceptos matemáticos que la describen. En un sentido amplio, una red neuronal consiste de tres elementos principales:

1.- Topología, como una red neuronal se organiza en capas y como estas son conectadas.

2.- Aprendizaje, como se guarda la información en la red.

3.- Recordar, como la información guardada en la red se recobrara.

Cada uno de estos elementos serán descritos en detalle después de discutir conexiones, elementos de proceso, y funciones **PE**.

Capítulo **1**

CONCEPTOS DE UNARED NEURONAL ARTIFICIAL

INTRODUCCION.

Cada red neuronal tiene por lo menos dos componentes físicos; conexiones y elementos de proceso (**PE**). La combinación de estos dos componentes crea una red neuronal. Una analogía cómoda es comparar la con cualquier gráfica, donde los bordes son análogos a las conexiones y los nodos son análogos a los elementos de proceso. En suma las conexiones y elementos de proceso, las funciones de compuertas y modelos de entrada/salida son también elementos básicos en el diseño, implementación y utilización de una red neuronal. Después de una descripción de la terminología, cada uno de estos elementos serán examinados en seguida.

1.1.- TERMINOLOGIA.

La terminología de redes neuronales es variada, se utilizara la figura 1.1 para introducir la terminología que se utilizara para la expecificación de una red.

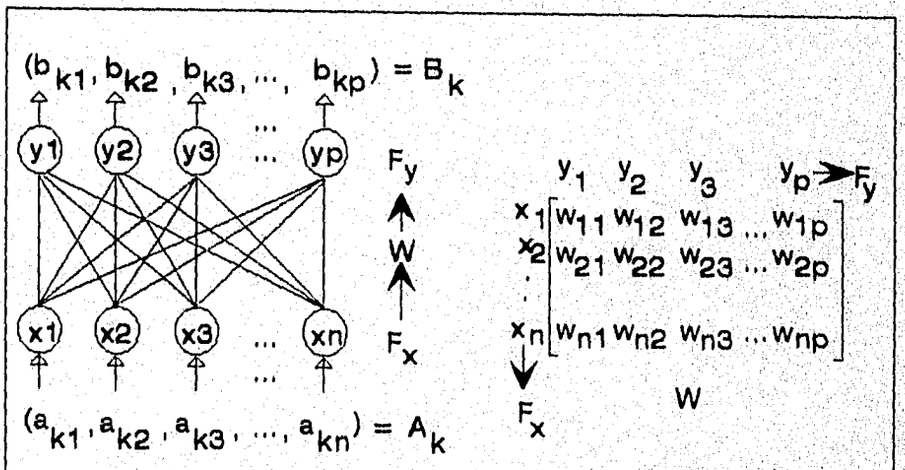


Figura 1.1 Utilizada para la explicación de la terminología de redes.

Los vectores de Entrada y Salida serán denotados por el subíndice de una letra mayúscula del principio del alfabeto.

Los vectores de entrada serán denotados como:

$$A_k = (a_{k1}, a_{k2}, \ldots a_{kn}) \quad k = 1,2,.., m$$

y el vector de salida como:

$$B_k = (b_{k1}, b_{k2}, \ldots b_{kp}); \quad k = 1,2,\ldots, m.$$

Los elementos de proceso en una capa serán denotados por la misma variable del subíndice. La colección de PEs en una capa forman un vector y estos vectores serán denotados por una letra mayúscula del final del alfabeto. En la mayoría de casos, tres capas de **PEs** bastarán. Se denota la capa de entrada de un **PEs** como:

$$F_X = (x_1, x_2, \ldots, x_n)$$

donde cada x_i recibe una entrada de los componentes correspondientes del vector de entrada a_{ki}. La próxima capa de **PEs** será el $\mathbf{F_Y}$, después el $\mathbf{F_Z}$ **PEs** (si esta capa es necesaria). La dimensión de estas capas depende en su utilización. Para la red de la figura 1.1, por ejemplo, la segunda capa de la red es la capa de salida, tanto el número de F_Y tiene que ser igual a la dimensión del vector de salida. En esta ocasión, se denota a la capa de salida como:

$$F_Y = (y_1, y_2, \ldots y_p)$$

donde cada elemento y_j son correlacionado con los elementos **j** de $\mathbf{B_{kj}}$.

Se guardan las cargas de conexión en una matriz de cargas. La matriz de carga sera denotada por letras mayúsculas de la mitad del alfabeto, tal como **U**, **V**, y **W**. Para el ejemplo de la figura 1.1, esta red neuronal de dos capas requiere de una matriz de carga totalmente conectada la capa **n** de $\mathbf{F_X}$ a la capa **p** de $\mathbf{F_Y}$, la matriz muestra la colección completa de cargas entre $\mathbf{F_X}$ y $\mathbf{F_Y}$, donde la carga w_{ij} es la carga de conexión de los elementos **i** de $\mathbf{F_X}$, x_i, a los elementos **j** de $\mathbf{F_Y}$, y_j.

1.1.1.- VECTORES DE ENTRADA Y SALIDA.

Una red Neuronal no puede operar a menos que tenga datos. Algunas redes neuronales requieren solamente de vectores individuales y otras requieren de vectores pares. Note que la dimensión del vector de entrada no es necesariamente la misma que el vector de salida. Cuando una red solamente trabaja con

vectores individuales, es una red autoasociativa. Cuando una red trabaja con vectores pares es heteroasociativa

Una de las claves de cuando aplicar una red neuronal es determinar que tienen que representar los vectores. Por ejemplo, en el reconocimiento del habla hay varios tipos diferentes de características que puede ser empleadas, incluso el código lineal del coeficiente de predicción, el espectro de Fourier, el histograma de cruce de compuertas (o umbrales), correlación de valores cruzados, y muchos otros. La selección apropiada y representación de estas característica pueden afectar grandemente la ejecución de la red.

En algunas ocasiones la representación de las características como la formación de un vector es limitado por el tipo de proceso que la red neuronal puede hacer. Unas redes pueden solamente procesar datos binarios, tal como la red Hopfield, la teoría binaria de la resonancia adaptable y el estado del cerebro en una caja. Otras pueden procesar datos de valores reales tal como la backpropagation y cuantificación de vectores de aprendizaje, crear la mejor colección posible de característica y representar apropiadamente estas característica es el primer paso hacia el éxito de alguna red neuronal solicitada.

1.1.2.- CONEXIONES.

Una red neuronal es equivalente a una gráfica continua (gráfica), Un diagrama que tiene bordes (conexiones) entre nodos **PEs** que permiten que la información fluya solamente en una dirección (la dirección denotada por la flecha), la información fluye en un diagrama por los bordes y es tomada la señal por los nodos. Dentro de la representación gráfica, las conexiones sirven para un solo propósito; determinan la dirección del flujo de la información. Como un ejemplo, en la figura 1.1 la información fluye de la capa F_x por la conexión **W** a la capa F_y. La red neuronal ampliada de la representación gráfica incluye una carga por cada borde (conexión) que modula la cantidad de señal de salida que es pasada al nodo (**PE**) bajo la conexión adyacente. Para mayor sencillez, el doble papel de las conexiones será empleado. En una conexión ambos definen el flujo de la información por la red y modulan la cantidad de información a pasar através de los **PEs**.
Las cargas de conexión se ajustan durante un proceso de aprendizaje que captura la información. Las cargas de conexión que tienen un valor positivo es una conexión de excitación, con valores negativos es de inhibición. Una carga de conexión que tiene un valor de cero es lo mismo que no tener una conexión presente. Por ser permitido solamente un subestado de toda conexión posible que tiene valores no cero.

Es a menudo deseable que para cada **PE** se tenga un valor interno parcial (valor mínimo ó de umbral), la parte (a) de la figura 1.2 muestra el **PE** y_j con tres conexiones de $\mathbf{F_x}$ { w_1, w_2, w_3} y un valor parcial θ_j. Conviene considerar este valor parcial como una conexión extraordinaria w_0 que emana del $\mathbf{F_x}$ x_0, con el agregado que x_0 es siempre igual a 1, como muestra la parte (b). Esta representación matemática equivalente simplifica muchas discusiones. Por lo que este método de representar el valor parcial (mínimo) será intrínsecamente empleado.

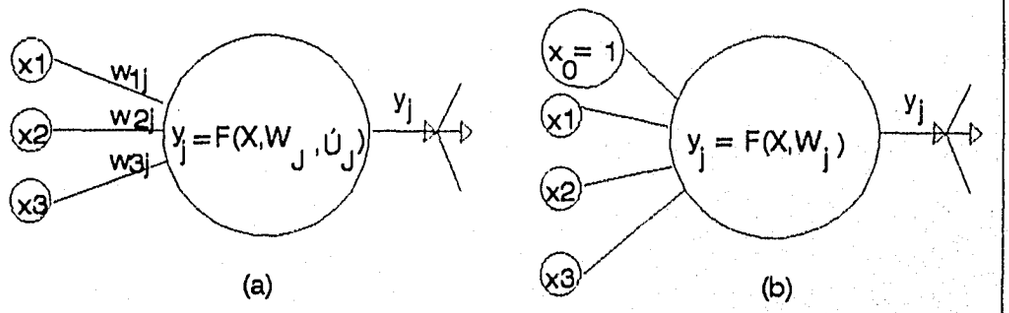


Figura 1.2 Representación de los elementos de proceso (PE)

(a) Muestra un PE con tres conexiones y un valor Θ_j.

(b) Contiene tres conexiones y el valor Θ_j es remplazado con x_0.

1.1.3.- ELEMENTOS DE PROCESO.

El elemento de proceso (**PE**) es la porción de la red neuronal donde se hace todo el calculo. La figura 1.2 ilustra la mayoría de los tipos comunes de **PE**. Un **PE** puede tener conexiones de entrada, como es el caso cuando el **PE** es una capa de entrada y recibe solamente valores del correspondiente componente del modelo de entrada, o puede tener varias cargas de conexión, como es el caso del $\mathbf{F_y}$ mostrado en la figura 1.1, donde hay una conexión de cada $\mathbf{F_x}$ a cada $\mathbf{F_y}$. Cada **PE** toma la información que ha sido enviada bajo sus determinadas conexiones y produce un solo valor de salida. Hay dos importantes cualidades que un PE tiene que poseer:

1.- El **PEs** requiere solamente información local. Toda la información necesaria para que un **PE** produzca un valor de salida esta presente en las entradas y residen dentro del PE, ningun otro valor de información es requerido en la red.

2.- El **PEs** produce solamente valores de salida. Este valor de salida es propagado bajo las conexiones del **PE** emisor a otro **PE** receptor, o servira como una salida de la red.

Estas dos cualidades permiten que las redes neuronales operen en paralelo. Como fue hecho con las conexiones, la clasificación del valor del PE se refiere su etiqueta a un sinónimo. Por ejemplo, el elemento **j** del $\mathbf{F_Y}$ en la figura 1.1 es y_j y el valor del **PE** es también y_j.

Hay varios mecanismos para calcular la salida de un elemento procesado. El valor de salida del **PE** mostrado en la figura 1.2(b), y_j, es una función de salida de la capa anterior, $F_X = X = (x_1, x_2,\ldots,x_n)$ y las cargas de $\mathbf{F_X}$ a y_j, $W_j = (w_{1j}, w_{2j},\ldots.w_{nj})$. Matemáticamente, la salida de este **PE** esta en función de sus entradas y sus cargas,

$$y_j = F(X, W_j) \qquad (1\text{-}1)$$

En seguida tres ejemplo de funciones de actualización.

1.1.3.1.- COMBINACION LINEAL.

La mayoría de las operaciones realizadas en la computación común ejecuta un **PE** por una combinación lineal (producto punto) del valor de entrada **X** con la carga determinado de la conexión W_j, posiblemente seguido por una operación no lineal. Para el **PE** de la figura 1.2(b), la salida y_j, es calculada de la ecuación.

$$y_j = f(\sum_{j=0}^{n} x_i\, w_{ij}) = f(X \bullet W_j) \qquad (1\text{-}2)$$

donde $W_j = (w_{1j}, w_{2j},\ldots.w_{nj})$ y **f** es una de las funciones **PE** no lineales descritas en la sección 1.2. El producto punto actualizado tiene la misma calidad apelada esta es intrínseca a su calculo. Usando la relación $A_k \bullet W_j = \cos(A_k, W_j)\|A_k\|\|W_j\|$, se ve que es más grande el producto punto (asumiendo unos largos y fijos A_k y W_j), los dos vectores son muy semejantes. Por lo tanto, el producto punto puede ser visto como una medida semejante.

1.1.3.2.- CONEXIONES DE VARIACION MEDIA.

En algunas ocasiones un **PE** tendrá dos conexiones interconectandas a **PEs** en vez de una, como muestra la figura 1.3.

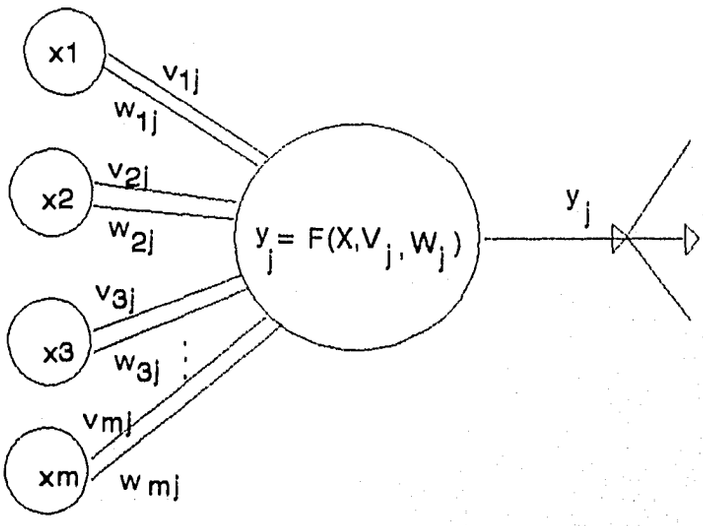


Figura 1.3 Ejemplo de conexiones de variación media.

Utiliza una de estas conexiones dobles que permitir poner una de las conexiones limitantes que representan la media de una clase y la otra, la variación de la clase[1]. En este caso, el valor de salida del **PE** depende de las entradas y las dos conexiones colocadas; estas son, $y_j = F(X,V_j,W_j)$, donde las conexiones medias son representadas por $W_j = (w_{1j}, w_{2j},....,w_{nj})$ y las conexiones de variación $V_j = (v_{1j}, v_{2j},...,v_j)$ para el PE y_j. Con este esquema, la salida de y_j es calculada de la diferencia entre la entrada X y la media W_j, dividido por la variación V_j, la cantidad resultante al cuadrado, y pasa este valor por una función Gaussiana no lineal (PE) que producirá el valor final de la salida como sigue:

$$y_j = g(\sum_{i=1}^{n}(w_{ij} - x_i / v_{ij})^2) \qquad (1\text{-}3)$$

1 Robinson A., M. Niranjan, and F. Fallside (1988), "Generalizing the Nodes of the Error Propagation Network", Cambridge University Engineering Department, Technical Report CUEDIF-INENG/TR.25.

donde la función Gaussiana no lineal es:

$$g(x) = \exp (-x^2 / 2) \qquad (1\text{-}4)$$

Note que es posible quitar las dos conexiones en una red de variación media, si la variación es conocida y estacionaria, dividida por la variación del proceso anterior de la red neuronal, la sección 1.2.5 describe la función Gaussiana no lineal (**PE**) a detalle.

1.1.3.3.- CONEXIONES MINIMAS/MAXIMAS.

Otro uso menos común de conexiones dobles es asignar uno de los vectores limitadores, es decir V_j, tiene el mínimo limite para la clase y el otro vector W_j, tiene el máximo limite para la misma clase. Medir la cantidad del modelo de entrada que cae dentro (del resultado) de los limites que producirá un valor mínimo ó máximo de activación[2]. La figura 1.4 ilustra esta noción por una representación gráfica para los limites mínimos y máximos.

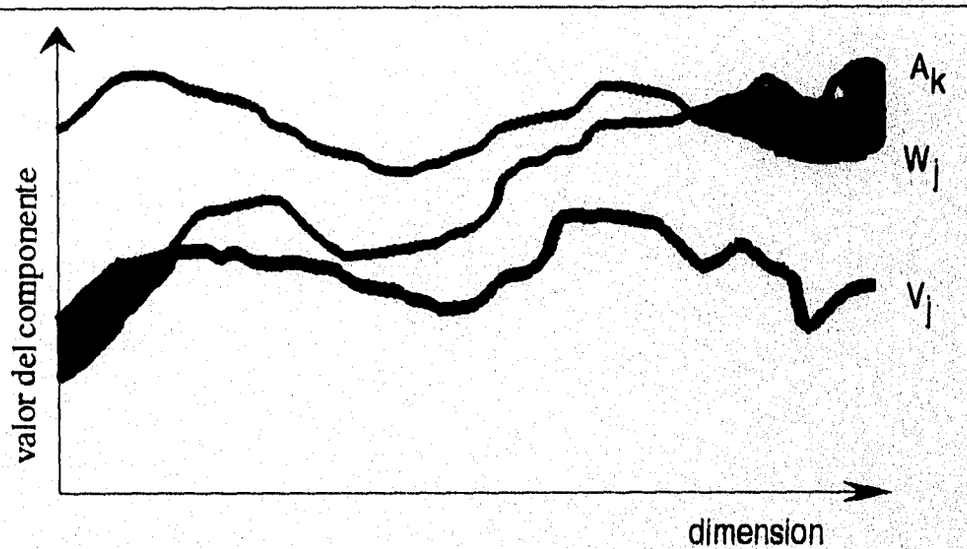


Figura 1.4 Representación grafica de los limites máximos y mínimos

2 Simpson P. (1991), "Fuzzy min-max Classification with Neural Networks", Heuristic, vol. 4, No. 7, pp. 1-9.

El eje de las ordenadas de la gráfica representa los valores mínimos y máximos de cada elemento del vector y el eje de las abscisas de la gráfica representa la dimensión del espacio clasificado. Se compara el modelo de entrada X con el limite de la clase, la cantidad de diferencia entre el limite de la clase, V_j y W_j, y **X** es mostrada en la región sombreada la media de esta región sombreada produce el valor de activación y_j.

Refiriendonos otra vez a la figura 1.4, note que el máximo W_j es el punto máxima permitido en la clase **j** y el limite mínimo V_j es el punto mínimo permitido en la clase **j**. Medir el grado en que **X** no cae entre V_j y W_j es realizado por la cantidad de la medida relativa de que **X** cae fuera de la clase **j**. Una medida que fue propuesta por (Simpson 1990) y utilizada por (Kosko, 1990)[2], la fuzzy medida de la cresta baja del riso, que resultó en la ecuación:

$$y_j = (1 - \text{cresta superior } (X, W_j))\ (1 - \text{cresta baja}) \qquad (1\text{-}5)$$

Cuando $y_j = 1$, **X** es situada completamente dentro de los limites mínimos/máximos. Cuando $y_j = 0$, **X** cae completamente fuera de los limites mínimos/máximos. Cuando $0 < y_j < 1$, el valor describe el grado que **X** cae entre los limites mínimos/máximos.

1.2.- FUNCIONES DE LOS ELEMENTOS DE PROCESO (PE).

Las funciones **PE**, también se refirieren como funciones de activación o funciones de aplastamiento, Un **PE** traza un posible dominio infinito a un rango preespecificado. Aunque el número de posibles funciones **PE** es infinito, cinco son regularmente empleadas por la mayoría de redes neuronales:

1.- Función Lineal.
2.- Función Escalon.
3.- Función Rampa.
4.- Función Sigmoide.
5.- Función Gaussiana.

Con la excepción de la función lineal, todas estas funciones introducen una dinámica no lineal en la red para determinado valor de salida dentro de un rango fijo. Se describe brevemente cada función **PE** y se muestran en las partes (a)-(e) de la figura 1.5.

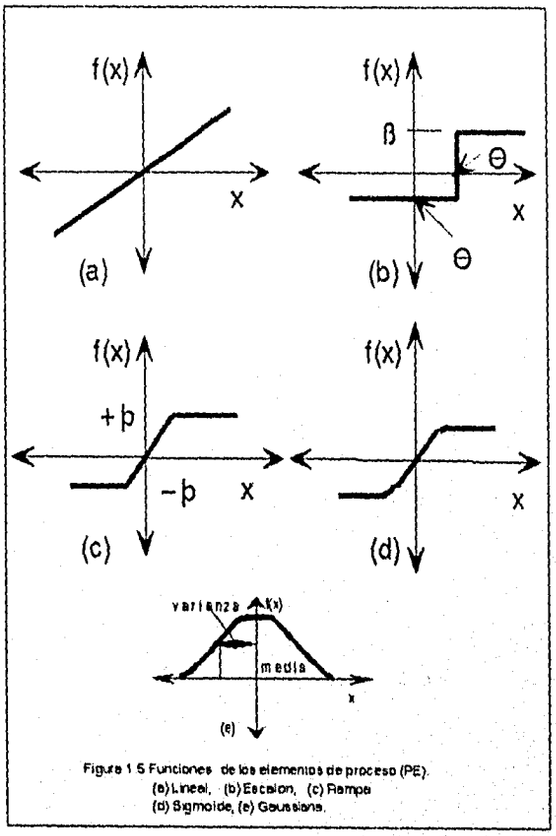


Figura 1.5 Funciones de los elementos de proceso (PE).
(a) Lineal, (b) Escalon, (c) Rampa
(d) Sigmoide, (e) Gaussiana.

1.2.1.- FUNCION LINEAL.

La función lineal (figura 1.5(a)) produce una salida lineal modulada de la entrada **x** como es descrito por la ecuación:

$$f(x) = \alpha x \qquad (1\text{-}6)$$

donde **x** esta en el rango de los números reales y α es un escalar positivo. Si $\alpha = 1$, es equivalente a quitar la función **PE** completamente.

1.2.2.- FUNCION ESCALON.

La función de escalon (fig. 1.5(b)) produce solamente dos valores, ß y δ. Si la entrada a la función **PE x** es igual o mayor a un valor θ predefinido, entonces la función de escalon produce el valor **ß**; de otra manera produce el valor $-\delta$, donde **ß** y δ son escalares positivos. Matemáticamente se describe esta función como:

$$f(x) = \begin{cases} ß \text{ si } x >= \theta \\ -\delta \text{ si } x < \theta \end{cases} \qquad (1\text{-}7)$$

Típicamente, la función de escalon **PE**, produce un valor binario en respuesta al signo de la entrada, emite +1 si **x** es positiva y 0 si no lo es. Para las asignaciones $\beta = 1$, $\delta = 0$, y $\theta = 0$, la función de paso se hace una función de paso binario:

$$f(x) = \begin{cases} 1 \text{ si } x >= 0 \\ 0 \text{ de otra manera} \end{cases} \qquad (1\text{-}8)$$

que es común a redes neuronales tal como la red neuronal Hopfield y la memoria asociativa bidireccional. Una variación pequeña de la ecuación 8 es la función bipolar **PE**:

$$f(x) = \begin{cases} 1 \text{ si } x >= 0 \\ -1 \text{ de otra manera} \end{cases} \qquad (1\text{-}9)$$

que reemplaza el valor de salida 0 con un -1. En sistemas de recompensa por golpe se utiliza tal como la asociativa recompensa por penalización[3], el valor negativo es utilizado para asegurar los cambios, mientras no haya un 0 deseado.

1.2.3.- FUNCION RAMPA.

La función de rampa (fig. 1.5(c)), es una combinación de las funciones lineal y de escalon. La función de rampa **PE**, coloca limites superiores e inferiores para el valor que la función produce y permite una respuesta lineal entre los limites. Este punto de saturación es simétrico alrededor del origen y es discontinuo en los puntos de saturación. La función de rampa se define como:

$$f(x) = \begin{cases} \tau \text{ si } x >= \tau \\ \tau \text{ si } |x| < \tau \\ -\tau \text{ si } x <= -\tau \end{cases} \qquad (1\text{-}10)$$

donde Þ es el valor de saturación para la función, y los puntos $x = \tau$ y $x = -\tau$ es donde la discontinuidad en "f" existe.

---

3 Barto A. (1985), "Learning y Statical Comparation of Self-Intersted Neuron-Like Computing Units", Human Neurobiol, Vol. 4, pp. 229-256.

1.2.4.- FUNCION SIGMOIDE.

La función sigmoide (fig. 1.5(d)), es una versión continua de la función de rampa. La función sigmoide (en forma de "S"), es una función limitada, monótona, una función no decreciente que proporciona un grado no lineal de respuesta dentro de un rango predefinido.
La mayoría de funciones sigmoide son las funciones logísticas:

$$f(x) = 1 \; / \; 1 + e^{-\alpha x} \qquad (1\text{-}11)$$

donde $\alpha > 0$ (usualmente $\alpha = 1$), que proporciona un valor de salida de 0 a 1. Esta función es familiar en estadística (como la función Gaussiana de distribución), en química (describe las reacciones catalíticas), y en sociología (describir el crecimiento de la población (humana)). Note que una relación entre 1-11 y 1-8 existe, cuando $\alpha = \infty$ en (1-11), el declive de la función sigmoide entre 0 y 1 se hace infinitamente empinado y en efecto, se forma la función de escalon descrita por la ecuación (1-8). Dos alternativas a la logística función sigmoide es la tangente hiperbólica.

$$f(x) = \tanh(x) \qquad (1\text{-}12)$$

en los rangos de -1 a 1, y aumentando la proporción del cuadrado de los rangos:

$$f(x) = \begin{cases} x^2 \; / \; 1 + x^2 & \text{si } x > 0 \\ 0 \text{ de otra manera} \end{cases} \qquad (1\text{-}13)$$

en los rangos de 0 a 1.

1.2.5.- FUNCION GAUSIANA.

La función Gaussiana (fig. 1.5(e)), es una función radial (simetría del origen), que requiere de un valor de variación $v > 0$ para formar la función Gaussiana. En unas redes se utiliza la función Gaussiana en conjunción con una colección de dobles conexiones como lo describe la ecuación (1-3), y en otras ocasiones la variación es predefinida (Specht, 1990). En esta ocasión, la función PE es:

$$f(x) = \exp(-x^2 \; / \; v) \qquad (1\text{-}14)$$

donde **x** es la media y **v** es la variación predefinida.

1.3.- COMO APRENDEN LAS REDES NEURONALES.

La mayoría de las cualidades de las redes neuronales se refieren a su habilidad de aprender. Aprender, en este contexto, es definido como un cambio en el valor de las cargas de conexión que resulta en la captura de la información que puede más tarde ser recordada. Varios procedimientos están disponibles para cambiar el valor de las cargas de conexión. Después de presentar una terminología, ocho métodos de aprendizaje serán descritos. Para continuar con la discusión, el algoritmo de aprendizaje será descrito en notación de modo punto (como notación opuesta a un vector). En suma, el algoritmo del aprendizaje, será descrito como una ecuación de tiempo discreto (opuesta a una de tiempo continuo). La ecuación de tiempo discreto es más accesible a simulaciones de computadoras digitales.

1.3.1.- APRENDIZAJE SUPERVISADO VERSUS NO SUPERVISADO.

Todos los métodos de aprendizaje pueden ser clasificados en dos categorías: aprendizaje supervisado y no supervisado. El aprendizaje supervisado es un proceso que incorpora un maestro externo y/o información global. El algoritmo del aprendizaje supervisado se discutira en la sección siguiente incluye la corrección de errores de aprendizaje, refuerzo del aprendizaje, aprendizaje stochastic, y sistemas de hardwired. Ejemplos de aprendizaje supervisado incluyen el decidir cuando detener el aprender, decidir cuánto tiempo y con qué frecuencia presentar cada asociación para el aprendizaje y suplir la ejecución de información (error). El aprendizaje supervisado es clasificado en dos subcategorias: Aprendizaje estructural y temporal. El aprendizaje estructural se refiere a encontrar la mejor relación posible de entrada/salida para cada individual modelo de parejas. Ejemplos de aprendizaje estructural incluyen modelos iguales a y clasificación de modelos. La mayoría de algoritmos de aprendizaje discutidos en las páginas siguientes se enfocan en al aprendizaje estructural. El aprendizaje temporal se refiere a capturar una serie de modelos necesarios para lograr resultados finales. En el aprendizaje temporal, la respuesta actual de la red es dependiente de las entradas y respuestas anteriores. En el aprendizaje estructural, no hay tal dependencia. Ejemplos de aprendizaje temporal incluyen la predicción y el control. El algoritmo del refuerzo del aprendizaje sera discutido con un ejemplo de un procedimiento de aprendizaje temporal.

El aprendizaje no supervisado, también es referido como una auto-organización, es un proceso que no incorpora un maestro externo y contando solamente con la información local durante todo el proceso de aprendizaje. El aprendizaje no supervisado organiza los datos presentes y descubre sus emergentes propiedades colectivas. Ejemplo de aprendizaje no supervisado

en las secciones siguientes incluyen el aprendizaje Hebbian, principales componentes del aprendizaje, la diferencial del aprendizaje Hebbian, aprendizaje minimo/maximo, y el aprendizaje competido.

1.3.2.- APRENDIZAJE FUERA DE LINEA VERSUS APRENDIZAJE EN LINEA.

La mayoría de técnicas de aprendizaje utilizan el aprendizaje fuera de linea. Cuando todo el modelo colocado es utilizado para condicionar las conexiones anteriores a ser utilizadas en la red, es llamado aprendizaje fuera de linea. Por ejemplo, el algoritmo de aprendizaje backpropagation (vea Sección 1.9.2), ajusta las conexiones en redes neuronales de capas múltiples, pero requiere de miles de ciclos para todas las parejas de modelos hasta que la ejecución deseada de la red es alcanzada. Una vez que la red es adecuadamente ejecutada, las cargas son limpiadas y el resultado de la red es utilizado para ser recordado mas tarde. Los sistemas de aprendizaje fuera de linea tiene el requisito intrínseco que todos los modelos tienen que ser residentes para recordarlos, tales requisito no permiten tener nuevos modelos incorporados automáticamente en la red. La mayoría de estos nuevos modelos tienen que ser agregados a la colección entera de modelos y una reconversión de la red tiene que ser realizada.

No todas las redes neuronales realizan aprendizaje fuera de linea. Algunas redes pueden agregar información nueva "en la ejecución" no destructiva. Si un modelo nuevo necesita ser incorporar en las conexiones de la red, puede hacerse inmediatamente sin alguna pérdida de información guardada antes. La ventaja de las redes de aprendizaje fuera de linea es que usualmente proporcionan soluciones superiores a problemas difíciles tales como clasificaciones no lineales, pero el aprendizaje en linea permite a la red aprender. Un desafío en el porvenir de redes neuronales de computo es el desarrollo de técnicas de aprendizaje que proporcionen un alto rendimiento del aprendizaje en linea sin costos extremosos.

1.4.- CORRELACIONES DE HEBBIAN.

La forma simple de ajustar los valores de las cargas de conexión en una red neuronal esta basada sobre la correlación de los valores de activación. Los motivos fundamentales de ajustes por la correlación ha sido atribuida a Donald O. Hebb (1949), de la hipótesis del cambio eficaz en una sinapsis (su habilidad de activación o como lo simulamos en nuestras redes neuronales, la carga de conexión), es impulsado por una habilidad neuronal para producir una señal de salida. Si una neurona **A** fue activada y esta activación de **A** causa una conexión de activación con una neurona **B**, entonces la

eficacia de la conexión sinaptica entre A y B tiene que crecer.

1.4.1.- VALORES ILIMITADOS DE LOS PE Y CARGAS.

Esta forma de aprendizaje, ahora es comúnmente referido como aprendizaje Hebbian, ha sido matemáticamente caracterizado como la correlación de ajuste de cargas.

$$w_{ij}^{new} = w_{ij}^{old} + a_{ki}\, b_{kj} \qquad (1\text{-}15)$$

donde $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, p$, x_i es el valor de los elementos **i** del **PE** en la capa F_X de una red de dos capas; y_j es el valor de los elementos **j** del **PE** F_Y y las cargas de conexión entre los dos **PE** es w_{ij}. En general, el valor del **PE** puede estar en el rango de los números reales y las cargas son ilimitadas. Cuando el valor del **PE** y el valor de la conexión son ilimitadas, estas redes neuronales de dos capas son sensibles a la teoría lineal de sistemas. Una red neuronal como la lineal memoria asociativa[4], emplea este tipo de aprendizaje y analiza las salidas de estas redes con la teoría lineal de sistemas como una guía. El número de vectores en una red es mostrado al utilizar (1-15) con cargas ilimitadas y las conexiones pueden ser limitadas por la dimensión de los modelos de entrada.

1.4.2.- VALORES PE LIMITADOS Y CARGAS ILIMITADAS.

Recientes, implementaciones que restringen el valor del **PE** y/o las cargas de (1-15) han sido empleados. Estas redes (llamadas redes Hopfield porque John Hopfield había entusiasmado a personas de su potencial[5]), restringen el valor del **PE** a un numero binario {0,1} o un valor bipolar {-1, +1}. Se utiliza la ecuación (1-15) para estos tipos de correlaciones.
Estas redes de valores discretos típicamente involucran una forma de aprendizaje llamado de realimentación, resultando en la necesidad de demostrar que cada entrada producirá una respuesta estable (salida). Limitar el valor del **PE** durante el proceso introducidos no linealmente en el sistema, eliminando una de las teorías de análisis de sistemas lineales que anteriormente había sido realizado. Agregar

---

4 Anderson J. (1970), "Two Models for Memory Organization Using Interactive Traces", Math. Biosci., Vol. 8, pp. 137-160.

5 Hopfield J. (1982), "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Nat. Acad. Sci., U.S.A., Vol. 79, pp. 2554-2558.

realimentacion en el proceso de aprendizaje forma un valor discreto, no lineal, sistemas dinámicos. Las versiones de capas simples de esta regla de aprendizaje son descritas como redes Hopfield y las versiones de dos capas como la memoria asociativa bidireccional. Algunos de los análisis más recientes de estas redes fueron hechos por Amari S. que utilizo la teoría estadística de la neurodinamica[6] , mostro que estas redes son estables. Más tarde, Hopfield hallo un método alternativo para comprobar la estabilidad. También, el número de vectores que una red neuronal de esta forma puede guardar es limitado.

1.4.3.- VALORES LIMITADOS DE LOS PE y CARGAS LIMITADAS.

A veces ambos valores del **PE** y las cargas son limitados. Éstas son dos formas de tales sistemas: La primera forma es simplemente un promedio corriente de la cantidad de correlaciones entre dos **PE**. La ecuación es:

$$w_{ij}^{new} = (1 / k)\ (a_{ki} b_{kj} + (k - 1)\ w_{ij}^{old}) \qquad (1\text{-}16)$$

que describe la media de la correlación durante la presentación de los elementos **k** del par de modelos (A_k, B_k), donde $A_k = (a_{k1}, a_{k2}, \ldots, a_{kn})$; $B_k = (b_{k1}, b_{k2}, \ldots, b_{kp})$; y **k** es el numero del modelo actual, k = 1,2,.., m. La misma información que fue guardada al utilizar (1-15) es guardada al utilizar (1-16), la existencia de cargas de conexión simplemente limita el intervalo de unidades en este caso.

El otro ejemplo de correlación en redes neuronales es la ecuación del aprendizaje con valores limitados **PE** y cargas limitadas, que es la ampliación del código de la ecuación, definido por:

$$w_{ij}^{new} = \begin{cases} 1 \text{ si } a_{ki} b_{kj} = 1 \\ 1 \text{ si } w_{ij}^{old} = 1 \\ 0 \text{ de otra manera} \end{cases} \qquad (1\text{-}17)$$

Esta ecuación asigna un valor binario a una conexión si el **PE** en cada terminal de la conexión tienen ambos el valor de 1

---

6 Amari S. (1977), "Neural Theory of Association and Concept Formation", Biol. Cybernet, Vol. 26, pp. 175-185.

sobre el curso del aprendizaje. La ecuación de aprendizaje es equivalente a hacer la operación lógica:

$$w_{ij}^{new} = (a_{ki} \cap b_{kj} \cup w_{ij}^{old} \qquad (1\text{-}18)$$

donde $\cap$ y $\cup$ son las operaciones de intersección y unión, respectivamente. Redes neuronales que utilizan esta forma de aprendizaje incluyen la Matriz del Aprendizaje y la memoria asociativa Willshaw. Esta ecuación de aprendizaje tenía una gran cantidad de potencial. Por la codificación de información en un ventor binario (es decir, por ejemplo, solamente 32 componentes de salida de 1 millón son puestos a 1, los otros son puestos a 0), es posible guardar una cantidad tremenda de información en la red. El problema miente en cuanto a la creación del código necesario para hacer tal almacenaje tan denso[7].

1.5.- COMPONENTE PRINCIPAL DEL APRENDIZAJE.

Unas redes neuronales tienen un algoritmo de aprendizaje diseñado para producir, de una carga dada, el componente principal de los modelos dados de entrada. El componente principal de un dato dado es encontrado por primera vez al formar la matriz de covariacion (o correlación) de un modelo dado y entonces encontrar el mínimo de los vectores ortogonales de ese tramo del espacio de la matriz de covariacion. Una vez que la base fija es encontrada, es posible reconstruir algún vector en el espacio con una combinación lineal de la base del vector. El valor de cada escalar en la combinación lineal representa la "importancia" de la base del vector. Es posible pensar en la base del vector como una característica del vector y la combinación de éstos presentada como una característica especial del vector que es utilizada para construir modelos. Por lo tanto, el propósito de un componente principal en la red es descomponer un modelo de entrada en valores que representen la importancia relativa de las características fundamentales de los modelos.

El primer trabajo con el componente principal del aprendizaje fue hecho por Oja E., que razonó el aprendizaje Hebbian con una llamada realimentación que automáticamente fuerza en las cargas la extracción del componente principal de los datos de entrada.

La ecuación utilizada por Oja es:

---

7 Helcht-Nielsen R. (1990), Neuro Computing, Reading, MA:Addison-Wesley.

$$w_{ij}^{new} = w_{ij}^{old} + b_{kj} (\alpha a_{ki} - \beta b_{kj} w_{ij}^{old}) \qquad (1\text{-}19)$$

donde a_{ki} son los componente **i** de los elementos k del modelo de entrada A_k, **i** = 1, 2,.. n; b_{kj} son los componentes **j** de los elementos **k** del modelo de salida B_k, **j** = 1, 2,.., p; **k** = 1, 2,..., m; α y β son constantes diferentes de cero.

Una variante del trabajo de Oja ha sido desarrollada por Sanger (1989) y es descrita por la ecuación:

$$w_{ij}^{new} = w_{ij}^{old} + \tau k \, (a_{ki} b_{kj} - b_{kj} \sum_{n=1}^{i} y_h \, w_{jh}) \qquad (1\text{-}20)$$

donde las variables son semejantes a las de (1-19) con la excepción de las constantes diferentes de cero, el tiempo de decrecimiento del aprendizaje, el parámetro τ**k**. Las ecuaciones (1-19) y (1-20) son muy semejantes; la diferencia de los paréntesis es que (1-20) incluye más información en la llamada realimentación y utiliza un valor decayente del aprendizaje. Ha habido muchos análisis y aplicaciones del componente principal de redes. Tiene que ser notado que ambos Oja y Sanger en el componente principal de redes solamente extrajeron el primero componente principal el "uno" y ellos se limitan a redes con **PE** lineales.

1.6.- LA DIFERENCIAL DEL APRENDIZAJE HEBBIAN.

El Aprendizaje Hebbian es extendido al capturar los cambios temporales que ocurre en una serie de modelos. Esta ley de aprendizaje, es llamada diferencial del aprendizaje Hebbian, ha sido independientemente derivada por Klopf (1986) en la forma de tiempo discreto y por kosko en la forma de tiempo continuo. La forma general tiene algunas variantes y unas semejanzas con leyes de aprendizaje de las secciones siguientes. Otras varias combinaciones han sido estudiadas más allá de las presentadas aquí.

1.6.1.- DIFERENCIAL BASICA DEL APRENDIZAJE HEBBIAN.

La diferencial del aprendizaje Hebbian correlaciona los cambios en los valores de activación del **PE** con la ecuación:

$$w_{ij} (t + 1) = w_{ij}(t) + \Delta x_i(t) + \Delta y_j (t - 1) \qquad (1\text{-}21)$$

donde $\Delta x_i = x_i(t) - x_i(t - 1)$ es la cantidad de variación en los elementos **i** del **PE** F_x al tiempo **t** y $\Delta y_j (t - 1) = y_j(t - 1) - y_j(t - 2)$, es la cantidad de variación de los elementos **j** del **PE** F_y, a tiempo **t - 1**.

1.6.2.- APRENDIZAJE CONDUCIDO POR EL REFORZAMIENTO.

Klopf A. utilizo el caso más general de (1-21) capturando los cambios del **PE** F_X sobre el punto anterior al tiempo **k** y modulando cada cambio por el correspondiente valor de la carga para la conexión en una red neuronal de dos capas. La ecuación Klopf es:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta y_j \sum_{n=1}^{K} \alpha\,(t-h)\,|w_{ij}(t-h)|\,\Delta x_i\,(t-h) \qquad (1\text{-}22)$$

donde $\alpha(t - h)$ es una función decreciente en el tiempo que regula la cantidad de variación y $w_{ij}(t)$ es el valor de la conexión del x_i al y_j al tiempo **t**. Klopf se refiere a los cambios sinapticos con $\Delta x_i(t - h)$, **h** = 1, 2,..., k, como conductores y los cambios postsinapticos $\Delta y_j(t)$ como el reforzamiento: por lo tanto el nombre de aprendizaje conducido por el reforzamiento.

1.6.3.- CORRELACION COVARIANTE.

Sejnowski T. propone la correlación covariante de los valores de activación del **PE** en una red neuronal de dos capas utilizando la ecuación:

$$w_{ij}^{new} = w_{ij}^{old} + \mu[\,(a_{ki} - \bar{x}_i)\,(b_{kj} - \bar{y}_j)\,] \qquad (1\text{-}23)$$

donde los términos entre corchetes representan la covariancia, la diferencia entre el supuesto valor (medio) de activación de los valores (x_i y y_j) testados del **PE** y los valores de los vectores de entrada y salida (a_{ki} y b_{kj}), respectivamente. El parámetro $0 < \mu > 1$ es la velocidad de aprendizaje. El valor testado en el **PE** representa el valor medio del **PE**.

Sutton R. y Barto A. proponen un tipo semejante de regla de aprendizaje covariante, sugieren la correlación de los valores esperados de x_i con la variación de y_j como se expresa en la ecuación:

$$w_{ij}^{new} = w_{ij}^{old} + \mu x_i\,(b_{kj} - y_j) \qquad (1\text{-}24)$$

1.7.- El APRENDIZAJE COMPETIDO.

El aprendizaje competido, lo introdujo Grossberg S. y Malsburg C., es un método que automáticamente crea clases para un modelo dado de entrada. El aprendizaje competido es un procedimiento de dos pasos que une el proceso de recordar con el proceso de aprendizaje en una red neuronal de dos capas. En la figura 1.6 cada $\mathbf{F_x}$ representa un componente del modelo de entrada, y cada $\mathbf{F_y}$ representa una clase.

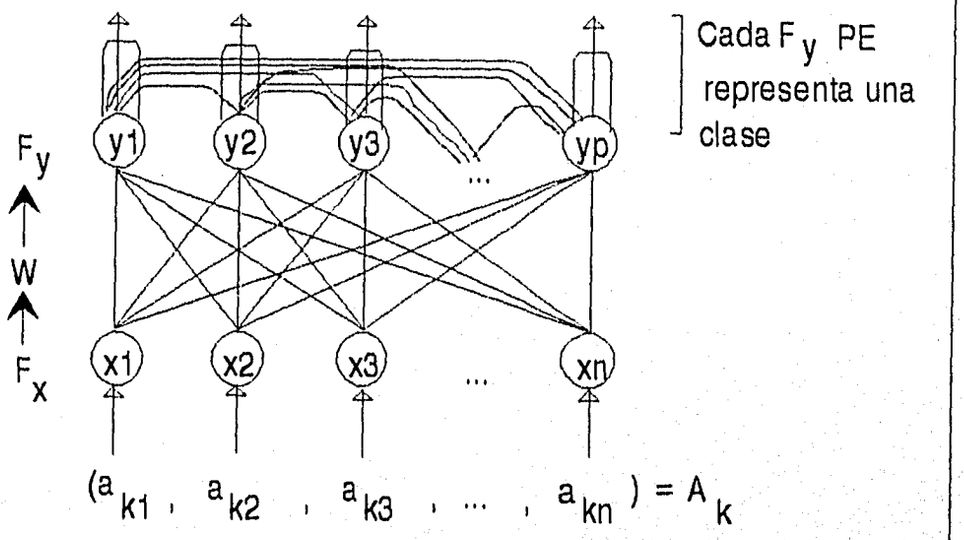


Figura 1.6 Ejemplo de uno red de aprendizaje competido

Paso 1: Determinar la ganancia del F_y. Un modelo de entrada A_k es pasado por las conexiones de la capa de entrada F_x a la capa de salida F_y en un modelo con alimentación controlada utilizando la ecuación del producto punto actualizada:

$$y_j = \sum_{i=1}^{n} x_i \bullet w_{ij} \qquad (1\text{-}25)$$

donde x_i son los elementos **i** del **PE** en la capa de entrada F_x, **i** = 1, 2,..., n, y_j son los elementos **j** del **PE** en la capa de salida F_y, **j** = 1, 2,..., p, y w_{ij} son los valores de las cargas de conexión entre x_i y y_j. Cada colección de conexiones que terminan en un PE F_B, digamos y_j, es un vector de referencia $W_j = (w_{1j}, w_{2j},..., w_{nj})$ que representa la clase **j**. El vector de referencia W_j es cerrado en la entrada A_k que tiene que proporcionar el alto valor de activación. Si los modelos de entrada A_k, **K** = 1, 2,..., m, y el vector de referencia W_j, **j** = 1, 2,.... p, son normalizados a una larga unidad Euclidiana, entonces se tiene la siguiente relación:

$$0 <= (y_j = A_k \bullet W_j = \sum_{i=1}^{n} a_{ki} w_{ij}) <= 1 \qquad (1\text{-}26)$$

donde el A_k es muy semejante a W_j esto es lo más cercano al producto punto unitario (vea la sección 1.1.3.1). Los valores del producto punto "y_j" son utilizados como los valores iniciales para el competidor ganador de todas las iteraciones (vea la sección 3.3.4): El resultado de estas iteraciones es idéntico a buscar el **PE** F_y y encontrar el **PE** con un largo valor del producto punto. Utilizando la ecuación:

$$y_j = \begin{cases} 1 \text{ si } y_j > y_k \text{ para todo } j <> k \\ 0 \text{ de otra manera} \end{cases} \qquad (1\text{-}27)$$

es posible encontrar el **PE** F_y por el valor mas alto del producto punto, llamado el PE ganador. El vector de referencia asociado con el PE ganador es el vector de referencia ganador.

Paso 2: Ajuste del valor de la conexión ganadora del **PE** F_y. En el aprendizaje competido con un ajuste dinámico del ganador que anteriormente se describio, hay solamente una carga de conexión dada que es ajustada la carga de conexión del vector de referencia ganador. En la ecuación es automáticamente ajustado el vector de referencia ganador y no es otro que:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha(t)\, y_j\, (a_{ki} - w_{ij}) \qquad (1\text{-}28)$$

donde $\alpha(t)$ no es cero, función decreciente en el tiempo. El resultado de esta operación es el movimiento del vector de referencia hacia el vector de entrada. Sobre varios vectores de datos presentados (en el orden de $O(n^3)$), el vector de referencia se hará el centro de la agrupación de datos[8].

Ha habido varias variaciones de este algoritmo, pero uno de los mas importantes es el mecanismo de la conciencia[1]. Para agregar una conciencia a cada F_y, un Fy es solamente

---

8 Kohonen T. (1986), "Learning Vector Quantization for Pattern Recognition", Helsinki University of Technology, Technical Report No. tkk-a601.

1 DeSiene D. (1988) "Adding a Conscience to Competitibe Learning", in Proc., Int. Conf. Neural Networks, vol. 1, pp. 117-124.

permitido como ganador, a si la ganancia es igualmente probable. La ganancia igualmente probable fuerza a mejora en ambos la calidad de la solución y el tiempo de aprendizaje. Redes Neuronales que emplean el aprendizaje competido incluye diseños de características de auto-organización, la teoría de la resonancia adaptable I y II.

1.8.- APRENDIZAJE MAXIMO-MINIMO.

Sistemas clasificados como Máximo-Mínimo utilizan un par de vectores para cada clase (vea la sección 1.1.3). La clase j es representada por el PE y_j y es definido por el vector limitador V_j (el Mínimo vector) y W_j (el Máximo vector). El aprendizaje Máximo - Mínimo en un sistema neuronal es realizado por la ecuación:

$$\overset{new}{y}_{ij} = \min(a_{ki}, \overset{old}{y}_{ij}) \qquad (1\text{-}29)$$

para el mínimo vector y

$$\overset{new}{w}_{ij} = \max(a_{ki}, \overset{old}{w}_{ij}) \qquad (1\text{-}30)$$

para el Máximo vector. Los puntos mínimos y máximos son tratados como un limite para una dada función de calidad/transferencia, que proporciona un mecanismo fácilmente ajustable y analiza las clases existentes formadas en una red neuronal (Simpson, 1991 y 1992)

1.9.- CORRECCION DE ERRORES EN EL APRENDIZAJE.

La corrección de errores en el aprendizaje, ajustando las cargas de conexión entre los **PEs** en proporción a la diferencia entre el valor deseado y el calculado de cada capa de salida del PE. La corrección de errores de aprendizaje en dos capas es realizado en un diseño lineal capturandolo entre los modelos de entrada y salida. La corrección de errores de aprendizaje en múltiples capas en un diseño no lineal pueden ser capturados entre las entradas y salidas. En las siguientes dos secciones, cada una de estas técnicas serán descritas.

1.9.1.- CORRECCION DE ERRORES EN EL APRENDIZAJE EN DOS CAPAS.

Considere la red de dos capas de la figura 1.7, suponga que la carga **W** es inicializada con un valor pequeño aleatorio, el modelo de entrada $\mathbf{A_k}$ es pasado por la carga de conexión **W** que

producirá un valor dado de $\mathbf{F_Y}$, $\mathbf{Y} = (y_1, y_2, \ldots, y_p)$. La diferencia entre el valor de salida calculado **Y** y el valor de salida deseado del modelo $\mathbf{B_k}$ es el error, se calcula el error para cada $\mathbf{F_Y}$ de la ecuación:

$$\delta_j = b_{kj} - y_j \qquad (1\text{-}31)$$

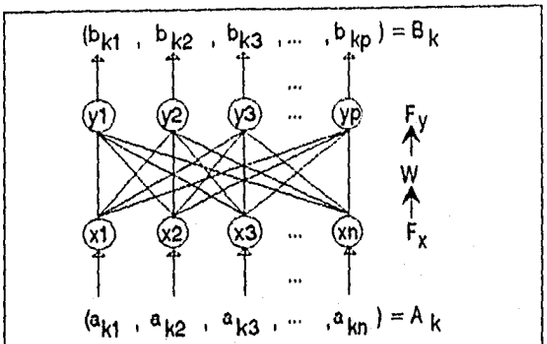


Figura 1.7 Red de dos capas para la corrección de errores.

El error es utilizado para ajustar las cargas de conexión por la ecuación:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha\delta_j a_{ki} \qquad (1\text{-}32)$$

donde el valor positivo de la constante α es la velocidad de aprendizaje.
El fundamento para la regla de aprendizaje descrita por (1-31) y (1-32) es sólido. Para darse cuenta que la mejor solución puede ser lograda cuando todos los errores para un modelo dado a través de toda la salida del **PE**, y_j, es minimizado, nosotros podemos construir el siguiente valor de la función.

$$E = \tfrac{1}{2} \sum_{j=1}^{p} (b_{kj} - y_j)^2 \qquad (1\text{-}33)$$

Cuando **E** es cero, la proyección de la entrada a la salida es perfecta para el modelo dado. Para moverlo en la dirección opuesta a la pendiente del valor de la función con respecto a las cargas, podemos lograr la solución óptima (suponiendo que cada movimiento para la pendiente α es suficientemente pequeño). Repitiendo matemáticamente, el algoritmo de la corrección de errores en el aprendizaje en dos capas es calculado como sigue:

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \left[\tfrac{1}{2} \sum_{j=1}^{p} (b_{kj} - \sum_{i=1}^{n} a_{ki} w_{ij})^2 \right]$$

$$= (b_{kj} - \sum_{i=1}^{n} a_{ki} w_{ij})\ a_{ki}$$

$$= (b_{kj} - y_j) a_{ki} \qquad (1\text{-}34)$$

Aunque el valor de la función es solamente con respecto a un solo modelo, ha sido demostrado que el movimiento en la dirección opuesta de la pendiente para cada modelo, cuando toma el acumulado, hace un movimiento en la pendiente ruidoso en el que todavía se logra el apropiado resultado final.

El Perceptron y el **ADALINE**, son dos de las prominentes redes neuronales de las mas recientes, que emplean la corrección de errores en el aprendizaje. Incluyendo, el estado del cerebro en una caja que utiliza la corrección de errores en el aprendizaje en dos capas anteriormente descrito en un código de autoasociacion para una capa.

1.9.2.- CORRECCION DE ERRORES EN EL APRENDIZAJE EN MULTIPLES CAPAS.

Un problema que una vez atormentó en la corrección de errores en el aprendizaje es su incapacidad de extender el aprendizaje más allá de una red de dos capas. Con solamente una regla de aprendizaje para dos capas, solamente en trayectorias lineales puede ser conseguido. ¿Había habido varios ensayos para extender el algoritmo de la corrección de errores en el aprendizaje en dos capas a múltiples capas, pero el mismo problema sugiere observar: cuantos errores en cada capa oculta del **PE** son responsables del error de la capa de salida **PE** ?, utilizando la red neuronal de tres capas de la figura 1.8, para explicar el problema en múltiples capas (en este caso dado en tres capas), es calculada la cantidad de error en cada capa oculta del **PE**, "y_j", tiene que influir en el error de la capa de salida del **PE**.

Este problema, llamado problema de asignamiento de influencia[10], fue solucionado por la realización de una función diferencial continua PE para la capa oculta permitiendo la normalización de la cadena parcial de la diferencial a ser utilizada para calcular los cambios de las cargas para alguna red. Para la red de tres capas de la figura 1.8, el error de salida atravez de todo el $\mathbf{F_z}$ se a encontrado utilizando el valor de la función:

---

10 Barto A. (1984). "Simulation Experiments with Goal-Seeking Adaptive Elements", Air Force Wright Aeronautical Laboratory, Technical Report AFWAL-TR-84-1022.

$$E = \tfrac{1}{2} \sum_{j=1}^{p} (b_{kj} - z_j)^2 \qquad (1\text{-}35)$$

La salida del $\mathbf{F_z}$, z_j, es calculado utilizando la ecuación:

$$z_j = \sum_{i=1}^{p} y_i w_{ij} \qquad (1\text{-}36)$$

y cada $\mathbf{F_y}$ (la capa oculta), y_j es calculado al utilizar la ecuación:

$$y_i = f\left(\sum_{h=1}^{p} a_{kh} v_{hi}\right) = f(r_i);\ r_i = \sum_{h=1}^{n} a_{kn} v_{hi} \qquad (1\text{-}37)$$

La función de la capa oculta PE es:

$$f(\tau) = 1/\ 1 + e^{-\tau} \qquad (1\text{-}38)$$

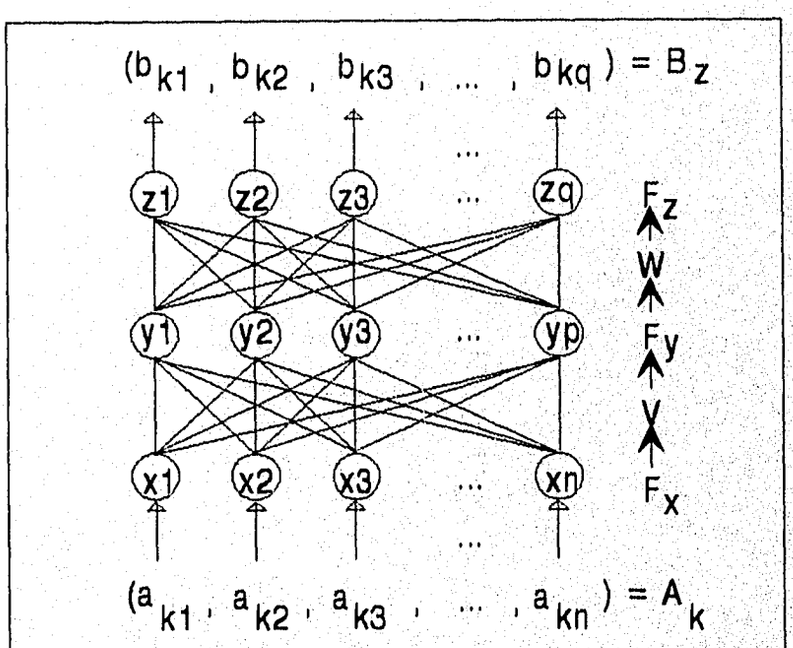


Figura 1.8 Corrección de errores en multiples capas.

Utilizando el mismo principio descrito en la sección anterior, hacemos los ajustes de las cargas por el movimiento en el valor de la función, en la dirección opuesta de la pendiente a un mínimo (donde el mínimo es considerado para la trayectoria de la entrada / salida que producirá una cantidad

pequeña del error total). Se ajustan las cargas de conexión entre el F_Y y F_Z utilizando la misma forma de la ecuación de la corrección de errores de aprendizaje en dos capas pero derivandola; produciendo por ello:

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} [\tfrac{1}{2} \sum_{j=1}^{q} (b_{kj} - z_j)^2]$$
$$= (b_{kj} - z_j) y_i = \delta_j y_i \qquad . \quad (1\text{-}39)$$

donde el valor positivo, de la constante de la velocidad de aprendizaje α ha sido agregado para ajustar la cantidad de cambios realizados con cada movimiento hacia bajo de la pendiente. Los siguientes, ajustes entre las cargas de conexión F_X y F_Y están fundamentadas por la utilización de la normalización de la parte diferencial de la cadena:

$$\frac{\partial E}{\partial v_{hi}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial \tau_i} \frac{\partial \tau_i}{\partial x_h} \frac{\partial x_h}{\partial v_{hi}}$$
$$= \sum_{i=1}^{p} (b_{ki} - y_i)\, y_i w_{hi}\, f'(r_i) a_{kh} \qquad (1\text{-}40)$$

donde **ß** es positivo, el valor de la constante de la velocidad de aprendizaje. La versión de este algoritmo para múltiples capas es comunmente referenciado como la regla de errores en el aprendizaje backpropagation o simplemente backpropagation. Utilizando la normalización de la cadena, podemos calcular los cambios de las cargas para un numero arbitrario de capas. El número de iteraciones que se tengan que hacer para cada modelo en los datos dados es grande, hacer esto en el algoritmo de aprendizaje fuera de linea es muy lento. De las ecuaciones (1-39) y (1-40), ajustando las cargas quedan como:

$$w_{ij}^{new} = w_{ij}^{old} - \alpha \frac{\partial E}{\partial w_{ij}} \qquad (1\text{-}41)$$

y

$$v_{hi}^{new} = v_{hi}^{old} - ß \frac{\partial E}{\partial v_{hi}} \qquad (1\text{-}42)$$

donde α y **ß** son constantes de valores positivos que regulan la cantidad de ajustes realizados con cada movimiento en la pendiente.
Extendiendo la backpropagation a la utilización de conexiones de variación media entre el $\mathbf{F_X}$ y $\mathbf{F_Y}$. La figura 1.9 muestra una red de tres capas con variación media que es una versión del algoritmo de la corrección de errores en el aprendizaje en múltiples capas.

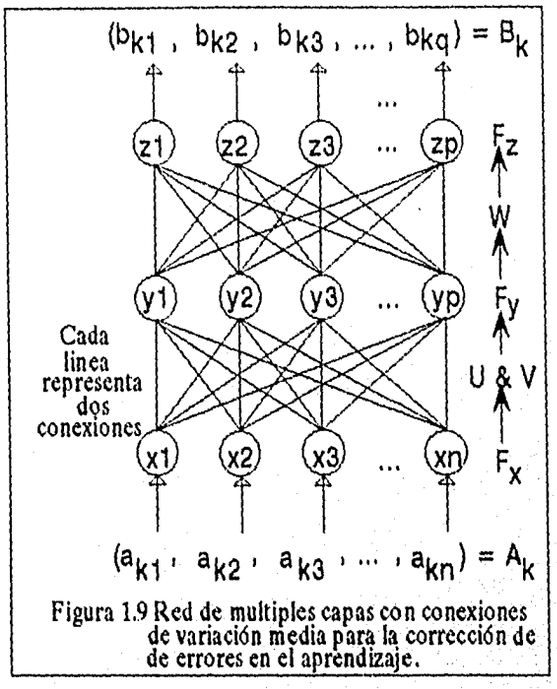


Figura 1.9 Red de multiples capas con conexiones de variación media para la corrección de de errores en el aprendizaje.

El valor de la capa oculta $\mathbf{F_y}$ es calculado con la ecuación:

$$y_i = g(r_i);\ r_i = \sum_{h=1}^{n} (\mu_{hi} - a_{kh} / v_{hi})^2 \qquad (1\text{-}43)$$

donde μ_{hi} representa la fuerza media de conexión entre los elementos **h** de $\mathbf{F_x}$ y los elementos **i** de $\mathbf{F_y}$, v_{hi} es la variación de la fuerza de conexión entre los elementos **h** del $\mathbf{F_x}$ y los elementos **i** del $\mathbf{F_y}$ y la función **PE** es la función Gaussiana.

$$g(x) = e^{-x/2} \qquad (1\text{-}44)$$

Los valores de salida del $\mathbf{F_z}$, se forma entonces de la combinación lineal de la capa oculta Gaussiana utilizando la ecuación:

$$z_j = \sum_{i=1}^{p} y_i w_{ij} \qquad (1\text{-}45)$$

donde w_{ij} es la fuerza de conexión entre los elementos **i** del $\mathbf{F_y}$ y los elementos **j** del $\mathbf{F_z}$. El calculo de la pendiente para cada colección de cargas produce las siguientes ecuaciones:

$$\frac{\partial E}{\partial \mu_{hi}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_i}\frac{\partial y_i}{\partial r_i}\frac{\partial r_i}{\partial \mu_{hi}}$$

$$= \sum_{j=1}^{q} (b_{kj} z_j)\, w_{ij}\, g'(r_i)\ (\mu_{hi} - a_{ki} / v_{hi}^2) \qquad (1\text{-}46)$$

$$\frac{\partial E}{\partial v_{hi}} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial y_i} \frac{\partial y_i}{\partial r_i} \frac{\partial r_i}{\partial v_{hi}}$$

$$= \sum_{j=1}^{q} (b_{kj} - z_j)\, w_{ij}\, g'(r_i)\, (\mu_{hi} - a_{ki} / v_{hi}^3) \quad (1\text{-}47)$$

$$\frac{\partial E}{\partial w_{ij}} = (b_{kj} - z_j)\, y_i \quad (1\text{-}48)$$

De estas ecuaciones, se encuentran las ecuaciones actualizadas que son:

$$\mu_{hi}^{new} = \mu_{hi}^{old} - \Sigma \frac{\partial E}{\partial \mu_{hi}} \quad (1\text{-}49)$$

$$v_{hi}^{new} = v_{hi}^{old} - \beta \frac{\partial E}{\partial v_{hi}} \quad (1\text{-}50)$$

$$w_{ij}^{new} = w_{ij}^{old} - \tau \frac{\partial E}{\partial w_{ij}} \quad (1\text{-}51)$$

donde α, β y τ son constantes diferentes de cero.

El algoritmo backpropagation fue introducido por Werbos P. y redescuvierto independientemente por Parker D. y Rumelhart R. y Williams R.. El algoritmo presentado aquí es una versión simple. Hay varias variaciones en el algoritmo, incluyendo topologías alternativas de capas múltiples, métodos para optimizar el numero de capas ocultas y el número de PE en cada capa oculta y muchos más[11]. Aunque muchas salidas quedan sin resolver con el procedimiento de corrección de errores en el backpropagation, tal como un número apropiado de parámetros del aprendizaje, la existencia de un mínimo local durante el aprendizaje, el aprendizaje es extremamente largo en tiempo, el número óptimo y la configuración de capas ocultas, la habilidad de este método de aprendizaje que captura automática en un diseño no lineal, del resto tiene una fuerza significativa.

---

11 Simpson P. (1990), Artificial Neural Systems:Foundations, Paradigms, Applications and Implementation, Elmsford, NY:Pergamon Press.

1.10.- REFUERZO DEL APRENDIZAJE.

La idea inicial para el refuerzo del aprendizaje la introdujo Widrow B., N. K. Gupta, y S. Maitra. El Refuerzo del aprendizaje es semejante a la corrección de errores en el aprendizaje en aquellas cargas que son reforzadas para realizar las acciones apropiadas y pegandolas a una mala acción realizada. La diferencia entre estas dos técnicas de supervisión del aprendizaje es que la corrección de errores en el aprendizaje utiliza más información específica del error para reunir los valores de los errores de cada capa oculta del PE, mientras el refuerzo del aprendizaje no utiliza información especifica del error para determinar la acción de la red. Mientras la corrección de errores en el aprendizaje tiene un vector de los valores totales únicos que utiliza para la corrección de errores solamente un valor es utilizado que descrive la acción de la capa de salida durante el reforzamiento del aprendizaje. Esta forma de aprendizaje es ideal en situaciones donde la información especifica del error no esta disponible, pero es la información completa de la acción, es tal como la predicción y el control.
Una red neuronal de dos capas tal como la de la figura 1.10 sirve como una buena estructura para el algoritmo del refuerzo del aprendizaje (aunque también en redes de múltiples capas puede utilizarse el refuerzo del aprendizaje).

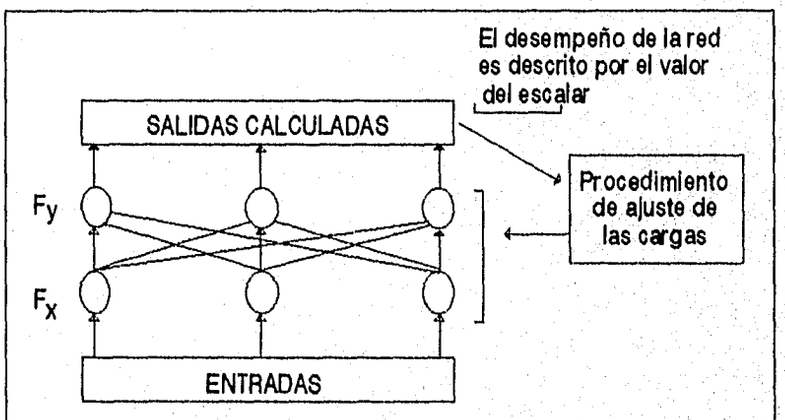


Figura 1.10 Ejemplo de refuerzo del aprendizaje.

La ecuación del refuerzo del aprendizaje es:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha\,(\tau - \theta_j)\, e_{ij} \qquad (1\text{-}52)$$

donde τ es el valor del escalar del éxito o fracaso proporcionado por el ambiente, θ_j es el valor del reforzamiento del **PE** para los elementos **j** del F_Y, e_{ij} es la elegibilidad canónica de la carga de los elementos **i** del F_X

a los elementos j del F_y y $0 < \alpha < 1$ es un valor constante de la velocidad del aprendizaje. En la corrección de errores en el aprendizaje, se hace una pendiente descendente en el espacio del error. El Refuerzo del aprendizaje hace una pendiente descendente en el probable espacio. La elegibilidad canónica de w_{ij} es dependiente de una previa selección de la distribución probable que sera utilizada para determinar si el valor de salida calculado es igual al valor de salida deseado y es definido como:

$$e_{ij} = \frac{\partial}{\partial w_{ij}} \ln g_i \qquad (1\text{-}53)$$

donde g_i es la probabilidad de la igualdad de la salida deseada y la salida calculada, definida como:

$$g_i = P_r (y_j = b_{kj} \, ^3 \, W_j, A_k) \qquad (1\text{-}54)$$

que es leido como la probabilidad de que y_j sea igual a b_{kj} dada la entrada $\mathbf{A_k}$ y el correspondiente vector de cargas $\mathbf{W_j}$. Redes nueronales que emplean el refuerzo del aprendizaje incluyen al adaptable crítico heurístico y la red asociativa con penalización de ganancia.

1.11.- APRENDIZAJE STOCHASTIC.

El aprendizaje Stochastic utiliza procesos aleatorios, probabilisticos y una relación de ajuste de energía de las cargas de conexión en una red neuronal de múltiples capas. Para la red neuronal de tres capas de la figura 1.8, se describe el procedimiento stochastic como sigue:

1.- Cambios aleatorios de los valores de salida de una capa oculta **PE** (la capa oculta **PE** utiliza una función de escalon binaria).

2.- Evaluación del cambio utilizando la diferencia resultante en la energía de la red neuronal como una guía. Si la energía después del cambio es baja se mantiene el cambio. Si el cambio en la energía no es bajo después del cambio aleatorio, acepta el cambio según una preeleccion de la distribución probabilistica.

3.- Después de varios cambios aleatorios, la red eventualmente se hará "estable". Reuniendo los valores de la capa oculta PE y la capa de salida.

4.- Repita los pasos 1 al 3 para cada par de modelos dados en los datos; entonces utilice los valores reunidos estadisticamente para ajustar las cargas.

5.- Repita los pasos 1 al 4 hasta que la ejecución de la red sea adecuada.

La probabilidad de aceptación del estado de energía más alto, a pesar de un incremento temporal de energía, permite a la red neuronal que escape un mínimo de energía local en favor de una energía mínima mas grande. Este proceso de aprendizaje, se fundamenta en simular un reconocimiento, es regido por un parámetro de temperatura que disminuye lentamente el numero de probabilidad de aceptar un estado de energía mas alto. La maquina Boltzmann fue la primera red neuronal que empleo el aprendizaje stochastic. Szu H. tiene una depuración del procedimiento utilizando la función de distribución Cauchy en lugar de la función de distribución Gaussian, que resulta en una red que converge a una solución mucho más rápida.

1.12.- SISTEMAS DE HARDWIRED.

Unas redes neuronales tienen sus cargas de conexión predeterminadas para un problema específico. Estas cargas son el Hardwired y no cambia una vez que ha sido determinado. La mayor parte de los populares sistemas de Hardwired son para la optimizacion de redes neuronales. Los trabajos de optimizacion de redes para construir un valor de la función es cuando se minimiza la solución aun espontaneo problema de optimizacion. Para trasladar la energía de la función en las cargas dadas y valores parciales, la red neuronal realiza una optimizacion en paralelo. Dados los valores iniciales del problema, la red correra a una solución estable. Esta técnica ha sido aplicada a una extensión ancha de problemas (Simpson, 1990a), incluso para inventarios, rutas, y optimizacion de recursos.

Otros dos tipos de Harwired de redes incluyen el filtro avalancha y la red neuronal probabilistica[12]. Se consideran a estos sistemas como Hardwired de redes porque los modelos de datos son normalizados a unidades largas y utilizados como cargas de conexión. A pesar de la falta de un procedimiento adaptable del aprendizaje, cada una de estas redes neuronales son muy poderosas en sus propios derechos.

1.13.- RESUMEN DE PROCEDIMIENTOS DE APRENDIZAJE.

Varios atributos de cada algoritmo de aprendizaje en redes neuronales han sido descritos. La tabla 1 describe seis atributos claves de los procedimientos de aprendizaje discutidos:

---

12 Specht D. (1990), "Probabilistic Neural Networks", Neural Networks, Vol. 3., pp. 109-118.

1.- Tiempo de Aprendizaje. Cuánto tiempo lleva realizar la técnica de aprendizaje capturando la información adecuadamente (rápido, lento, muy lento, o extremamente lento)

2.- En línea / Fuera de Linea. Es la técnica de aprendizaje con un algoritmo, en linea ó fuera de linea.

3.- Supervisado / No Supervisado. Es la técnica de aprendizaje con un procedimiento supervisado o no supervisado.

4.- Lineal / No Lineal. La técnica de aprendizaje capas de capturar trayectorias no lineales.

5.- Estructural / Temporal. El algoritmo de aprendizaje en que se puede capturar la información en forma estructural, temporal o ambas.

6.- Capacidad de Almacenaje. Es la capacidad de un buen almacenaje de información relativo al número de conexiones en la red.

La información proporcionada en la tabla 1 es como una guía y no esta pensada como una descripción precisa de las cualidades de cada red neuronal. Para una descripción mas detallada de cada algoritmo de aprendizaje para redes neuronales, a Simpson P., Hecht-Nielsen R., o Maren A., C. Harsten, y R. Pap.

ALGORITMO DE APRENDIZAJE	TIEMPO DE APRENDIZAJE	EN-LINEA FUERA DE LINEA	SUPERVISADO NO SUPERVISADO	LINEAL NO LINEAL	ESTRUCTURAL TEMPORAL	CAPACIDAD DE ALMACENAJE
APRENDIZAJE HEBBIAN	RAPIDO	EN-LINEA	NO SUPERVISADO	LINEAL	ESTRUCTURAL	MALO
COMPONENTE PRINCIPAL DEL APRENDIZAJE	LENTO	FUERA DE LINEA	NO SUPERVISADO	LINEAL	ESTRUCTURAL	BUENO
DIFERENCIAL DE L APRENDIZAJE HEBBIAN	RAPIDO	EN-LINEA	NO SUPERVISADO	LINEAL	TEMPORAL	NO DETERMINADO
APRENDIZAJE COMPETIDO	LENTO	EN-LINEA	NO SUPERVISADO	LINEAL	ESTRUCTURAL	BUENO
APRENDIZAJE MAXIMO-MINIMO	RAPIDO	EN-LINEA	SUPERVISADO	NO LINEAL	ESTRUCTURAL	BUENO
CORRECCION DE ERRORES EN EL APRENDIZAJE EN DOS CAPAS	LENTO	FUERA DE LINEA	SUPERVISADO	LINEAL	AMBOS	BUENO
CORRECCION DE ERRORES EN EL APRENDIZAJE EN MULTIPLES CAPAS	MUY LENTO	FUERA DE LINEA	SUPERVISADO	NO LINEAL	AMBOS	MUY BUENO
REFORZAMIENTO DEL APRENDIZAJE	EXTREMADAMENTE LE	FUERA DE LINEA	SUPERVISADO	NO LINEAL	AMBOS	BUENO
APRENDIZAJE STOCHASTIC	EXTREMADAMENTE LE	FUERA DE LINEA	SUPERVISADO	NO LINEAL	ESTRUCTURAL	BUENO
SISTEMAS DE HARDWIRED	RAPIDO	FUERA DE LINEA	SUPERVISADO	NO LINEAL	ESTRUCTURAL	BUENO

1.14.- EL RECORDAR EN REDES NEURONALES.

En la sección anterior se acentuó el almacenaje de información por una ancha extensión de procedimientos de aprendizaje. En esta sección, el énfasis está en recobrar la información ya guardada en la red. Algunas de las ecuaciones del recuerdo ha sido introducir como una parte de los procesos de aprendizaje. Otras serán introducidas aquí por primer vez. Las técnicas del recuerdo descritas aquí caen en dos categorías anchas: Recuedo sin realimentación y Recuerdo con realimentación.

1.14.1.- RECUERDO SIN REALIMENTACION.

El recuerdo sin realimentación es realizado en redes que no tiene conexiones de realimentacion. En la mayoría de técnicas comunes del recordamiento sin realimentación es la combinación lineal seguida por una función **PE**.

$$y_i = f\left(\sum_{j=1}^{n} x_i w_{ij}\right) \qquad (1\text{-}55)$$

donde la función **f** es una de las descritas en la sección 1.2. Para una red con alimentación controlada que utiliza conexiones dobles, donde una colección de cargas de conexión **W** representa la media y la otra colección de cargas de conexión **V** representan la variación, la ecuación del recuerdo es:

$$y_j = g\left(\sum_{i=1}^{n} (w_{ij} - x_i / v_{ij})^2\right) \qquad (1\text{-}56)$$

donde **g** es la función Gaussian **PE**.

Para una red sin realimentación que utiliza conexiones dobles donde dada la carga de conexión **V** que representa el mínimo vector y la otra carga de conexión **W** representa el máximo vector y se confina el sistema a la unidad hipercubica, hay dos posible ecuaciones del recuerdo: la primera es la ecuación base del "producto de complementos" (Simpson, 1991b).

$$y_j = \left[\, 1 - \frac{1}{n} \sum_{i=1}^{n} \max\left(0, \min\left(1, \tau\left(v_{ji} - x_i\right)\right)\right) \,\right]$$

$$x \left[\, 1 - \frac{1}{n} \sum_{i=1}^{n} \max\left(0, \min\left(1, \tau\left(x_i - w_{ij}\right)\right)\right) \,\right] \qquad (1\text{-}57)$$

y la otro es la de productos relativos (Simpson, 1992).

$$y_j = \frac{1}{2n} \sum_{i=1}^{n} \left[\max\left(0, 1 - \max\left(0, \tau \min\left(1, x_i - w_{ij}\right)\right)\right) \right.$$

$$\left. + \max\left(0, 1 - \max\left(0, \tau \min\left(1, v_{ji} - x_i\right)\right)\right) \right] \qquad (1\text{-}58)$$

donde x_i es el valor de la capa de entrada $\mathbf{F_X}$, τ es el valor que regula la sensitividad de la calidad de la función, y y_j es el valor de salida de los elementos **j** del $\mathbf{F_Y}$ referente a la figura 1.4, (1-57) mide el grado a que el modelo de entrada $\mathbf{A_k}$ cae entre los vectores máximos y mínimos de la clase **j**, donde un valor medio de 1 de $\mathbf{A_k}$ cae completamente

entre V_j y W_j si el más cercano y_j esta en 0 existe la más grande disparidad entre A_k y la clase **j**, con un valor de 0 significa que A_k esta completamente fuera de la clase. Note que hay muchas otras posibles funciones que puede ser utilizadas aquí. También note que la relación entre la red neuronal y colecciones distorsionadas es realizado cuando cada PE es visto como una colección distorsionada separada.

1.14.2.- RECUERDO CON REALIMENTACION.

Son redes que tienen conexiones de realimentacion empleando una ecuación de recuerdo con realimentacion de la forma:

$$x_j\ (t+1) = (1-\alpha)\ x_j(t) + \beta \sum_{i=1}^{n} f(x_i(t))\ w_{ij} + a_{ki} \quad (1\text{-}59)$$

donde $x_j\ (t+1)$ es el valor de los elementos **j** en una red neuronal de una simple capa a tiempo **t + 1**, **f** es una función monotonica no decreciente (función sigmoid), α es una constante positiva que regula la cantidad de decrecimiento del valor del PE que tiene durante un intervalo de tiempo unitario, β es una constante positiva que regula la cantidad de realimentacion, que los otros PE proporcionan a los elementos **j** y a_{ki} es el valor constante de entrada de los componentes **i** del modelo de entrada **k**. Una cuestión que surge en sistemas de recuerdo con realimentacion es la estabilidad. Se logra estabilidad cuando en una red cesan de cambiar los valor anteriores que han sido dados en una colección inicial de entradas A_k y tiene que procesarlos por un rato. Si la red no es estable, no sera muy utilizada. Idealmente, las entradas iniciales a la red neuronal con realimentacion representan el modelo de entrada y el estado estable que la red alcanza representa la proxima salida del sistema. Un importante teorema fue presentado por Cohen y Grossberg en (1983), que comprobó esto, para una clase ancha de redes neuronales bajo una mínima colección de fuerzas, la red se haría estable en un período finito de tiempo para algunas condiciones iniciales. Este teorema tratado en sistemas que tienen cargas fijas, es una extensión al teorema Cohen-Grossberg, Kosko B. mostro que una red puede aprender y recordar al mismo tiempo y sin embargo permanece estable[13].

13 Kosko B. (1990), "Unsupervised Learnin in Noise", IEEE Trans. Neural Networks, Vol. 1, pp. 44-57, Mar.

1.14.3.- RESPUESTAS DE INTERPOLACION VERSUS RESPUESTAS PROXIMAS/CONTIGUAS.

Además de recordar la existencia de operaciones sin realimentación o realimentada, hay otro atributo importante asociado con el recordar, a saber respuesta de salida. Hay dos tipos de respuestas de salida en redes neuronales: proximas ó contiguas y de Interpolación la figura 1.11 ilustra la diferencia. Suponga que la tres caras dispuestas en pares en la figura 1.11(a) han sido guardadas en una red. Si una entrada que es una combinación de dos de las caras es presentado a la red, hay dos caminos que la red puede seguir. Si la salida es una combinación de las dos salidas correctas asociadas con las entradas dadas, entonces la red ha hecho una interpolación (figura 1.11(b)). De lo contrario, la red puede determinar cual de las caras guardadas es la mas cercana asociada con la entrada y responde con la salida asociada para esa cara (figura 1.11(c)). Las redes neuronales sin realimentacion, de modelos de maquinas son típicamente redes de respuesta interpolada (la backpropagation y la lineal memoria asociativa). Las redes sin realimentación de clasificación de modelo (cuantificación del vector de aprendizaje) y las redes de modelos de maquinas con realimentacion (la red Hopfield y la memoria asociativa bidireccional) son típicamente redes de respuesta proximas ó contiguas.

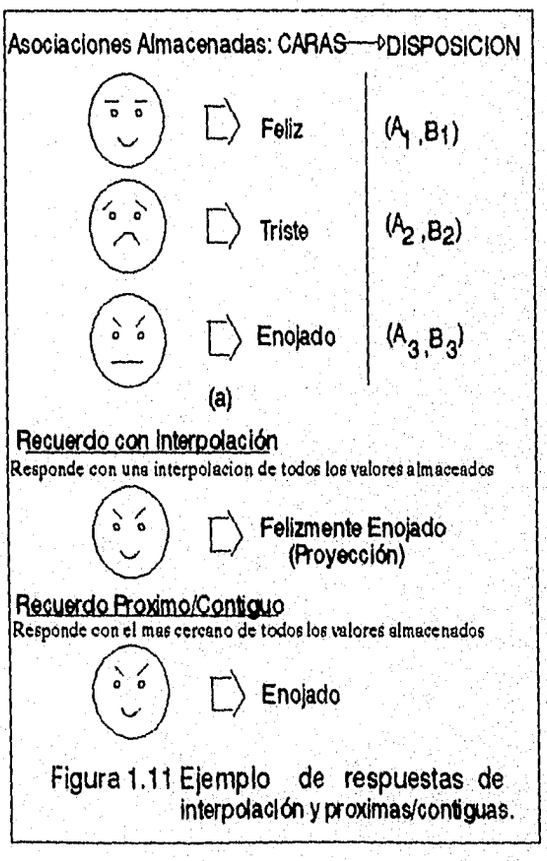


Figura 1.11 Ejemplo de respuestas de interpolación y proximas/contiguas.

1.15.- SISTEMAS DINAMICOS NEURONALES.

Describimos los sistemas dinámicos neuronales por un sistema diferencial de primer orden o ecuaciones diferenciales que dirigen las evoluciones en el tiempo, de las activaciones neuronales o potencial de la membrana. Las diferentes ecuaciones diferenciales que gobiernan los sistemas dinámicos sinapticos. Para los campos F_X y F_Y, denotamos las ecuaciones diferenciales de activación como:

$$x_1 = g_1(F_X, F_Y, \ldots) \tag{1-60}$$

$$\vdots$$

$$x_n = g_n(F_X, F_Y, \ldots) \tag{1-61}$$

$$y_i = h_1(F_X, F_Y, \ldots) \tag{1-62}$$

$$\vdots$$

$$y_p = h_p(F_X, F_Y, \ldots) \tag{1-63}$$

o, en notación vectorial,

$$x = g\ (F_X, F_Y, \ldots) \tag{1-64}$$
$$y = h\ (F_X, F_Y, \ldots) \tag{1-65}$$

donde x_i y y_j denotan respectivamente el tiempo de activación en las funciones de las **i** neuronas en $\mathbf{F_X}$ y las **j** neuronas en $\mathbf{F_Y}$. Los razonamientos de las funciones $\mathbf{g_i}$ y $\mathbf{h_j}$ también incluyen información sinaptica y de entrada. No incluimos el tiempo como una variable independiente. Como un resultado, en la teoría de sistemas dinámicos, los modelos de redes neuronales se clasifican como sistemas dinámicos autónomos. Los sistemas dinámicos no autónomos pueden permitir el cambio en la activación x_i dependiendo, dicen, de adicionarle t^2. Sistemas autónomos son usualmente más fáciles de analizar que los sistemas no autónomos. El tiempo juega un papel especial en la dinámica neuronal: el tiempo es "rápido" a nivel neuronal. En sistemas neuronales de mamíferos, las fluctuaciones de la membrana ocurren al nivel de milisegundos. En Hardware o inplementaciones computacionales de redes neuronales, las fluctuaciones neuronales pueden en principio ocurrir al nivel de nanosegundos, En contraste, el tiempo es "lento" al nivel sinaptico. En sistemas neuronales de mamíferos, las fluctuaciones sinapticas ocurren al nivel de segundos o minutos. Pensamos más rápido de lo que aprendemos. Note la ausencia del tiempo en la derivada de segundo orden (aceleración) y la derivada parcial en (1-60) a (1-65). Esto representa una distinción entre modelos de redes neuronales y los modelos clásicos "modelos neuronales", donde a menudo las ecuaciones diferenciales dan un mayor detalle, las multivariables describen la forma de neuronas individuales o se comportan como la sinapsis.

1.15.1.- DECLARACION DEL ESPACIO NEURONAL.

Definimos el espacio de los sistemas dinámicos neuronales a tiempo t con los instantáneos vectores de activación:

$$X(t) = (x_1(t),\ldots,x_n(t)) \qquad (1\text{-}66)$$
$$Y(t) = (y_1(t),\ldots,y_p(t)) \qquad (1\text{-}67)$$

Por comodidad, a veces identificaremos los grupos topologicos o campos $\mathbf{F_X}$ y $\mathbf{F_Y}$ con la declaración de sus respectivos vectores de activación **X** y **Y**. El declarar el espacio del vector para el campo $\mathbf{F_X}$ en el espacio real extendido $\mathbf{R^n}$ (añadiendo infinito positivo y negativo). Semejantemente $\mathbf{R^p}$ es el espacio declarado de $\mathbf{F_Y}$. La declaración total del espacio o juntar el sistema dinámico neuronal es el producto del espacio $\mathbf{R^n}$ **x** $\mathbf{R^p}$. Un punto especifico en el espacio declarado es una foto de toda la conducta neuronal. Un índice del tiempo en la curva define una trayectoria en el espacio declarado. Una trayectoria describe el tiempo de evolución de las activaciones en la red. Podemos concatenar los campos $\mathbf{F_X}$ y $\mathbf{F_Y}$ en un solo campo más grande $\mathbf{F_Z} = [\mathbf{F_X}\ ^3\ \mathbf{F_Y}]$, el campo aumentado. Entonces se declara el espacio en la red que corresponde a la dimensión mas alta del vector en el espacio $\mathbf{R^{n+p}}$. En la terminología de Kohonen [1988], este aumento convierte una red heteroasociativa en una red autoasociativa. Tenemos que usar esta formal equivalencia con cuidado. Justamente como los números 4 y 1+1+1+1 son formalmente equivalentes pero tienen diferentes propiedades de calculo, la concatenación de campos puede tener diferentes cálculos, métricos u otras propiedades. Por ejemplo, métricos o clasificaciones cercanas-proximas a menudo degradan en dimensiones más altas. (Ejemplo del efecto dependiente de la dimensión: el volumen de la unidad Euclidiana hiperesferica crece arriba de la quinta dimensión, disminuye para dimensiónes más altas), cuando los campo $\mathbf{F_X}$ y $\mathbf{F_Y}$ poseen otras estructuras topologicas de conectividad, su concatenación se hace aún menos plausible.

1.15.2.- SEÑALES QUE DECLARAN EL ESPACIO COMO HIPERCUBICO.

Observemos la señal que declaramos **S(x)** del campo $\mathbf{F_X}$ a tiempo **t**. **S(x)** denota las señales de **n** vectores emitidos por las neuronas en $\mathbf{F_X}$.

$$S(x(t)) = (S_1^X(x_1(t)),\ldots, S_n^X(x_n(t))) \qquad (1\text{-}68)$$

S_i^X denota la señal de la función de las **i** neuronas en el campo $\mathbf{F_X}$. Diferentes neuronas puede tener diferentes características de señales no lineales. Para una notación mas sencilla omitimos el índice superior en la identificación del

campo. Por lo tanto $S_i(x_i)$ y $S_j(y_j)$ denotan respectivamente las señales de las funciones de las i neuronas en el campo F_X y las señales de las funciones de las j neuronas en el campo F_Y. La señal que declara el espacio de F_X consiste de todas las señal declaradas. El limite de la señal significa que la señal declara el espacio en una dimensión n hipercubica. Anderson [1983] se refiere a este como "el cerebro declarado en una caja". En el especial pero caso común donde la señal lleva valores en un intervalo unitario de [0,1], la señal que declara el espacio es igual a la unidad hipercubica I^n o $[0,1]^n$. En general nosotros podemos adaptar y traducir la señal para que la unidad hipercubica sea igual a la señal que declara el espacio de la red. Entonces el cubo del producto $I^n \times I^p$ define la señal que declara el espacio de los dos campo en la red $\{F_X, F_Y\}$ y I^{n+p} define la señal que declara el espacio de la concatenación de los campos en la red $[F_X \ ^3 F_Y]$. En redes neuronales estables con realimentacion. El punto de equilibrio tiende a ocurrir cerca del vértice 2^n del cubo. La señal monotonica de la función produce un extremo efecto que tiende a amplificar la realimentacion. Hopfield y Tank [1985] explotaron esta propiedad cuando construyeron redes para solucionar problemas de optimización, en particular en problemas de vendedores de viajes. En los capítulos 2 y 3 del volumen de [Kosko, 1991] aplica esta técnica de optimización en el procesamiento de imágenes. La unidad hipecubica I^n y por lo tanto la señal binaria declara el espacio, también define el poder del fuzzy colocando $F(2^n)$ en los n elementos, el colocar todos los subconjuntos fuzzy del conjunto $X = \{x_1,\ldots,x_n\}$ de n elementos x_i, un multivalor o colocar un fuzzy A correspondiente a un punto en I^n. Las i coordenadas de A indican el grado a que el elemento x_i pertenece a la colección A, la capacidad del valor de x_i (unidad fuzzy). Tanto en el usual subconjunto 2^n de X que corresponde al vértice 2^n del cubo de la unidad I^n. Por lo tanto geométricamente muchas neuronas y cálculos fuzzy coinciden. Una aplicación común de este isomorfismo es la detección de la interpretación de las características del valor de la señal $S(x_i(t))$ como el grado de la calidad del elemento x_i en las características dadas en A a tiempo t, o el grado a que un modelo probado de entrada pertenece a la clase "i" del modelo. Colecciones infinitas, tales como R^n, también contienen subconjuntos fuzzy la base de la función radial define un campo receptivo esférico un subconjunto fuzzy de R^n.

1.15.3.- ACTIVACIONES NEURONALES EN EL TERMINO DE MEMORIA CORTA.

El término de memoria corta (**SMT**) en modelos de redes neuronales como la declaración de los vectores de activación **X** y **Y**. Los modelos o la forma de la onda es persistente, sin embargo breve, entre las activaciones que representan acontecimientos (**SMT**), fenómenos puramente psicológicos. Usualmente olvidamos en materia de segundos un nombre nuevo o numero de teléfono cuando nos encontramos a varias personas en una convención o fiesta. Esto provee informales evidencias de que los modelos de activación compiten para el almacenaje en **SMT**. En la sinapsis se codifica en el termino de memoria larga (**LTM**) la información de modelos. Otra conjetura interesante en redes neuronales es el dormir. El sueño puede permitir "ganar" **SMT** a los modelos de activación que son cargados bajamente en **LTM**. Altos niveles (hormonales) al despertar pueden favorecer concurrentes modelos de activación **SMT** en la codificación de procesos competidos: Al levantarse, aumenta el aprendizaje. La alta sensibilidad de nuestro día invade nuestros sueños de noche. Fisiología y psicología, carne y espíritu, juntas cuando identificamos modelos neuronales de activación con **SMT**. El potencial de la membrana provee una versión moderna de la glándula pineal de Descartes, que el naturalista del siglo diecisiete creyó juntar mente y cuerpo. Podemos también identificar activaciones neuronales con la modalidad de sentidos. Los modelos de activación pueden representar modelos auditivos, olfatorio, táctiles o visuales. O pueden representar la velocidad de contracción de los músculo en el sistema motor. Esta generalidad figurativa es unificadora y refuerza la teoría de redes neuronales. Los campo neuronales de activación, el potencial de la membrana, componen el lenguaje común del cerebro, un lenguaje adaptable que integra el sistema sensorial, motor y actividades cognoscitivas. Las fluctuaciones de campos neuronales pueden proveer el fenómeno de la dinámica en el pensamiento.

Capítulo 2

Arquitecturas y Modelos de Redes Neuronales Artificiales

INTRODUCCION.

La clasificación de bloques para la construcción de redes neuronales es conocido como la topología, la cual se desarrolla de los vectores, conexiones y funciones **PE** (elementos de proceso) que han sido descritos, las capas de **PEs** son interconectadas por las conexiones de las cargas. Después de introducir la terminología, se describirán seis comunes topologías de redes neuronales.

2.1.- TERMINOLOGIA.

2.1.1.- CAPAS.

Las redes neuronales son organizadas en capas de **PEs**. Dentro una capa, los **PEs** son semejante en dos cosas: 1). Las conexiones que alimenta la capa de **PEs** son de la misma fuente: por ejemplo, el **PE** en la capa $\mathbf{F_x}$ de la Figura 2.1, todas reciben sus entradas del vector de entrada y el **PE** de la capa $\mathbf{F_y}$ recibe sus entradas desde el $\mathbf{F_x}$. 2) El **PE** en cada capa utiliza la misma dinámica de actualización, por ejemplo, todos los **PE** utilizan el mismo tipo de valores de conexiones en el centro y fuera del centro.

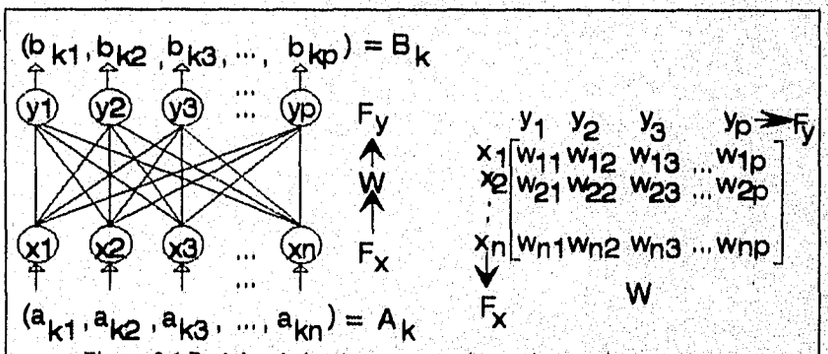


Figura 2.1 Red donde las dos capas están totalmente interconectadas y su matriz de cargas.

2.1.2.- CONEXIONES INTRACAPAS VERSUS INTERCAPAS.

Hay dos tipos de conexiones que una red neuronal emplea: las conexiones de Intracapas y conexiones de Intercapas. Las conexiones Intracapas ("intra" del latín "dentro"), son conexiones entre **PEs** en la misma capa. Conexiones intercapas ("inter" del latín "entre"), son conexiones entre **PEs** en capas diferentes. Es posible tener redes neuronales que consisten de uno o ambos, tipos de conexiones.

2.1.3.- REDES CON REALIMENTACION VERSUS REDES SIN REALIMENTACION.

Cuando una red tiene conexiones que alimentan la información solamente en una dirección (entrada a salida), sin alguna ruta de realimentación, es una red sin realimentación (o lineal). Si la red tiene alguna ruta de realimentación, donde se define la realimentación como alguna ruta por la que la red permite que un mismo **PE** sea visitado dos veces, entonces es una red con realimentación.

2.2.- ENTRADA EN ESTRELLA, SALIDA EN ESTRELLA Y EL ADALINE.

Las dos redes sencillas son la entrada en estrella y la salida en estrella. La entrada en estrella (figura 2.2(a)), es la mínima codificación para esta red, es un ejemplo sencillo de un procedimiento de codificación para la entrada en estrella que llevaría al vector $A_k = (a_{k1}, a_{k2}, \ldots, a_{kn})$, a la normalización utilizando los valores de la carga $W_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$, como muestra la ecuación:

$$v_{ij} = a_{ki} \Big/ \sum_{i=1}^{n} a_{ki} \qquad (2\text{-}1)$$

para toda $i = 1, 2, \ldots, n$.

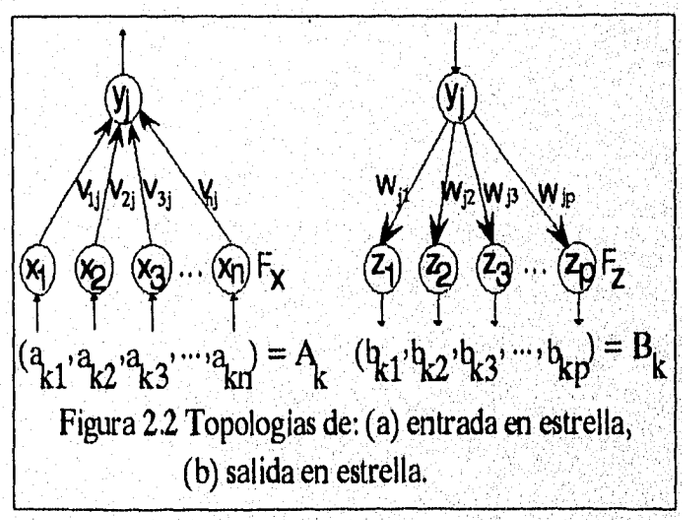


Figura 2.2 Topologías de: (a) entrada en estrella, (b) salida en estrella.

Lo contrario de la entrada en estrella es la salida en estrella (figura 2.2(b)). La salida en estrella, es el diseño del recuerdo para la red. Un vector de la salida en estrella es generado utilizando la ecuación:

$$z_i = y_j \quad w_{ji} \qquad (2\text{-}2)$$

para toda **i** = 1, 2,..., p, donde las cargas son determinadas de (2-1) o de algún algoritmo de aprendizaje descrito en el capitulo 1. El ADALINE[1] tiene la mismo topología que la entrada en estrella (figura 2.2(a)), pero las cargas V_j son ajustadas al utilizar el algoritmo de la mínima media al cuadrado (**LMS**). En la estructura de la adaptación de la señal procesada, una topología semejante con la misma funcionalidad a que es referida como el filtro de la respuesta finita a un impulso (**FIR**). Las aplicaciones del filtro (**FIR**) son en la cancelación del ruido, cancelación del eco, adaptación de antenas y en control son numerosas.

2.3.- REDES DE CAPAS SIMPLES: AUTOASOCIACION, OPTIMIZACION Y CONTRASTE EN EL INCREMENTO.

Más allá de las redes de entrada en estrella, salida en estrella, las mínimas redes neuronales son las redes de intraconexiones de capas simples. La figura 2.3 muestra la topología de una red neuronal de una capa que consiste de **n** F_X.

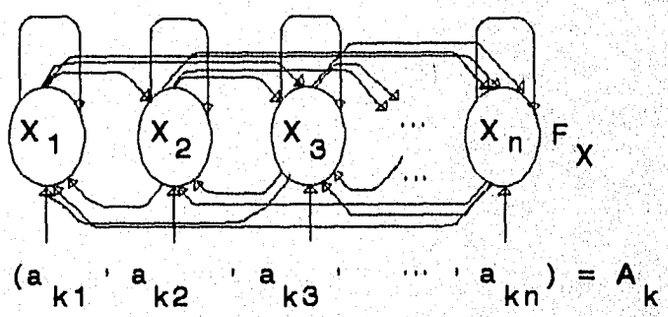


Figura 2.3 Red de una capa intraconectada.

Las conexiones son de cada F_X a otras F_X, que producen una matriz de conexión con n^2 entradas. La red neuronal de capas simples acepta un vector de entrada de dimensión **n** por uno de tres modos:

---

1 Widrow B. y M. Hoff (1960), "Adaptive Switching Circuits" in 1960 WESCON Convention Record: Part IV, pp. 96-104.

1.- Solo el **PE** de inicialización. El vector de entrada es utilizado para inicializar el F_X del **PE** y el vector de entrada no influye después en el proceso.

2.- El **PE** de inicialización y su influencia es constante. El vector de entrada es utilizado para inicializar el F_X y la entrada queda como un valor constante, que influye en todo el proceso.

3.- Solamente la influencia es constante. Los **PEs** son inicializados todos con cero y los vectores de entrada tienen un valor contante que influye en todo el proceso.

Las redes de una capa son utilizadas para desempeñar cuatro tipos de procesos: terminación de un vector, alejamiento de ruido, optimización y contraste en el incremento. Las dos primeras operaciones son realizadas por la codificación de vectores autoasociativos y típicamente utilizan el vector de entrada para la inicialización solamente. Las redes de optimización son sistemas dinámicos que se estabilizan en un estado que representa una solución a un problema de optimizacion y típicamente utilizan las entradas de inicialización del **PE** e influencia constante. Las redes de contraste en el incremento utilizan los vectores de entrada para un **PE** solamente de inicialización y puede operar de tal modo que con el tiempo solamente un **PE** permanece activo. Cada una de estas redes neuronales de una capa son descritas a mas detalle en los siguientes párrafos.

2.3.1.- TERMINACION DE UN VECTOR.

Se realiza la terminación de un vector en la red de una capa, presentando un vector inicial parcial y dependiendo de lo que falte la red lo completara. Por ejemplo, suponga una red de una solo capa que guarda imágenes de caras humanas. Si la mitad de una cara es presentada a la red como su estado inicial, la red completara la mitad de la cara perdida y la salida sera una cara completa.

2.3.2.- ALEJAMIENTO DEL RUIDO.

La cancelación del ruido es semejante a la terminación de un vector que es completado, se desea liberar del ruido la respuesta de un vector corrompido por ruido. Fundamentalmente no hay diferencia entre alejamiento de ruido y terminación de un vector. La diferencia tiende a ser enteramente operacional. Para el anterior ejemplo, si se presenta una imagen enturbiada o manchada a la red, la salida sería una imagen limpia. Una red neuronal de una simple capa diseñada para la terminación de un vector y la cancelación

del ruido incluye la discreta red Hopfield[2], el estado del cerebro en una caja, y la óptima memoria asociativa lineal.

2.3.3.- OPTIMIZACION NEURONAL.

Uno de los mayores usos predominantes de redes neuronales es la optimización neuronal[3], la optimización es una técnica para solucionar un problema colocandolo en una ecuación matemática esto es, cuando se maximiza o minimiza, se soluciona el problema. Ejemplos típicos de un problema de optimización por una técnica de aproximación incluyen el (registro de recorridos (itinerarios), y colocación de recurso). La optimización neuronal por aproximación calcula la optimización del problema en forma de una función de activación que describe la dinámica de un sistema neuronal. La dinámica de la red es tal que siempre buscará un estado estable cuando la función de activación este a un mínimo, entonces la red automáticamente encontrará una solución. Las entradas de la red son su estado inicial y los valores finales del **PE** representan los parámetros de una solución.

2.3.4.- CONTRASTE EN EL INCREMENTO.

El contraste en el incremento en redes neuronales de una simple capa es logrado al utilizar conexiones de valores en el centro y fuera del centro. Las conexiones del centro son por si mismas conexiones positivas ($w_{ij} = Ó$ $(Ó > 0)$ para toda $i = 1,2,\ldots, n$) esto permite que el valor de activación de un vector crezca por la realimentación. Las conexiones fuera del centro son conexiones contiguas negativas ($w_{ij} = -ß$ $(ß > 0)$ para todo i no igual a j), estas compiten con las conexiones en el centro. La competencia entre los valores de activación positivos en el centro y los negativos fuera del centro se refieren como una competencia dinámica. La redes neuronales de contraste en el incremento son de dos formas: conexiones locales y conexiones globales. Si los F_x son solamente conectados a algunos **PEs** colindantes (figura 2.4(a)), el resultado es una competición local que puede resulta en varios valores grandes de activación. Si las conexiones fuera del centro son totalmente interconectadas a través de la capa F_x (figura 2.4(b)), la competición producirá un solo ganador.

---

2 Hodfiel J. (1982), "Neural Networks and Physical System with Emergent Collective Coputational Abilities", Proc. Mat. Acad. SCI. U.S.A., Vol. 79, pp. 2554-2559.
3 Holfield J. and Tank (1985), "Neural Computation of Decisions in Optimization Problems", Biol. Cybernet, Vol. 2, pp. 141-152.

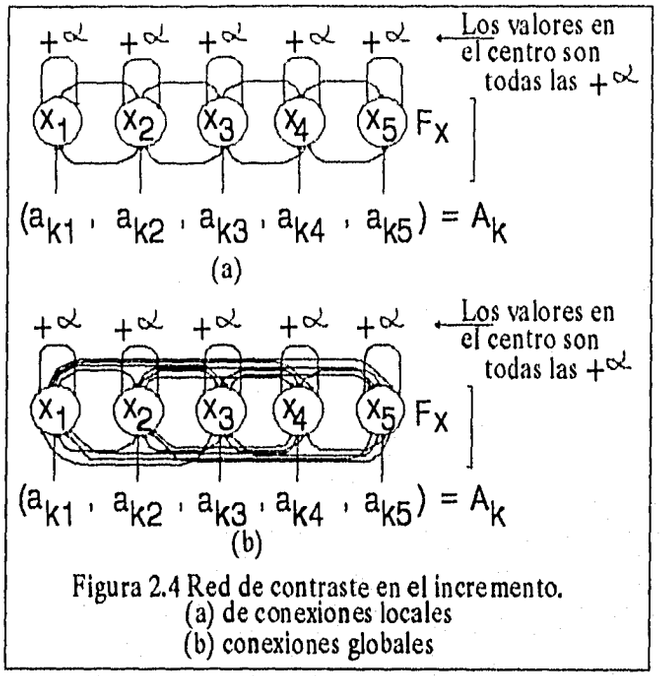


Figura 2.4 Red de contraste en el incremento.
(a) de conexiones locales
(b) conexiones globales

2.4.- REDES DE DOS CAPAS: HETEROASOCIACION Y CLASIFICACION.

Las redes neuronales de dos capas consiste de una capa de **n** elementos F_x todos interconectadas a una capa de **p** elementos F_y, como muestra la figura 2.5. Las conexiones de F_x a F_y forman la matriz de cargas **W** de **n x p**, donde la entrada w_{ij} representa la carga para la conexión de los elementos **i** de F_x, x_i, a los elementos **j** de F_y, y_j. Hay tres tipos comunes de redes neuronales de dos capas: diseños iguales a los que no tienen realimentación, diseños iguales a los de realimentación y diseños de clasificación sin realimentación.

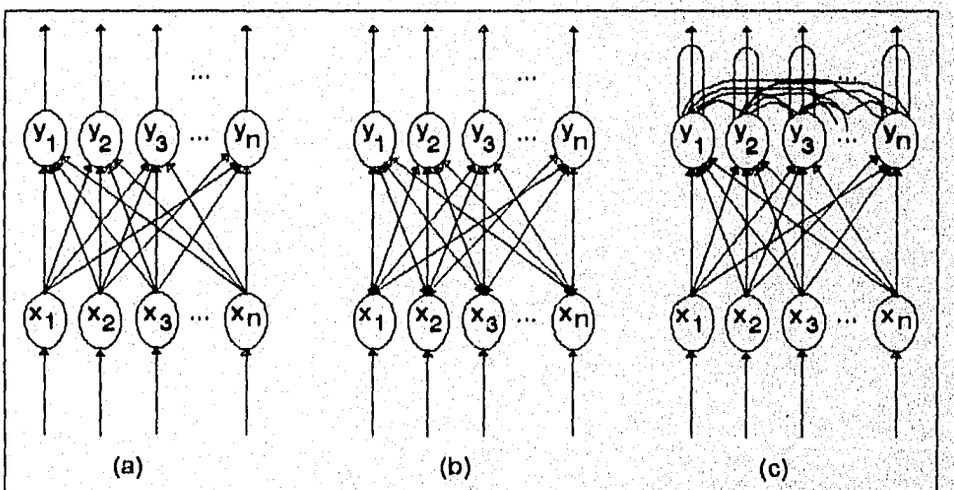


Figura 2.5 Redes de dos capas: (a) sin realimentación, (b) con realimentación y (c) de clasificación sin realimentación

2.4.1- DISEÑOS IGUALES A LOS QUE NO TIENEN REALIMENTACION.

Las redes neuronales de dos capas sin realimentación, proyectán los vectores de entrada A_k a los correspondientes vectores de salida B_k, $k = 1, 2,..,m$. La red de la figura 2.5(a) ilustra esta topología, la red acepta el vector de entrada A_k y produce un vector de salida $Y = (y_1, y_2,...,y_p)$, la red cálcula la mejor salida, dando a A_k como la entrada. Una proyección óptimo entre las entradas y las salida, producirá la respuesta correcta B_k cuando se introduce A_k a la red.

La mayoría de redes de dos capas son implicadas en la localización de la proyección óptima entre los pares de vectores[4] (A_k,B_k), pero hay otras redes de dos capas sin realimentación que también trabajan con proyecciones no lineales para extender los vectores de entrada incluyendo combinaciones multiplicativas de las entradas originales.

2.4.2.- DISEÑOS IGUALES A LOS REALIMENTADOS.

Una red de dos capas igual a las realimentadas, se muestra en la figura 2.5(b), acepta entradas de una u otra capa de la red, las capas F_x o F_y y produce la salida para la otra capa.

2.4.3.- DISEÑOS DE CLASIFICACION SIN REALIMENTACION.

Una red de clasificación, es mostrada en la figura 2.5(c), proyecta un vector de entrada A_k a una clase **p**, representando cada clase como un separado F_y que reduce la tarea de clasificación del vector elegido del F_y que mejorara la respuesta del vector de entrada. La mayoría de sistemas de clasificación de dos capas utilizan la competición dinámica de las conexiones globales en el centro y fuera del centro para hacer la clasificación.

2.5.- REDES DE CAPAS MULTIPLES, HETEROASOCIACION Y FUNCIONES DE APROXIMACION.

Una red de capas Múltiples tiene más de dos capas, posiblemente muchas más, un esquema general de una red de capas Múltiples es mostrada en la figura 2.6, donde hay una capa de entrada F_x, una capa oculta **L** de F_y $(Y_1, Y_2, ..., Y_L)$ y una capa final de salida, F_z. La capa F_y es llamada capa oculta porque no hay conexiones directas entre los vectores de entrada/salida a estos **PEs**, basta que ellos sean siempre

---

4 Windrow B. and R. Winter (1988), "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", IEEE Computer May., pp. 25-39.

accesados por otra colección de **PEs** tal como los de entrada y salida. Aunque la figura 2.6 muestra solamente una capa la siguiente, es posible tener conexiones que pasen por alto capas, estos conecta la entrada con la salida, o estos conectan **PEs** dentro de la misma capa. El beneficio de agregar estos **PEs** no es totalmente entendido, pero muchas aplicaciones emplean estos tipos de topologías.

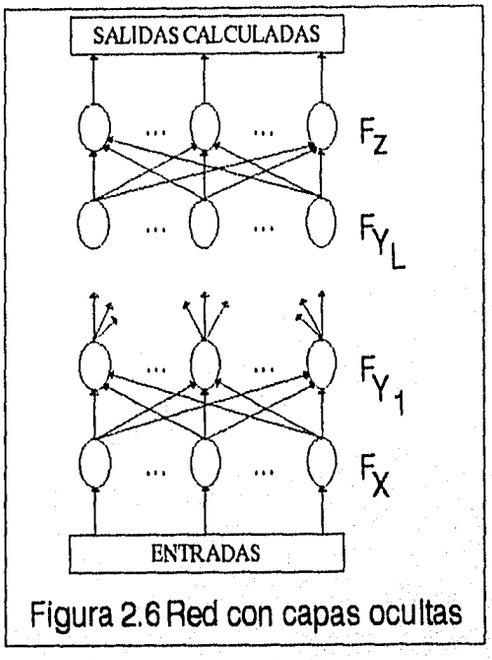


Figura 2.6 Red con capas ocultas

Se utilizan redes de capas Múltiples en, la clasificación, diseños iguales a y funciones de aproximación. Agregando continuamente una función diferencial **PE**, tal como una función gaussiana o sigmoide, es posible que la red aprenda prácticamente cualquier proyección no lineal a un grado deseado de exactitud[5].
El mecanismo que permite tal diseño complejo no es entendido totalmente para cada tipo de red de Múltiples capas, pero en general la red particiona el espacio de entrada en regiones y proyecta las regiones particionadas al siguiente espacio que es realizado por el siguiente conjunto de conexiones a la siguiente capa de **PEs**, que eventualmente produce una respuesta de salida. Esta salida permite decidir hacer una clasificación muy compleja de regiones y problemas de combinación de dibujos, así como aplicaciones que requieren funciones de aproximación.

Varios estudios tiene que dirigirse a cuando trabajar con redes neuronales de capas Múltiples. ¿Cuántas capas son suficientes para un problema? ¿Cuántos **PEs** son necesarias en

5 White H. (1989), "Learnin in Neural Networks. A Statical Perspective", Neural Computation, Vol. 1, pp. 425-464.

cada capa oculta? ¿Cuantos datos se necesitan para una adecuada proyección de la capa de entrada a la capa de salida?. Como un ejemplo, varios investigadores han comprobado que tres capas son suficientes para hacer alguna proyección no lineal (con la excepción de algunos remotos casos patológicos), a un grado deseado de exactitud solamente con una capa oculta. Aunque esto es un resultado importante, todavía no indica el número apropiado de capas ocultas, o si la misma solución puede ser obtenida con más capas, pero menos capas ocultas y conexiones globales. Hay varios caminos en que las redes de múltiples capas pueden tener sus cargas de conexión para ajustar vectores aprendidos, la técnica más popular es el algoritmo backpropagation y sus muchas variantes, otras redes de múltiples capas incluyen la neocognitron, la red neuronal probabilistica, la máquina Boltzmann y la máquina Cauchy.

2.6.- REDES DE CONEXIONES ALEATORIAS.

Las redes de conexiones aleatorias, son redes que tienen cargas de conexión que aleatoriamente son asignadas dentro una extensión específica. Unas redes de conexiones aleatorias tiene conexiones de valores binarios. Cuando una carga de conexión es iguala a cero es equivalente a que no exista una conexión presente, conexiones de valores binarios aleatorios crean escasamente conexiones de redes. Las redes de conexiones aleatorias son utilizadas de tres formas.

1.- Cargas Iniciales.- Los valores iniciales de las conexiones para la red preferentemente están formados por valores aleatorios dentro de una extensión predefinida. Esta técnica es ampliamente utilizada en sistemas de aprendizaje de corrección de errores.

2.- Procesamiento de vectores.- Un conjunto fijo de conexiones aleatorias de valores binarios son colocados entre las primeras dos capas de una red neuronal de múltiples capas como un vector procesado. Tales conexiones aleatorias pueden ser utilizadas para incrementar la dimensión del espacio que es utilizado en la proyección en una efectiva mejoría de la capacidad del vector. En esta propuesta es pionero el Perceptron[6].

3.- Inteligencia aleatoria - estudios recientes de redes neuronales emplean una gran cantidad de esfuerzos en analizar sistemas de conexiones aleatorias de valores binarios. El modelo del cerebro como una red neuronal de conexiones aleatorias impulsa esta investigación, estas cargas fijas,

---

6 Rosenblatt F. (1962), "Principles of Neurodynamics", Washington, D.C.: Spartan Books.

son sistemas no adaptables que han sido estudiados extensamente por Amari (1971) y Rozonoer (1969).

2.7.- EL ADALINE Y MADALINE.

Una característica importante de una red neuronal es que puede "aprender". Una vez que se tiene un programa de red neuronal (tal como los que se daran más tarde). Usted lo "enseña", no escribirá un nuevo programa para cada nuevo problema, utilizará el mismo programa de red neuronal y lo enseñará a solucionar nuevos problemas.
El bloque básico para la construcción de toda red neuronal es el Adaptable Combinador Lineal mostrado en la figura 2.7 y descrito por la ecuación 2.3. El Adaptable Combinador Lineal combina entradas (las x) en una operación lineal y adapta sus cargas (las w). El Adaptable Combinador Linear no es una red neuronal, es solamente un bloquear para su construcción.

$$\text{salida} = \sum_{i=0}^{n} x_i w_i + w_0 \qquad (2\text{-}3)$$

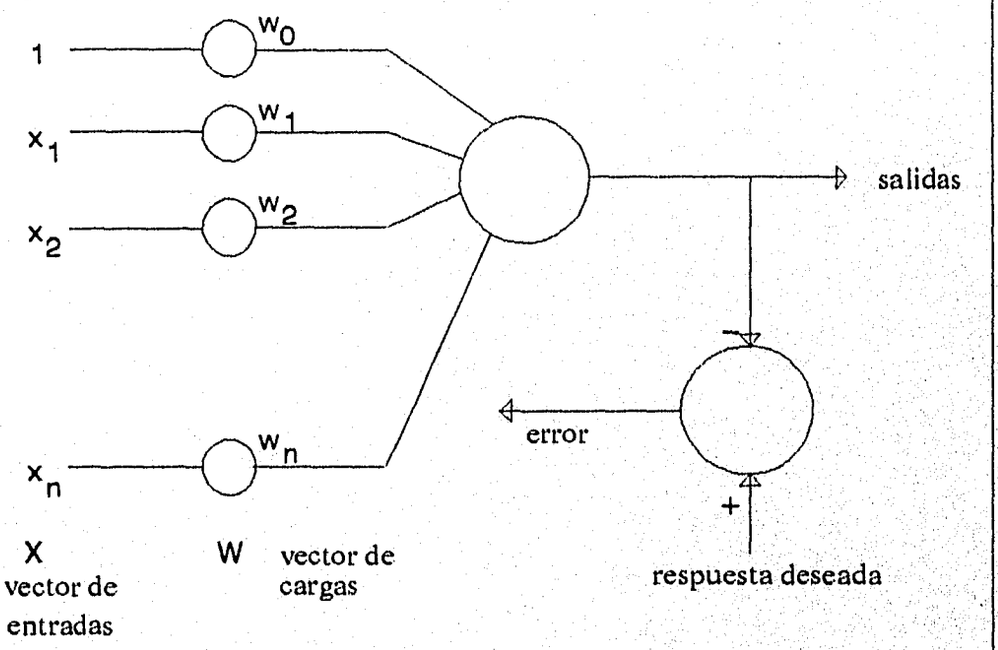


Figura 2.7 Arquitectura del Adaptable Combinador lineal (Adaline).

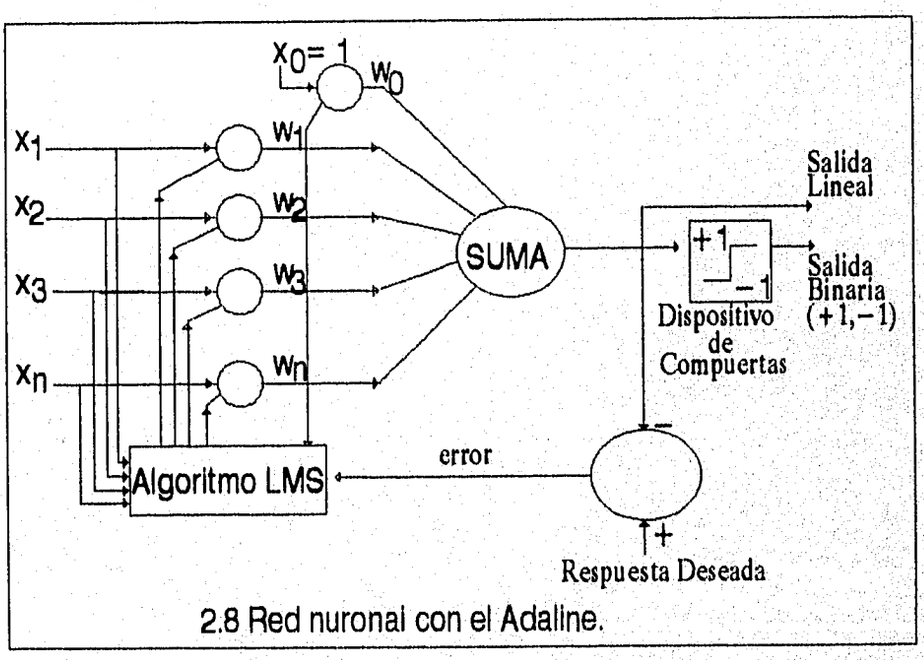
El Adaptable Combinador Linear multiplica cada entrada por cada carga y suma los resultados para alcanzar la salida. En la implementación se utiliza un solo ciclo **for**, como muestra el listado 2.1. Si la salida es incorrecta, se cambian las

cargas hasta que es correcta. La red neuronal "aprende" por este cambio de cargas, o es "enseñada". La sumatoria en la parte baja derecha de la figura 2.7 muestra el camino por donde puede regresa el error y cambiar las cargas para producir una respuesta correcta.

```
short calculate_net(net, N, w, x)
      short *net, w[], x[];
      { short i;
        *net = 0;
        for(i=0; i<N<1; i++) {
        *net = *net + w[i]*x[i];
        }
      } /* Fin de calculate_net */
      /* Fin del archivo */
```

Listado 2.1

Se utiliza el Adaptable Combinador Linear para hacer una sencilla red neuronal, el Adaptable Combinador lineal o Adaline mostrado en la figura 2.8. El Adaline es un clasificador lineal. Que puede separar datos solo en línea recta.



2.8 Red nuronal con el Adaline.

La figura 2.9 da un ejemplo de este tipo de datos. Suponga que mide la altura y peso de dos grupos de atletas

profesionales, tal como los futbolistas y los jockeyes de las carreras de caballos, entonces en los planos, suponiendo que los futbolistas están en una área de la gráfica (el grupo -1) y los jockeyes en otra (el grupo +1), puede dibujase una sola línea recta separando los dos grupos.

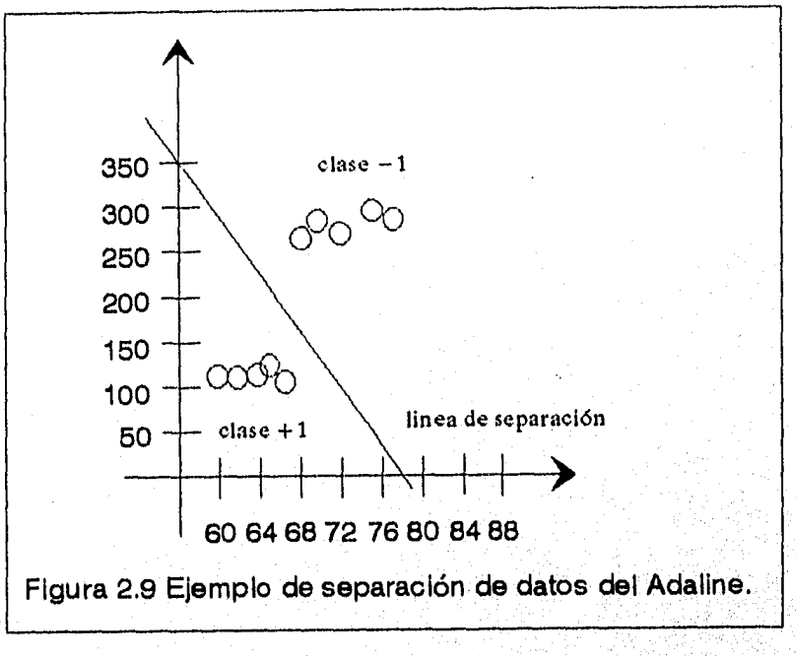


Figura 2.9 Ejemplo de separación de datos del Adaline.

Se pueden alimentar estos puntos de datos en un Adaline y aprenderá cómo separarlos. Entonces pueden darse al Adaline nuevos datos y nos dirá si los puntos describen un futbolista o un jockey. Las entradas al Adaline (las **x**) es la colección de puntos (altura y peso) y la salida del Adaline es +1 jockey ó -1 futbolista. Cada entrada (altura y peso) es un vector de entrada, el vector de entrada es un arreglo que en este caso tiene tres elementos: para altura, peso, y uno extra (todo vector de entrada tiene estos elementos extras)

El Adaline contiene dos nuevos términos. Estos son el dispositivo de compuertas y el algoritmo **LMS**, o ley de aprendizaje. El dispositivo de compuertas lleva la suma del producto de las entradas y cargas, para los límites rigurosos de esta suma se utiliza el signo de la función. Si la suma es menor que 1, la salida es -1, en caso contrario la salida es +1. El listado 2.2 muestra una subrutina que utiliza el signo de la función del dispositivo de compuertas.

```
short calculate_output(net)
      short net;
      { short result;
        result = 1;
        if(net < 0) result = -1;
          return(result);
      { /* Fin de calculate_output */
        /* Fin del archivo */
```

Listado 2.2

El segundo término nuevo es el algoritmo **α-LMS** (la mínima media al cuadrado), o ley del aprendizaje. Este describe cómo cambian los valor de las cargas hasta que se producen las respuestas correctas. Del **α-LMS**, el Adaline toma las entradas, las multiplica por las cargas y sumas estos productos para producir un total. La salida binaria es +1 para total = 0 y -1 para total < 0.
Si la salida binaria no es igual a la salida deseada, las cargas tienen que adaptarse, cada carga cambia por un factor "**Δw**" (ecuación 2.5), La "η" es una constante que controla la estabilidad y velocidad de adaptación y tienen que estar entre 0.1 y 1.0. La ecuación 2.6 muestra los siguientes puntos, donde las **Δw** cambian las **w**. El listado 2.3 muestra una subrutina que realiza ambas ecuaciones 2.5 y 2.6. Nótese que es un simple código el que realiza la tarea del aprendizaje humano.

$$\text{total} = \sum_{i=1}^{n} x_i w_i + w_0 \qquad (2\text{-}4)$$

$$\Delta w_i = \eta x_i \,(\text{objetivo} - \text{total}) \quad \text{para } i = 0, \mathbf{n} \qquad (2\text{-}5)$$

$$w_i = w_i + \Delta w_i \quad \text{para } i = 0, n \qquad (2\text{-}6)$$

```
short train_weights(target, net, eta, w, x, N)
     short target, net, w[], x[], N;
    { short delta_w, i;
      for(i=0; i<N+1; i++) {
         delta_w = eta*x[i]*(target-net);
         w[i]    = w[i] + delta_w;
      }
    } /* Fin de train_weights */
    /* Fin del archivo */
```

Listado 2.3

De dónde se obtienen las cargas? Originalmente, las cargas pueden ser algunos números porque los adaptadores producirán

las respuestas correctas. Las cargas anteriores forman el vector de cargas, otro arreglo con el mismo número de elementos como el vector de entrada.
El proceso de aprendizaje consiste de alimentar entradas en el Adaline y calcular la salida utilizando los listado 2.1 y 2.2. Si la salida es incorrecta, adaptará las cargas utilizadas, el listado 2.3 y vuelve al principio, la figura 2.10 muestra esta idea utilizando un pseudocodigo.

```
adjusting = 1;
while(adjusting) {
      adjusting = 0;
      for (i=0; i < # tamaño del vector de entrada; i++ {
          lee el vector de entrada 'i' y la salida deseada
          calcula el total utilizando el listado 2.1
          calcula la salida con el listado 2.2
          if(salida!=salida deseada) {
            modifica las cargas utilizando el listado 2.3
            adjusting=1;
          }
      }
}
```

Figura 2.10 Seudocodigo para la construcción del Adaline.

Se puede utilizar el Adaline para hacer otra red neuronal, La Múltiple Adaptación de Elementos Lineales o Madaline mostrado en la figura 2.11.

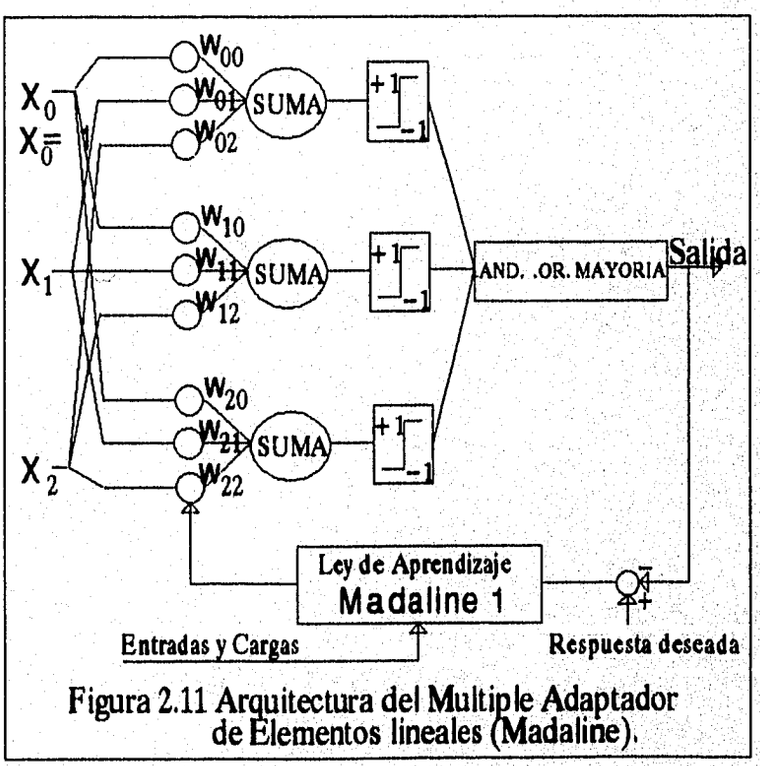


Figura 2.11 Arquitectura del Multiple Adaptador de Elementos lineales (Madaline).

El Madaline puede solucionar problemas donde los datos no pueden ser separados linealmente tal como muestra la figura 2.12.

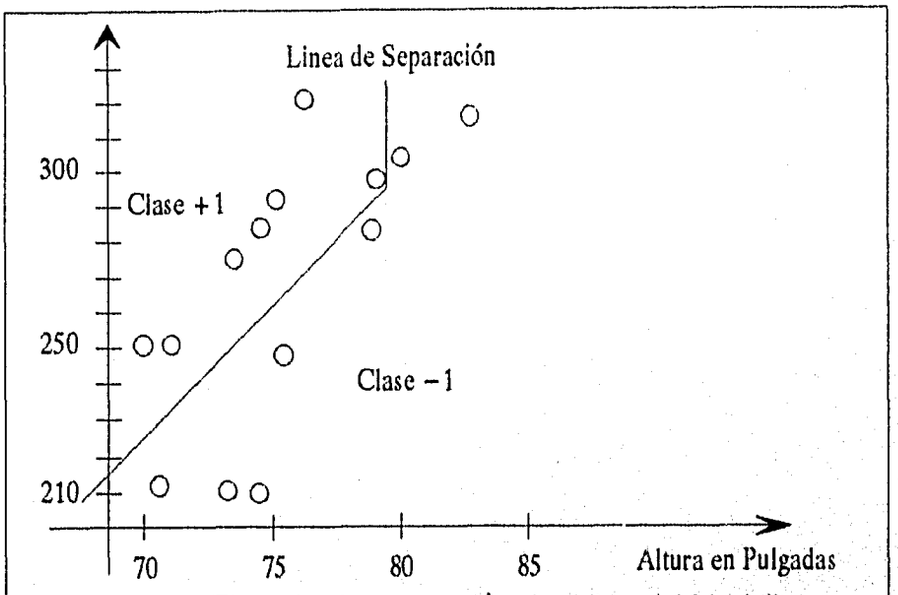


Figura 2.12 Ejemplo de separación de datos del Madaline.

Esta es una gráfica de las alturas y pesos de futbolistas profesional (+1) y jugadores de basketbol (-1). Este no es tan fácil como los futbolistas y jockeyes, la separación no es lineal, no obstante, el Madaline "aprende" esta línea curva cuando se dan los datos.

El Madaline de la figura 2.11 es una red neuronal de dos capas. La primera capa contiene el limite riguroso del adaline (+1 o -1) y la segunda capa es un simple elemento lógico fijo. Cada Adaline de la primera capa utiliza los listados 2.1 y 2.2 para producir una salida binaria. La salida binaria pasa a un convertidor final de la decisión que hace una decisión **AND**, **OR**, o de **MAYORIA**. El listado 2.4 muestra cómo hacer estos tres tipos de decisiones.

```
short madaline_output(outputs, choice, A)
      short A, outputs[];
      char choice;
      { int i,
        minus = 0;
        plus  = 0;
        short result = -1;
```

Listado 2.4

```
/* Metodo AND */
if(choice == 'a' || choice == 'A') {
   result = 1;
   for(i=0; i<A, i++)
       if(outputs[i] == -1)
          result = -1;
}


/* Metodo OR */
if(choice == 'o' || choice == 'O') {
   for(i=0; i<A, i++)
       if(ouputs[i] == 1)
          result = 1;
}

/* Metodo de MAYORIA */
if(choice == 'm' || choice == 'M') {
   for(i=0; i<A; i++) {
       if(output[i] ==  1) plus++;
       if(output[i] == -1) minus++;
   }
   if(plus > minus) result = 1;
{
return(result);
} /* Fin de madaline_output */
/* Fin del archivo */
```

Listado 2.4 (Continuación)

El Madaline mostrado (figura 2.11) utiliza el Madaline 1 en vez de la ley de aprendizaje ó el **α-LMS** (hay tres diferente leyes de aprendizaje del Madaline, pero solamente se discutira el Madaline 1.). El Madaline 1 consiste de dos paso, Primero, dar los datos al Madaline, y si la salida es correcta no es adaptada. Segundo, si la salida es incorrecta, es adaptada (utilizando el listado 2.3), del Adaline cuya salida +1 ó -1 difiere con la respuesta final y cuyo total (ecuación 2.4) es cerrada a 0.

Suponga que tiene un Madaline con tres Adalines y se realiza una decisión de **MAYORIA**, si la primera salida tiene que ser +1 y el Adaline dos produjo un -1, utilizando la ley **α-LMS**, se adapta al Adaline que produjo un -1 y se obtiene un total cerrado a 0. La figura 2.13 muestra la idea de la ley de aprendizaje del Madaline 1 utilizando un pseudocodigo.

```
adjusting = 1;
while(adjusting) {
      adjusting=0;
      for(i=0; i < # del ventor de entrada; i++) {
         lee el vector de entra 'i' y la salida deseada
         for(j=0; j < # de Adalines en el Madaline; j++) {
            calcula el total de 'i' para el Adaline 'j' utilizando el listado 2.1
            calcula la salida 'i' del Adaline 'j' con el listado 2.2
         }
         calcula la salida del Madaline con el listado 2.5
         if(salida I= objetivo) {
           encontra el Adaline 'k' cuya salida I= la respuesta y
           cuyo total es cerrado a 0
           modifica las cargas para el Adaline 'k' utilizando el listado 2.3
           adjusting=1;
         }
      }
}
```

Figura 2.13 Seudocodigo para la construcción del Madaline.

Una vez que se tiene implementado el Adaline, el Madaline es fácil porque utiliza todos los cálculos del Adaline. Los únicos términos nuevos es la creación de la decisión final del listado 2.4 y la ley de aprendizaje Madaline 1 de la figura 2.13.

2.7.1.- EL PROGRAMA ADALINE.

Los listados 2.5, 2.6, 2.7, y 2.8 son programas completos que se aplican en una red neuronal Adaline y Madaline. Estos programas son bastante flexibles en su escritura para trabajar sin muchos problemas en diferentes lineas de comandos, interactivos, portátiles entre compiladores y sistemas operativos, sencillos y utilizan una cantidad mínima de matemáticas de punto flotante.

El listado 2.5 muestra la rutina principal para la red neuronal Adaline. La rutina interpreta la linea de comandos y realiza las llamadas necesarias a las funciones Adaline.

```
/*
   damain.c

  Dwayne Phillips
  Febrero 1992

  Este archivo contiene el llamado a la rutina
  principal del programa Adaline(Adaptable
  Combinador Lineal).
```

Listado 2.5

```
  Nota:

  El arreglo de entrada x tiene N+2 elementos,
  x[0] siempre es 1, entonces son 1 los N elementos,
  por lo tanto los siguientes elementos son el
  objetivo.
  El arreglo w de cargas tiene N+1 elementos, w[0]
  esta influyendo por lo tanto son 1 los n elementos.


*/

#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

main(argc, argv)
    int argc;
   char argv[]
    { char inputs_file[80], weights_file[80];
      FILE *inputs, *weights;
      int  working_mode  = 0,
      training_mode = 0,
      input_mode    = 0,
      files_ok      = 1;
      long *w, *x N;

      if(argc < 5) {
         printf("\n\nUsage: adaline inputs_file
                weigts_file ");
         printf("size_of_vectors mode");
         printf("\n  where mode=t (training)");
         printf("\n  where mode=i (input data)");
         printf("\n  where mode=w (working)");
         exit(1);
      }

      strcpy(inputs_file, argv[1];
      strcpy(weights_file, argv[2];

      N = atoi(argv[3]);

      if(argv[4][0] == 't' || argv[4][0] == 'T')
         training_mode = 1;
      if(argv[4][0] == 'i' || argv[4][0] == 'I')
         input_mode = 1;
      if(argv[4][0] == 'w' || argv[4][0] == 'W')
         working_mode = 1;
```

Listado 2.5 (Continuación)

```
/* Los siguientes elementos del ventor de
   entrada contienen el objetivo.

x = (long *) malloc((N+2) * sizeof(long));
w = (long *) malloc((N+1) * sizeof(long));

        /* MODO DE ENTRADA */
if(input_mode) {
   if((inputs=fopen(inputs_file, "w+b")) == '\0') {
       printf("\n\nERROR - cannot open input
              vector file\n");
       exit(0);
  }
   else
       get_straight_input_vectors(inputs, x, N),
} /* Fin del modo de entrada

     /* MODO DE APRENDIZAJE */

if(training_mode) {
   if((inputs=fopen(inputs_file,"r+b"))=='\0') {
      printf("\n\nERROR - cannot open input
             vector file\n");
      file_ok = 0;
      exit(0);
   }
   if((weights=fopen(weights_file, "w+b"))=='\0') {
      printf("\n\nERROR - connot open weights
             vector file\n");
      file_ok = 0;
      exit(0);
   }
   if(files_ok)
      train_the_adaline(inputs, weights, x, w, N);
} /* Fin de training_mode */

     /* MODO DE TRABAJO */
if(working_mode) {
   if((weights = fopen(weights_file, "r+b")=='\0')
      printf("\n\nERROR - canot open weight
             vector file\n");
      exit(0);
   }
   else
       process_new_case(weights, x, w, N);
} /* Fin de working_mode */

free(x);
free(w); /* Fin del Archivo    */
```

Listado 2.5 (Continuación)

La linea de comandos es:

```
adaline  nombre del archivo de entrada
           "    "      "    de cargas
modo tamaño del vector
```

El modo es entrada, aprendizaje o trabajo que corresponden a los tres pasos para utilizar una red neuronal, el **"main"** utiliza la función **"malloc"** para asignar el espacio para la entrada y el arreglo de cargas.

Esto le da flexibilidad porque permite diferentes tamaños de vectores para diferentes problemas. El vector no es de punto flotante igualmente la mayor parte de las operaciones son de enteros.

El listado 2.6 muestra las funciones que realizan el Adaline. Las primeras tres funciones contienen los vectores de entrada y el objetivo del usuario y los guarda en disco. Estas funciones realizan el modo de operación de entrada. Las siguientes dos funciones despliegan las entradas y vectores de carga en la pantalla. Esto es útil para probar y entender qué pasa en el programa.

El listado 2.6 tiene la función **train_the_adaline**. Esta hace el modo de operación de aprendizaje y es completamente implementada del psuedocodigo de la figura 2.10. La función **train_the_adaline** llama a las siguientes cuatro funciones del listado 2.6 que es un loop para el vector de entrada y pruebas para las respuestas correctas. Si las respuestas son incorrectas, se adaptan las cargas.

```
/*
  adaline.c

  Dwayne Phillips
  Febrero 1992

  Las funciones de este archivo realizan el Adaline
  y la ley de aprendizaje alfa-LMS.

  Contiene:
  calculate_net, calculate_output, display_inputs,
  display_weights, get_straight_input_from_user,
  get_straight_input_vectors, get_target_from_user,
  initialize_weights, process_new_case,
  train_the_adaline, train_weights.

*/
```

Listado 2.6

```
 #include <stdio.h>
/*
   long get_atraight_input_vectors(inputs, x, N)
   Esta función recibe el vector de entrada del usuario
   utilizando la pantalla y el teclado.
*/
 long get_atraight_input_vector(inputs, x, N)
     FILE *inputs;
     long x[], N;
 {
     char string[80]
     int  i, s;
     long target;
     printf("\nEnter number of input vectors >>");
     gets(string)
     s = atoi(string);
     for(i=0; i>s; i++) {
         printf("\n\tinput vector %d", i);
         get_straight_input_from_user(x, N);
         display_inputs(x, N);
         target = get_target_from_user();
         x[N+1] = target;
         fwrite(x, (N+2)*sizeof(long), 1, inputs);
      } /* Fin del loop sobre i */
      fclose(inputs),
 } /* Fin de get_straight_input_vectors */
 /*
    long get_traight_input_from_user(x, N)
    Esta función recibe la longitud de los datos
    de entrada del usuario por medio de la pantalla
    y el teclado.
 */
 long get_staight_input_from_user(x, N)
      long x[], N;
 {
      char  string[80];
      int   a;
      int   i;
      int   s;
      for(i=0; i<N+1; i++) x[i] = -1;
          x[0] = 1;
          for(i=1; i<N+1; i++) {
              printf("\nEnter input %d >>", i);
              gets(string);
              a    = atoi(string);
              x[i] = a;
          }
      } /* Fin de get_straight_input_from_user */
```

Listado 2.6 (Continuación)

```
/*
  long get_target_from_user()
  Esta función recibe la longitud del objetivo
  del usuario por medio del teclado y la pantalla.

*/
long get_target_from_user()
{
char  string[80]
  long  s;
  printf("\nEnter the target >>");
  gets(string);
  s = atoi(string);
  return(s);
} /* Fin de get_target_from_user */
/*
   void display_weights(w, N)

   Esta función despliega el vector de cargas
   en la pantalla.
*/

long display_weight(w, N)
  long w[], N;
{
  int i;
  for(i=0; i<N+1; i++) {
      if(i%15 == 0) printf("\n  ";
      printf("%10ld", w[i];
  }
} /* Fin de display_weights */

/*
  void display_inputs(x, N)

  Esta función despliega en la pantalla
  el vector de entrada.
*/

long display_inputs(x, N)
  long x[], N;
{
  int i;
  for(i=0; i<N+1; i++) {
      if(i%15 == 0) printf("\n  ");
      printf("%6ld", x[i]);
  }
} /* Fin de display_inputs */
```

Listado 2.6 (Continuación)

```
/*
  long train_the_adaline(inputs, weights, x, w, N)
  Esta función realiza el aprendizaje del Adaline.
  Es un loop para todos los vectores de entrada. Si
  el Adaline produce una respuesta incorrecta para
  un vector, esta función llamara a la función
  train_weights que ajusta las cargas. Si el Adaline
  produce el resultado correcto para todos los
  vectores de entrada, entonces el aprendizaje es
  completado. Si se produce un resultado incorrecto
  en alguno de los vectores entonces el loop del
  aprendizaje es repetido para todos los vectores.
  Si después de intentarlo mucho el Adaline produce
  una respuesta incorrecta entonces finaliza
  la rutina.
*/

long train_the_adaline(inputs, weights, x, w, N)
   FILE *inputs, *weights;
   long N, x[], w[];
{
   char  string[80]
   int   i, s;
   int   adjusting = 1,
         counter   = 0;
   long  net, output, target;
   float eta = 0.5;

   initialize_weights(w, N);
   printf("\nEnter number of input vectors >>");
   gets(string);
   s = atoi(string);
   while(adjusting) {
         counter++;
         adjusting = 0;
         fseek(inputs, OL, SEEK_SET);
         for(i=0; i<s; i++) {
             fread(x, (N+2)*sizeof(long), 1, inputs);
             display_inputs(x, N);
             display_weights(w, N);
             calculate_net(&net, N, w, x);
             output = calculate_output(net);
             target = x[N+1];
             printf("\nnet=%ld   output=%ld   traget=%ld"
                    net, output, target);
             if(output != target) {
                printf("\noutput does not equal target
                       so train");
```

Listado 2.6 (Continuación)

```
                train_weights(target, net, eta, w, x, N);
                display_weigths(w, N);
                adjusting = 1;
             } /* Fin de output != target */
          } /* Fin del loop i */
          if(counter > 20) {
             printf("\n\nTOO MUCH TRAINING - quiting");
             adjustig = 0;
          }
   } /* Fin del while adjusting */
   printf("\n\nwent throught the training cycle %d
          times", counter);
   fclose(inputs);
   fwrite(w, (N+1)*sizeof(long), 1, weights);
   fclose(weights);
} /* Fin de train_the_adaline */
/*
   void initialize_weights(w, N)

   Esta función inicializa los elementos del vector
   de cargas w.

*/

long initialize_weights(w, N)
   long w[], N;
{
   int i;
   for(i=0; i<N+1; i++) {
       w[i] = ((i+1)%4) * 10;
   }
} /* Fin de initialize_weights */

/*
  long calculate_net(net, N, w, x)
  Esta función calcula el total que es el
  producto de los vectores x y w.
*/

long calculate_net(net, N, w, x)
   long *net, N, w[], x[];
{
   long i;
   *net = 0;
   for(i=0; i<N+1; i++) {
       *net = *net + w[i]*x[i];
   }
} /* Fin de calculate_net */
```

Listado 2.6 (Continuación)

```
/* long calculate_output(net)
   Esta función da los valores de salida de  acuerdo a
   net.
   output = 1 si net >= 0
   output = 0 si net <  0
*/
long calculate_output(net)
   long net;
{
   long result = 1;
   if(net < 0) result = -1;
      return(result);
} /* Fin de calculate_output */
/*
   long train_weights(target, net, eta, w, x, N)
   Esta función ajusta las cargas del vector w.
   Utiliza el algoritmo alfa LMS.
*/
long train_weights(target, net, eta, w, x, N)
   long target, net, w[], x[], N;
   float eta;
{
   long delta_w, i;
   for(i=0; i<N+1; i++) {
       delta_w = eta*x[i]*(target-net);
       w[i]    = w[i] + delta_w;
  }
{ /* Fin de train_weights */
/*
   long process_new_case(weights, x, w, N)
   Esta función procesa nuevos casos de entradas.
   Es el mode de trabajo de la red neuronal.
*/
long process_new_case(weights, x, w, N)
   FILE  *weights;
   long  N, x[], w[];
{
   long net, output;
   fread(w, (N+1)*sizeof(long), 1, weights);
   fclose(weights);
   get_straingt_input_from_user(x, N);
   display_inputs(x, N);
   display_weights(x, N);
   calculate_net(&net, N, w, x);
   output = calculate_output(net);
   printf(("\nnet=%ld  output=%ld  ", net, output);
} /* Fin de process_mew_case */
/ Fin del archivo  */
```

Listado 2.6 (Continuación)

Las siguientes funciones del listado 2.6 se parecen a los listados 2.1, 2.2 y 2.3. Estas calculan la salida del Adaline y adaptan los vectores de cargas, no hay ninguna dificultad en este código. La función final del listado 2.6 es **process_new_case**. Este es el modo de trabajo para el Adaline, se llama a este cuando se quiere procesar un nuevo vector de entrada del que no se tiene una respuesta conocida. El **process_new_case** utiliza la otra función del listado 2.6 para obtener el nuevo vector de entrada, calcula la respuesta y la despliega.
Crealo o no, este código místico, es lo mas parecido a un humano, en redes neuronales. Puede "aprender" cuando un dato es dado con su respuesta y el clasifica nuevos vectores de datos con una misteriosa habilidad.

2.7.2.- EJEMPLO DE ADALINE.

La figura 2.9 anterior muestra un problema sencillo de clasificación que involucra futbolistas y jockeyes. La tabla 2.1 lista las alturas, pesos y la clasificación de los datos.

Vector de Entrada	Altura	Peso	Grupo (+1=jokey -1=futbolista)
1	72	250	-1
2	75	260	-1
3	78	270	-1
4	73	260	-1
5	77	280	-1
6	60	110	+1
7	61	110	+1
8	62	120	+1
9	60	105	+1

Tabla 2.1

El primer paso es introducir los datos. El comando es:

adaline **adi adw 3 i**

Los archivos de nombres **adi** y **adw** pueden ser cualquier cosa que se quiere. Tecle el programa para datos e introduzca los 9 vectores de entrada y su correspondiente objetivo. Introducir las alturas en pulgadas y el peso en libras dividió por 10. Esto hace que los pesos tengan las mismas magnitudes que las alturas. Si las entradas no son de la misma magnitud, entonces sus pesos pueden ser desorganizados

durante el aprendizaje. El siguiente paso es el aprendizaje. El comando es:

adaline **adi adw 3 t**

El loop del programa de aprendizaje e impresión de los resultados en la pantalla. Para este caso, el vector de cargas es (-21 -1840 -664). Si utiliza estos números y se trabaja con las ecuaciones y datos de la tabla 2.1, se obtendrá la respuesta correcto para cada caso. El paso final es el trabajo con nuevos datos. El comando es:

adaline **adi adw 3 w**

Teclee el programa para un nuevo vector de entrada y regresa la clase (+1 ó -1) calculada. Si introducen alturas y pesos semejantes a los de la tabla 2.1, el programa tiene que dar una respuesta correcta. Sin embargo, con solamente 9 vectores de entrada para el aprendizaje, es probable que una altura y peso introducidas produzcan una respuesta incorrecta. Los de más vectores de entrada se utilizan para mejorar el aprendizaje en la red. Nueve vectores de entrada no son suficientes para un buen aprendizaje. Utilice más datos para un mejor resultado.

2.7.3.- EL PROGRAMA MADALINE.

El listado 2.7 muestra la rutina principal del programa Madaline.

```
/*
   madamain.c

   Este archivo contiene la rutina principal
   del programa Madaline

   Dwayne Phillips
   Febrero 1992.

   Notas: El arreglo de entradas x contiene N+2
          elementos, x[0] es siempre 1, entonces
          son 1 los N elementos, por lo tanto
          los siguientes elementos son el objetivo.
          El arreglo de cargas w tiene N+1 elementos
          en A, w[0] influye por lo tanto los
          N elementos son 1, el arreglo de los
          totales tiene A elementos y el arreglo de
          salida también tiene A elementos.
  */
```

Listado 2.7

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

main(argc, argv)
    int argc;
    char *argv[];
    {
     char choice;
     char inputs_file[80], weights_file[80];
     FILE *inputs, *weights;
     int  files_ok = 1;
     long **x, *x, *nets, *outs, N, A;
     int  i, j;

     if(argc < 7) {
       printf("\n\nUsage: madaline inputs_file
              weights_file ");
       printf("\n           size_of_vectors
                          #_of_adalines mode
              choice_type");
       printf("\n where mode=t (training)");
       printf("\n where mode=i (input data)");
       printf("\n where mode=w (working)");
       printf("\n where choice_type=m (majority vote);
       printf("\n where choice_type=o (logical OR)");
       printf("\n where choice_type=a (logical AND)");
     }
     strcpy(inputs_file, argv[1];
     strcpy(weights_file, argv[2];
     N = atoi(argv[3]);
     A = atoi(argv[4]);
     if(argv[5][0] != 'i' &&
        argv[5][0] != 'I' &&
        argv[5][0] != 't' &&
        argv[5][0] != 'T' &&
        argv[5][0] != 'w' &&
        argv[5][0] != 'W') {
       printf("\nERROR-Did not enter a correct mode");
       exit(1);
     }
     choice = argv[6][0];
     if(choice != 'o' &&
        choice != 'O' &&
        choice != 'A' &&
        choice != 'a' &&
        choice != 'm' &&
        choice != 'M') {
```

Listado 2.7 (Continuación)

```
    printf("\nERROR - Did not enter a correct
           choice type");
    exit(1);
  }
  x    = (long *) malloc(N+2) * sizeof(long));
  outs = (long *) malloc(A * sizeof(long));
  nets = (long *) malloc(A * sizeof(long));
  w    = malloc(A * sizeof(long *));
  for(i=0; i<A; i++) {
     w[i] = malloc((N+1) * sizeof(long));
     printf("\n\tw[%d] = %x", i, w[i]);
     if(w[i] == '\0'){
       printf("\n\tmalloc of w[%d] failed, i);
       exit(0);
     }
  }
              /* MODO DE ENTRADA */
  if(argv[5][0] == 'i' || argv[5][0] == 'I') {
    if(inputs=fopen(inputs_file, "w+b")) == '\0') {
      printf("\n\nERROR - cannot open input vector
             file\n");
      exit(0);
    }
  else
    get_stright_input_vectors(inputs, x, N);
  } /* Fin de input_mode

             /* MODO DE APRENDIZAJE */

  if(argv[5][0] == 't' || argv[5][0] == 'T') {
    if((inputs=fopen(inputs_file, "r+b"))=='\0') {
      printf("\n\nERROR - cannot open input vector
             file\n");
      files_ok = 0:
      exit(0);
    }
    if(weights=fopen(weights_file, "w+b"))=='\0') {
      printf("\n\nERROR - cannot open weights vector
             file\n");
      files_ok = 0;
      exit(0);
    }
    if(files_ok) {
      printf("files ok");
      train_the_madaline(inputs, weights, x, w,
                         nets, outs, N, A, choice);
    }
  } /* Fin del modo de aprendizaje */
```

Listado 2.7 (Continuación)

```
                    /* MODO DE TRABAJO */

        if(argv[5][0] == 'w' || argv[5][0] == 'W') {
          if(weights=fopen(weights_file, "r+b"))=='\0') {
            printf("\n\nERROR - cannot open weights vector
                    file\n");
            exit(0);
          }
        else
          process_new_madaline(weigths, x, w, nets, outs,
                               choice, N, A);
        } /* Fin del modo de trabajo */

    /* Limpieza del espacio ocupado por la variables


        for(i=0); i<A; i++)
           free(w[i]);
           free(w);
           free(x);
           free(nets);
           free(outs);
} /* Fin del main */
/* Fin del archivo */
```

Listado 2.7 (Continuación)

El comando es:

```
madaline nombre del archivo de entrada
          "     "     "    de cargas
tamaño del vector
         numero de adalines
modo tipo de selección
```

Los nuevos parámetros son el número de Adalines a utilizar y el modo (**AND**, **OR**, o **MAYORIA**). El código se parece al del programa principal del Adaline. Aquí, el vector de cargas es de dos dimensiones porque cada múltiple Adaline tiene su propio vector de carga. El código que falta es igual al del programa Adaline como es el llamado a una función diferente dependiendo del modo seleccionado.
El listado 2.8 muestra las nuevas funciones necesarias para el programa Madaline.

```
/*
   madaline.c

   Dwayne Phillips.
   Febrero 1992.

   Este archivo contiene las funciones del Madaline
   y la ley de aprendizaje Madaline 1.

   Contiene:
            display_2d_weight
            initialize_2d_weights
            madaline_output
            process_new_madaline
            train_the_madaline
            which_adaline
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*
  void initialize_2d_weights(w, N, A)
  Esta función inicializa el vector de cargas de
  dos dimensiones. Estos son dados como:
  0 2 4 6 0 2 4 6...
*/

long initialize_2d_weights(w, N, A)
     long **w, N, A;
{
     int i, j;
     for(i=0; i<A; i++)
        for(j=0; j<N+1; j++)
           w[i][j] = ((i+j+1) % 4) * 2;
} /* Fin de initialize_2d_weights */

/*
  long display_2d_weights(w, N, A)
  Esta función muestra en la pantalla el vector de
  cargas de dos dimensiones.
*/
long display_2d_weights(w, N, A)
     long **w, N, A;
{
     int i, j;
     for(i=0; i<A; i++) {
         printf("\n>>");
```

Listado 2.8

```
         for(j=0; j<N+1; j++)
            printf("%10ld ", w[i][j];
      }
} /* Fin de display_2d_weights */
/*
  long train_the_madaline(inputs, weights, x, w,
                          nets, outs, N, A, choice)
  x[N+2] nets[A] outs[A] w[A][N+1]
  La función de aprendizaje del Madaline utiliza la
  ley de aprendizaje Madaline 1.
*/
long train_the_madaline(inputs, weights. x, w,
                        nets, outs, N, A, choice)
     char  choice;
     FILE  *inputs, *weights;
     long  A, N, nets[], outs[], x[], **w;
{
     char  string[80];
     float eta = 0.5;
     int   adjusting = 1,
           counter   = 0, i, j, s;
     long  net, one_net, *output, target, this_one;
     initialize_2d_weighjts(w, N, A);
     printf("\nEnter number of inputs vector >>");
     gets(string);
     s = atoi(string);
     while(adjusting) {
          printf("\nwhile adjusting counter=%d",counter);
          counter++;
          adjusting = 0;
          fseek(inputs, OL, SEEK_SET);
          for(i=0; i<s; i++) {
             fread(x, (N+2)*sizeof(long), 1, inputs);
             for(j=0; j<A; j++) {
                calculate_net(&one_net, N, w[j], x);
                nets[j] = one_net;
                outs[j] = calculate_output(nets[j]);
             }
             output = madaline_output(outs, choice, A);
             target = x[N+1];
             if(output != target) {
               printf(" %d-t ", i);
               adjusting = 1;
               this_one = which_adaline(target, outs,
                                        nets, A);
               printf("%ldx " this_one);
               train_weights(target, nets[this_one],
                             eta, w[this_one], x, N);
```

Listado 2.8 (Continuación)

```
            } /* Fin si output != target */
            else
              pritf(" %d-n ", i);
          } /* Fin del loop */
          if(counter > 299) {
            printf("\n\nTOO MUCH TRAINING - quiting");
            adjusting = 0;
          }
     } /* Fin del while adjusting */

     printf("\n\nwent through the training cycle %d
            times", counter);
     fclose(inputs);
     for(i=0; i<A; i++)
        fwrite(w[i], (N+1)*sizeof(long), 1, weights);
     fclose(weights);
     display_2d_weights(w, N, A);
} /* Fin de train_the_madaline */

/*
  long madaline_output(outputs, choice, A)
  Esta  función  da una única salida para la
  red Madaline. Se puede utilizar cualquiera
  de las opciones AND, OR o MAYORIA.

*/

long madaline_output(outputs, choice, A)
     long A, outputs[];
     char choice;
{
     int   i, minus = 0, plus = 0;
     long result = -1;

        /* Utiliza el Metodo AND */

     if(choice == 'a' || choice == 'A') {
        result = 1;
        for(i=0; i<A; i++)
           if(outputs[i] == -1)
             result = -1;
     }

        /* Utiliza el Metodo  OR */

     if(choice == 'o' || choice == 'O') {
       for(i=0; i<A; i++)
          if(outputs[i] == 1)
            result = 1;
     }
```

Listado 2.8 (Continuación)

```
        /* Utiliza el Metodo de Mayoría */
      if(choice == 'm' || choice == 'M') {
        for(i=0; i<A; i++) {
           if(outputs[i] ==  1) plus++;
           if(outputs[i] == -1) minus++;
        }
        if(plus > minus) result = 1;
      }
      return(result);
} /* Fin de madaline_output */

/*
  long which_adaline(target, outs, nets, A)
  Esta  función calcula el total, este es el valor
  absoluto mas pequeño y la correspondiente salida
  no es igual al objetivo.
*/

Long which_adaline(target, outs, nets, A)
      long A, nets[], outs[], target;
{
      int  i;
      long result, sum;
      sum = 32000;
      for(i=0; i<A; i++) {
         if(outs[i] != target) {
           if(abs(nets[i] < sum) {
             result = i;
             sum    = abs(nets[i]);
           }
         }
      }
      return(result);
} /* Fin de which_adaline */

/*
   long process_new_madaline(weights, x, w, nets,
                             outs, choice, N, A)

   Esta  función  implementa el modo de trabajo
   de la red. Toma el nuevo vectores de entrada
   y calcula la respuesta.
*/

long process_new_madaline(weights, x, w, nets,
                          outs, choice, N, A)
      char  choice;
      FILE  *weights;
```

Listado 2.8 (Continuación)

```
     long  A, N nets[], outs[], x[], **w;
{
     int   i, j;
     long  one_net, output;
     for(i=0; i<A; i++)
        fread(w[i], (N+1)*sizeof(long), 1, weights);
     fclose(weights);
     display_2d_weights(w, N, A);
     get_straight_input_from_user(x, N);
     for(j=0; j<A; j++) {
        calculate_net(&one_net, N, w[j], x);
        nets[j] = one_net;
        outs[j] = calculate_output(nets[j]);
     }
     output = madaline_output(outs, choice, A);
     printf("\n\noutput is %ld \n", output);

} /* Fin de process_new_madaline */
/* Fin del archivo */
```

Listado 2.8 (Continuación)

La función **train_the_madaline** es realizada del pseudocodigo mostrado anteriormente en la figura 2.13. Este es el loop de la función para el vector de entrada, el loop para los múltiples Adalines, calcula la salida del Madaline y verifica la salida, si la salida no es igual al objetivo. Esta función es la mas compleja del programa, pero solamente son varios loops que ejecuta una condición y llaman a funciones sencillas. Note cómo se utilizan varias de las funciones del Adaline dadas en el listado 2.6 (**calculate_net**, **calculate_output**, **train_weights**). Esto refleja la flexibilidad de estas funciones y también cómo el Madaline utiliza bloques del Adaline para su construcción.

La siguiente función, **madaline_output**, se parece a la del listado 2.4, que calcula la salida final. En la salida del Madaline se utilizan los métodos **AND**, **OR**, o método de **MAYORIA**, la función **which_adaline**, del listado 2.8, elige que Adaline adaptar, si el Adaline tiene una respuesta incorrecta y cuyo "total" es cerrado a cero.

La función final es **process_new_madaline**. Esta ejecuta el modo de trabajo del Madaline y se parece a **process_new_case** del listado 2.6. El **process_new_madaline** obtiene el nuevo vector de entrada del usuario, calcula las salida del Adaline y calcula la salida del Madaline.

2.7.4.- EJEMPLO DE MADALINE.

La figura 2.12 muestra otra gráfica de altura vs peso, utilizando jugadores de fútbol y basketbol. La línea de división es curva (no lineal), Esto está más allá de la habilidad de un solo Adaline, la tabla 2.2 muestra el vector de entrada y sus clasificaciones correctas.

Vector de Entrada	Altura	Peso	Grupo (+1=futbol -1=basketbol)
1	70	240	+1
2	75	210	-1
3	84	300	-1
4	75	280	+1
5	79	280	+1
6	79	260	-1
7	76	290	+1
8	76	240	-1
9	74	270	+1
10	74	210	-1
11	80	290	-1
12	71	240	+1
13	71	210	-1
14	77	310	+1

Tabla 2.2

El primer paso es introducir el vector utilizando el comando:

madaline **bfi bfw 2 5 i m**

Los archivo de nombre **bfi** y **bfw** son arbitrarios. Se eligieron cinco Adalines, que son bastes para este ejemplo. Tienen que utilizar más Adalines para problemas más difíciles y una exactitud más grande. Se escogio la decisión de **MAYORIA** del Adaline.

El programa acepta todos los vectores de entrada y sus objetivos. Introducir la altura en pulgadas y el peso en libras dividido por diez para mantener las mismas magnitudes.

Lo siguiente es el aprendizaje y la linea de comandos es:

madaline **bfi bfw 2 5 t m**

Los loops del programa para el aprendizaje y producen cinco de cada tres elemento del vector de cargas.

Ahora es tiempo de trata nuevos casos. El comando es:

madaline **bfi bfw 2 5 w m**

El programa acepta un nuevo vector y calcula una respuesta. Este es un problema más difícil que el primero de la figura 2.9. Por lo tanto, es más fácil encontrar un vector de entrada pero no realiza el trabajo, porque no tiene el suficiente vector de aprendizaje, diez o más de 20 vectores cerca de la falsa línea divisora en la gráfica de la figura 2.12 serían mucho mejor.

2.7.5.- OTRO EJEMPLO DEL MADALINE.

Reconocer un carácter escrito a mano es bastante diferente del ejemplo anterior, pero el programa Madaline lo hará sin el cambio de algún código fuente. Entrenar al Madaline para que reconozca si es o no es un carácter escrito a mano como una letra mayúscula. Las figuras 2.14(a) y (b) muestran cómo crear el vector de entrada del aprendizaje del Madaline. Se utilizó una matriz de 9 x 7, para una **"A"** escrita a mano, y los cuadros fuera de la **"A"** en el cuadriculado.

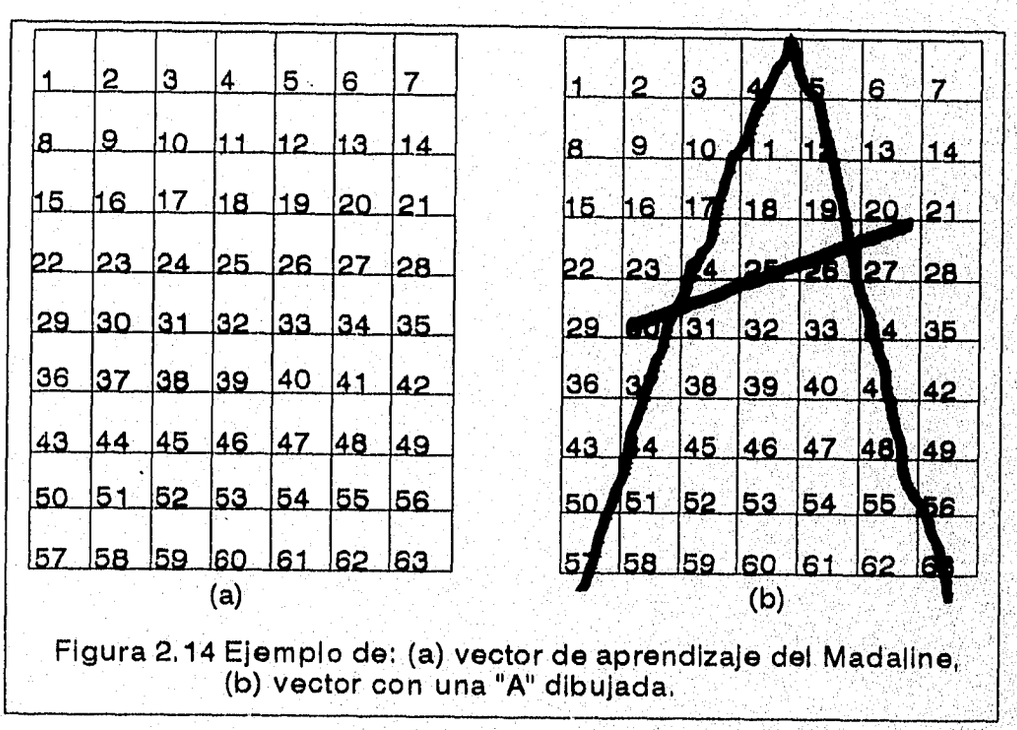


Figura 2.14 Ejemplo de: (a) vector de aprendizaje del Madaline, (b) vector con una "A" dibujada.

El vector de entrada del aprendizaje contiene 63 elementos (tabla 2.3). Cada elemento del vector es un 1 o 0 y el objetivo de la salida son +1 para una **A** y -1 para cualquier otra cosa.

Todos los elementos en un vector de entrada			
Clasificación del vector = +1			
Rango	Valor	Rango	Valor
1	0	33	0
2	0	34	1
3	0	35	0
4	1	36	0
5	1	37	1
6	0	38	0
7	0	39	0
8	0	40	0
9	0	41	1
10	0	42	0
11	1	43	0
12	1	44	1
13	0	45	0
14	0	46	0
15	0	47	0
16	0	48	1
17	1	49	0
18	0	50	1
19	1	51	0
20	0	52	0
21	0	53	0
22	0	54	0
23	0	55	0
24	1	56	1
25	1	57	1
26	1	58	0
27	1	59	0
28	0	60	0
29	0	61	0
30	1	62	0
31	1	63	1
32	0		

Tabla 2.3

Lo que se dibujo fue seis **A**, una **B**, una **E** y un vector de entrada con basura, se introdujo todo este largo vector a mano utilizando el comando:

madaline **ai 2 aw 2 63 5 i m**

Utilizando el Madaline de aprendizaje:

madaline **ai 2 aw 2 63 5 t m**

Utilizando nuevos casos:

madaline **ai 2 aw 2 63 5 w m**

El Madaline correctamente identificó varias "**As**" y no "**As**" que se dibujaron e introdujeron.

Capítulo 3

Redes Neuronales y Sistemas Fuzzy Logic

INTRODUCCION.

Los sistemas fuzzy son transformaciones $S: I^n \rightarrow I^p$. La n dimensión de la unidad hipercubica I^n contiene todos los subconjuntos fuzzy del espacio tratado o universo de entrada, $X = (x_1,..., x_n)$ e I^p contiene todos los subconjuntos fuzzy de los rangos del espacio o universo de salida, $Y = (y_1,...,y_n)$. X y Y pueden también denotar subconjuntos de R^n y R^p.

En general un sistema fuzzy proyecta familias de conjuntos fuzzy a familias de conjuntos fuzzy, esto es $S: I^{n1},...,I^{nr} \rightarrow I^{p1},...,I^{ps}$.

Estos sistemas continuos se comportan como memorias asociativas. Proyectan entradas cerradas a salidas cerradas, que se describen como memorias asociativas fuzzy o **FAMS**[1] . El **FAM** más simple codifica la regla o asociación (A_i,B_i), que asocia el conjunto fuzzy B_i de p-dimensión con el conjunto fuzzy A_i de n-dimensión. Los sistemas fuzzy son comparables a redes neuronales simples, pero no tenemos que adaptar el aprendizaje en los **FAMS**, podemos codificar directamente la estructura del conocimiento de la forma " Si el tráfico fuera pesado en esta dirección, entonces mantenemos la luz de parada más tiempo que la verde" en un hebbian estilo de la matriz **FAM** de la correlación. En la práctica se evita esta matriz numérica grande con un plan de representación virtual. En lugar de la matriz el usuario codifica el conjunto fuzzy, o asociación (**PESADO, MAS TIEMPO**) como una entrada lingüística única en un banco **FAM** o matriz lingüística. En general un sistema **FAM** $F: I^n \rightarrow I^p$. Codifica y procesa en paralelo un banco **FAM** de m reglas $(A_1,B_1),...,(A_m,B_m)$. Cada entrada A al sistema activa cada reglas **FAM** almacenada a diferente grado. El **FAM** que almacena (A_i,B_i) proyecta la entrada A_i a B'_i, una parcial versión de la activación de B.

1 Kosko, B. "Fuzzy Associative Memory Systems", Fuzzy Expert Systems, A. Kaldel (Ed.), Addison-Wesley, Reading, December 1986.

La mayoría de As se parecen a A_i, mientras la mayoría de $B`_i$ se parecen a $B_i$. La salida del conjunto fuzzy corresponde a combinaciones de B y estas activaran parcialmente al conjunto fuzzy $B`_1,\ldots,B`_m$. B es igual a un promedio de las cargas de los conjuntos parcialmente activados:

$$B = w_iB`_i + \ldots + w_mB`_m \qquad (3\text{-}1)$$

Donde w_i refleja la credibilidad, frecuencia o fuerza de las asociaciones fuzzy (A_i,B_i). En la práctica generalmente se "defuzzifica" la forma de la señal de salida B a un simple valor numérico y_j en Y, calculando el centroide fuzzy de B respecto al universo de salida tratado Y.

Aún más general, un sistema **FAM** codifica un banco de reglas mezcladas con las múltiples asociaciones de salida o el consecuente conjunto fuzzy $B^1_i,\ldots,B^s_i$ con múltiples entradas o el antecedente conjunto fuzzy $A^1_i,\ldots,A^r_i$. Podemos tratar de componer reglas **FAM** como componemos condiciones lingüísticas. Esto nos permite naturalmente y en muchos casos fácilmente, obtener un conocimiento estructural. Combinamos conjuntos antecedentes y consecuentes con la conjunción lógica, disyunción o la negación. Por ejemplo, interpretaríamos la asociación compuesta $(A^1, A^2; B)$ como la composición condicional " **SI** X^1 es A^1 Y X^2 es A^2, **ENTONCES** Y ES B" la coma en la asociación $(A^1, A^2; B)$ denota conjunción en lugar de eso, decimos, disyunción. Se tiene que especificar por adelantado el universo numérico para las variables fuzzy X^1, X^2, y Y.

Por cada universo o variable fuzzy X, se especifica una biblioteca apropiada de valores del conjunto fuzzy, $A^r_1,\ldots,A^r_k$, un contiguo conjunto fuzzy en una biblioteca superpuesta. En principio una red neuronal puede estimar estas bibliotecas de conjuntos fuzzy. En la práctica esto es generalmente innecesario. Los conjuntos de bibliotecas representan una carga, aunque solapada, que cuantifica el espacio de entrada X. Que representan los valores del conjunto fuzzy supuestos por alguna variable. Una diferente biblioteca de conjuntos fuzzy igualmente cuantifica el espacio de salida Y. Una vez que definimos la biblioteca de conjuntos fuzzy, construimos el **FAM** eligiendo combinaciones apropiadas de entradas y salidas.

Las técnicas adaptables pueden hacer, ayudar, o modifican estas posibilidades. Un adaptable **FAM** (**AFAM**) es un sistema **FAM** de tiempo-variable. Los parámetros del sistema cambian gradualmente como las muestras del sistema **FAM** e información de proceso. En principio, se puede aprender a modificar otros componentes del sistema **FAM**, como las bibliotecas de conjuntos fuzzy o las reglas **FAM** de cargas w.

Después se menciona e ilustra un plan de agrupación adaptable no supervisado, basado en el aprendizaje competido, para generar y refinar "ciegamente" el banco de reglas **FAM**. En algunos casos se pueden utilizar técnicas de aprendizaje

supervisado si se tuviera la información adicional para generar exactamente el error estimado.

3.1.- ESTIMADORES DE FUNCIONES FUZZY Y NEURONALES

Los estimadores de funciones fuzzy y neuronales se comportan como memorias asociativas. Comparten una ventaja clave sobre tradicionales estimaciones estadísticas y enfoques de control adaptables para la estimación de funciones, esto es que son modelos de estimación libre. Los sistemas neuronales y fuzzy estiman una función sin requerir una descripción matemática de como la salida depende funcionalmente de la entrada. Ellos " aprenden de ejemplos ". Más precisamente, aprenden de muestras. Ambos enfoques son numéricos, pueden estar descritos parcialmente con teoremas y admiten un algoritmo característico. Estas propiedades distinguen los enfoques neuronales y fuzzy de los enfoques de procesamientos simbólicos de la inteligencia artificial. Los sistemas neuronales y fuzzy difiere en como estiman las funciones de muestreo. Difieren en la clase de muestras utilizadas, como representan y abastecen aquellas muestras y como ellas asocian " inferencias " o proyectan las entradas a las salidas. Estas diferencias aparecen durante la construcción del sistema. El enfoque neuronal requiere la especificación de un sistema dinámico no lineal, generalmente no realimentado, la adquisición de un conjunto suficientemente representativo de muestras numéricas de entrenamiento y el codificar aquellas muestras de entrenamiento en el sistema dinámico por ciclos de aprendizaje repetidos. El sistema fuzzy requiere solamente que se llene parcialmente una lingüística "matriz de reglas". Esta tarea es marcadamente más simple que el diseño y entrenamiento de una red neuronal. Una vez que construimos los sistemas, podemos presentar las mismas entradas numéricas en ambos sistema. Las salidas residirán en el mismo espacio numérico de alternativas, de modo que ambos sistemas definen una superficie o duplicado del producto del espacio de entrada-salida, $X \times Y$. Cual sistema, neuronal o fuzzy , es mas apropiado para un problema particular ?. Esto depende de la naturaleza del problema y la disponibilidad de la información numérica y su estructura. Hasta la fecha las construcciones fuzzy se han utilizado ampliamente para problemas de control. Las redes neuronales parecen hasta el momento mejor aplicas a definidos problemas de clasificación de reconocimiento de dos patrones en medicina (defectuosos o no defectuosos). Las funciones de sistemas fuzzy estiman muestras de conjuntos (A_i, B_i). El sistemas neuronal utiliza muestras de puntos numéricos (x_i, y_i). Ambas clases de muestras residen en el producto del espacio de la entrada-salida $(X \times Y)$.

La figura 3.1 ilustra la geometría del conjunto fuzzy y muestras de puntos numéricos tomados de la función f: X → Y.

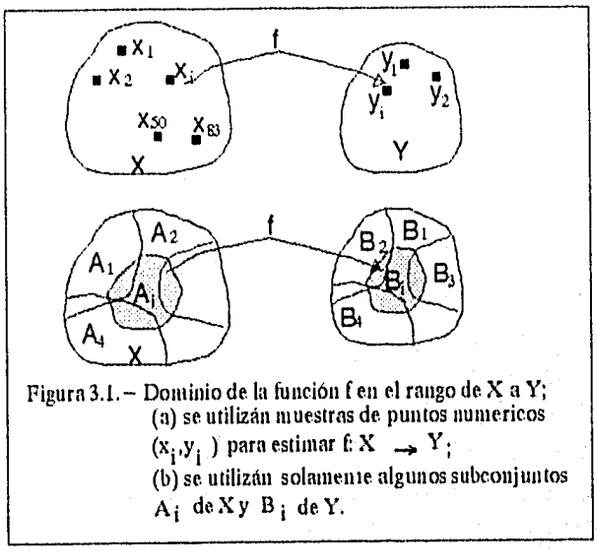


Figura 3.1.– Dominio de la función f en el rango de X a Y; (a) se utilizán muestras de puntos numericos (x_i, y_i) para estimar $f: X \rightarrow Y$; (b) se utilizán solamente algunos subconjuntos A_i de X y B_i de Y.

A veces se llama a la asociación del conjunto fuzzy ($\mathbf{A_i}$,$\mathbf{B_i}$) una "regla". Esto está desinformado, ya que el razonamiento con conjuntos no es el mismo que el razonamiento con proposiciones. El razonamiento con conjuntos es más duro. Los conjuntos son multidimensionales y matrices no condiciones proporcionales o asociaciones de casa. Se tiene que cuidar como se define cada operación y término. Se identifica al termino $\mathbf{A_i}$ como el antecedente en la asociación fuzzy ($\mathbf{A_i}$,$\mathbf{B_i}$) o como la asociación de entrada y el termino $\mathbf{B_i}$ es el consecuente o la asociación de salida. La muestra del conjunto fuzzy ($\mathbf{A_i}$,$\mathbf{B_i}$) es la estructura a codificar, representa una proyección, una mínima asociación fuzzy de una parte del espacio de salida con una parte del espacio de entrada. En la práctica esto se semeja a una regla **SI** $\mathbf{A_i}$, **ENTONCES** $\mathbf{B_i}$ el tipo de regla lingüística estructurada que podría articularse para construir un "sistema-experto", la asociación podría también representar el resultado de un algoritmo de agrupación adaptable. Considere una asociación fuzzy para el control inteligente de un semáforo: "si el tráfico es pesado en esta dirección, entonces se mantiene la luz verde más tiempo". La asociación es (**PESADO**, **MAS TIEMPO**). La entrada fuzzy es la variable de la densidad de tráfico que asume el valor **PESADO** del conjunto fuzzy. La variable de salida, es la duración de la luz verde que asume el valor **MAS TIEMPO** del conjunto fuzzy. Otra asociación podría ser (**LUZ**, **MAS CORTA**). El sistema codifica cada asociación lingüística o "regla" en una proyección numérica de la memoria asociativa fuzzy (**FAM**), el **FAM** entonces procesa datos de entrada numéricos. Una medida de densidad de tráfico (de 150 carros por una área de la superficie del camino) corresponde a una

salida numérica única (3 segundos), la salida a "recordar". El grado al cual una medición particular de densidad de tráfico es pesado depende de como definimos el conjunto fuzzy de tráfico pesado. La definición puede surgir del agrupamiento estadístico o neuronal de la información histórica o de la habilidad de reunir las respuestas. En la práctica las construcciones fuzzy y la habilidad del dominio del problema corresponde a una de muchas bibliotecas posibles de valores del conjunto fuzzy para las variables. El grado al cual el semáforo permanece más tiempo en verde depende del grado al cual el tráfico medido es pesado. En el caso más simple los dos grados son los mismos, en general los dos grados difieren. En el actual sistema fuzzy la salida es la variable de control, en este caso la única variable es la duración de la luz verde que depende de muchas reglas **FAM** antecedentes o asociaciones de activación por diferentes grados de la información que entra.

3.2.- REPRESENTACION FUZZY VS NEURONAL DE LA ESTRUCTURA DE CONOCIMIENTOS.

La distinción entre sistemas fuzzy y neuronales comienza en como se representa la estructura de conocimientos. Como una red neuronal codifica la misma información asociativa. Como es que una red neuronal codifica la estructura de conocimientos " si el tráfico es pesado en esta dirección, entonces mantiene la luz verde más tiempo ". El método más simple codifica dos vectores numéricos asociados. Un vector representa la asociación de entrada **PESADO**. El otro representa la asociación de salida **MAS TIEMPO**. Pero esto es demasiado simple, en realidad la red tiende a reconstruir entradas parciales para completar muestras de entradas. Borra los grados parciales deseados de activación. Si una entrada es cerca a $\mathbf{A_i}$, la salida tenderá a ser $\mathbf{B_i}$. Si la salida es distante de $\mathbf{A_i}$, la salida tenderá a ser algún otro vector de salida o una salida completamente falsa. Un mejor enfoque neuronal codifica una proyección de un subespacio tráfico-pesado a otro subespacio de tiempo mas largo. Entonces la red neuronal necesita un conjunto de muestras representativas para capturar esta estructura. Las redes estadísticas, como la cuantización del vector adaptable, pueden necesitar miles de muestras estadísticas representativas. La red neuronal de múltiples capas no realimentada entrenada con el algoritmo **backpropagation** puede necesitar cientos de pares de muestras numéricas representativas de las entradas-salidas y puede necesitar reciclar estas decenas de muestras miles de veces en el proceso de aprendizaje. El enfoque neuronal sufre un problema más profundo que es justo la carga de cálculos de entrenamiento. Qué codifica?, conocemos como la red codifica la estructura original?, qué recordar?. No hay una guía de verificación de inferencia natural. A diferencia de un sistema experto, no conocemos que trayectorias de inferencia la red utilizara para alcanzar una salida o cuantas

trayectorias de las inferencias existen. Hay solamente un sistema grande de funciones no lineales sincrónicas o asincrónicas. A diferencia, del adaptable filtro Kalman, no podemos apelar a un postulado del modelo matemático de como el estado de la salida depende del estado de la entrada. La estimación del modelo es libre, después de todo, la ventaja central de cálculos en redes neuronales. El costo es la inescrutabilidad del sistema. Estamos abandonados con una caja negra de cálculos no estructurada. No conocemos cual red neuronal realiza la codificación durante el entrenamiento o qué codificará u olvidará en el entrenamiento. Podemos caracterizar el comportamiento de la red solamente pasando completamente todas las entradas a través de la caja negra y grabando las salidas recordadas. La caracterización puede utilizar una sumatoria de escalares parecida al error medio al cuadrado.

Esta caracterización de caja-negra del comportamiento de la red involucra un dilema de cálculos. Por una parte, el mayor problema es el numero de casos de entrada-salida que necesitamos revisar que matemáticamente es imposible. La otra, cuando la cantidad de casos de entrada-salida es manejable, podemos almacenar también estos pares y llamarlos directamente y sin error, como buscar en una tabla. En el primer caso la red neuronal es irreal. En el segundo caso es innecesaria. Los estimadores neuronales estadísticos requieren un conjunto de muestras estadísticamente representativas. Podemos necesitar " crear " aleatoriamente estas muestras de un conjunto inicial de muestras pequeñas con técnicas sencillas o con la generación de números aleatorios de puntos agrupados cerca de las muestras originales. Ambos procedimientos de aumento de muestras suponen que el conjunto de muestras inicial representa suficientemente la probable distribución fundamental. En general no conocemos por adelantado cuan bien un conjunto de muestras dado refleja una distribución fundamental de los puntos desconocidos. Sin duda cuando la red se adapta en-línea, conocemos solamente muestras pasadas. El resto del conjunto de muestras residen en la no futura muestra. En contraste, el sistema fuzzy codifica directamente la muestra lingüística (**PESADO**, **MAS TIEMPO**) en una matriz numérica dedicada, quizás de dimensiones infinitas. La técnica de codificación por omisión es el procedimiento fuzzy **hebd**. Por practica en muchos problemas no tenemos que almacenar esta, matriz numérica, quizás infinita. En lugar de eso utilizamos un esquema de representación virtual, esto permite simplificar las entradas de puntos numéricos. Matemáticamente pasamos implícitamente un vector unitario (de un bit), o los pulsos deltas en el caso continuo, a través de la matriz de reglas **FAM**. En general describimos las entradas por una incertidumbre de la distribución probabilística o fuzzy. Tenemos que utilizar la matriz completa o reducir la entrada a un escalar promedio. Por ejemplo, si la entrada de tráfico pesada tiene 150 carros, podemos omitir la matriz **FAM**. Pero si la entrada es una curva Gaussiana con media de 150,

entonces en principio se tiene que procesar el vector de entrada en una matriz **FAM** (en la práctica podríamos utilizar solamente la media). Las dimensiones del banco **FAM** o matriz lingüística son generalmente pequeñas. Las dimensiones reflejan los niveles de cuantificación de los espacios de entrada y salida, el número de valores del conjunto fuzzy supuesto por las variables fuzzy. El enfoque fuzzy combina los enfoques puramente numéricos de redes neuronales y la modelación matemática con un enfoque simbólico de la rica estructura de la inteligencia artificial. Adquirimos conocimientos simbólicos o numéricos si utilizámos técnicas adaptables pero representados numéricamente. Las reglas **FAM** adaptables corresponden a un sentido común, frecuentemente no articulado, reglas de funcionamiento que mejoran con la experiencia. Este enfoque no abandona las técnicas de redes neuronales. En lugar de eso, los limita a parámetros no estructurados y a la estimación del estado, reconocimiento de patrones, formación de agrupaciones.

3.3.- PROYECCIONES FAMS.

Las memorias asociativas fuzzy (**FAM**) son transformaciones. Los **FAMS** proyectan conjuntos fuzzy a conjuntos fuzzy, como en la figura 3.1. En el caso más simple el sistema **FAM** consiste de una asociación única, como (**PESADO, MAS TIEMPO**). En general el sistema **FAM** consiste de un banco de diferentes asociaciones. Cada asociación corresponde a una diferente matriz numérica, o una diferente entrada en un banco **FAM**. No combinamos estas matrices como combinamos o superponemos redes neuronales de memorias asociativas, matrices (de productos exteriores). Almacenamos separadamente la matriz y el acceso a ella es en paralelo, esto evita replicas. Comenzamos con simples asociaciones **FAMS**. Para concretar asignamos el conjunto fuzzy al par (**A,B**) para codificar el control del trafico, (**PESADO, MAS TIEMPO**). Asignamos el dominio de la densidad del tráfico a las **n** variables numéricas $x_1, x_2, \ldots, x_n$ y se asigna el rango de la duración de la luz verde a las **p** variables $y_1, y_2, \ldots, y_p$. Los elementos x_i y y_j pertenecen respectivamente a los conjuntos fundamentales $X = \{ x_1, \ldots, x_n \}$ y $Y = \{ y_1, \ldots, y_p \}$, x_1 podría representar cero densidad de tráfico , y_p podría representar 10 segundos. Los conjuntos fuzzy **A** y **B** son multivalores o subconjuntos fuzzy de **X** y **Y**. Así **A** define un punto en la **n** dimensión de la unidad hipercubica $I^n = [\ 0,1\ ]^n$ y **B** define un punto en la **p** dimensión del cubo I^p. Equivalentemente, **A** y **B** definen la calidad de las funciones m_A y m_B que proyectan los elementos x_i de **X** y los y_i de **Y** a grados de calidad en [0,1]. La calidad del valor, o adaptación del valor (unidad fuzzy), indica cuánto x_i pertenece a o cae en el subconjunto **A** y cuánto y_j pertenece a **B**. Describimos esto con la función abstracta $m_A: X \rightarrow [\ 0,1\ ]$ y $m_B: Y \rightarrow [\ 0,1\ ]$. Veremos libremente los conjuntos tanto como las funciones, como puntos en el conjunto fuzzy. La

interpretación geométrica es conjuntos como puntos de los finitos conjuntos fuzzy **A** y **B**, como puntos en la unidad cubica que permiten una representación natural de vectores. Representamos **A** y **B** por los vectores numéricos $A = (a_1,..., a_n)$ y $B = (b_1,..., b_p)$, donde $a_i = m_A(x_i)$ y $b_j = m_B(y_j)$. Podemos interpretar las identificaciones **A = PESADO** y **B = MAS TIEMPO** para el problema en mano. Intuitivamente los valores de a_i deberian aumentar en la medida en que aumenta el índice **i**, quizás aproximadamente la calidad de la función sigmoide. La figura 3.2 ilustra los tres posibles subconjuntos fuzzy del universo tratado **X**.

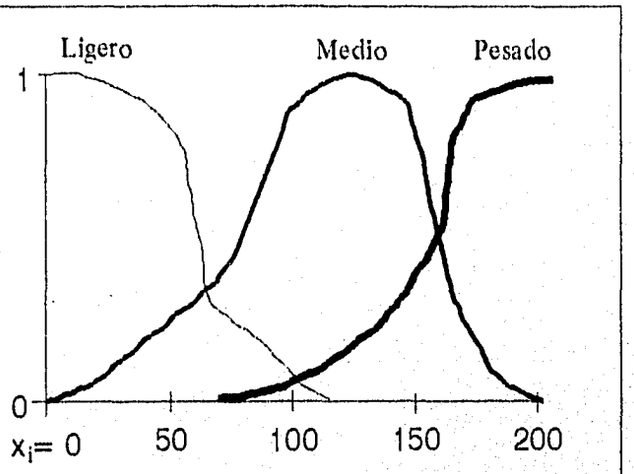


Figura 3.2.– Tres Posibles Subconjuntos de la densidad del trafico en el espacio X.

3.4.- LA MULTIPLICACION FUZZY VECTOR-MATRIZ: LA COMPOSICION MAXIMA-MINIMA.

La multiplicación fuzzy **vector-matriz** se semeja a la clásica multiplicación **vector-matriz**, pero reemplazamos el par de las multiplicaciones con el mínimo par y reemplazamos la columna (fila) sumada con la columna máxima. Denotamos esta relación como la composición fuzzy **vector-matriz**, o la relación de la composición **máxima-mínima**, por el operador de la composición "**o**". Para las filas de los correspondientes vectores **A**, **B** y la matriz fuzzy **M** de **n** por **p** (un punto en I^{nxp}):

$$A \circ M = B \qquad (3\text{-}2)$$

donde calculamos el componente b_j "recordado" tomando el fuzzy producto interior del correspondiente vector **A** con las **j** columnas de **M**:

$$b_j = \max_{1 \le i \le n} \min(a_i, m_{ij}) \qquad (3\text{-}3)$$

Se supone que los componentes del correspondiente vector son **A** = (.3 .4 .8 1) con la fuzzy matriz **M** dada como:

$$M = \begin{vmatrix} .2 & .8 & .7 \\ .7 & .6 & .6 \\ .8 & .1 & .5 \\ 0 & .2 & .3 \end{vmatrix}$$

Entonces calculamos el correspondiente vector a "recordar" **B = A o M** como:

$$\begin{aligned} b_1 &= \max\{\min(.3,.2),\min(.4,.7),\min(.8,.8),\min(1,0)\} \\ &= \max(.2, .4, .8, 0) \\ &= .8 \\ b_2 &= \max(.3, .4, .1, .2) \\ &= .4 \\ b_3 &= \max(.3, .4, .5, .3) \\ &= .5 \end{aligned}$$

Por lo tanto **B = (.8 .4. 5)**. Si codificamos de algún modo (**A**,**B**) en la matriz **M**, diríamos que el sistema **FAM** exhibe un perfecto recuerdo en la dirección delantera: **A o M = B**. La interpretación neuronal de la composición **máxima-mínima** es que cada neurona en el campo $\mathbf{F_Y}$ (o campo $\mathbf{F_B}$) genera un valor señal/activación para la composición lineal fuzzy. El paso de información hacia atrás a través de $\mathbf{M^T}$ nos permite interpretar el sistema fuzzy como una memoria asociativa bidireccional (**BAM**). El teorema del **FAM** bidireccional caracteriza exitosamente el recuerdo **BAM** por la correlación fuzzy o aprendizaje **Hebbian.** Como complemento también se menciona el operador de la composición **máximo-producto**, que reemplaza el mínimo producto en (8-2):

$$b_j = \max_{1 \le i \le n} a_i m_{ij} \qquad (3\text{-}4)$$

La literatura fuzzy frecuentemente confunde este operador de composición con el esquema fuzzy de la codificación de la correlación. La composición del **máximo-producto** es un método para " multiplicar " matrices fuzzy o vectores. La

correlación fuzzy también utiliza productos pares para ajustar valores, en la construcción de matrices.

3.5.- FUZZY FAMS HEBB.

La mayoría de los sistemas fuzzy encontrados en aplicaciones son los fuzzy **FAMS Hebb**. Son sistemas fuzzy **S**: $I^n \rightarrow I^p$, construidos de una manera simple similar a los neuronales, en la teoría de redes neuronales interpretamos la hipótesis de la clásica correlación Hebbian del aprendizaje sinaptico como un aprendizaje no supervisado con el producto de las señales S_iS_j:

$$\dot{m}_{ij} = -m_{ij} + S_i(x_i)S_j(y_j) \qquad (3\text{-}5)$$

Para un par dado de filas de los vectores bipolares (**X**,**Y**), la interpretación neuronal da la matriz de correlación del **producto-exterior**:

$$M = X^TY \qquad (3\text{-}6)$$

Se definen puntos en la matriz fuzzy **hebb** para la mínima de las "señales" $\mathbf{a_i}$ y $\mathbf{b_j}$, un esquema de codificación llamado codificación de la **mínima-correlacion**:

$$m_{ij} = \min(a_i, b_j) \qquad (3\text{-}7)$$

Dada en notación de matrices como el fuzzy **producto-exterior**:

$$M = A^T \circ B \qquad (3\text{-}8)$$

Mandani (1977) y Togai (1986) independientemente llegaron a la norma fuzzy **hebbian** (3-7) como un operador de multivalores de implicación lógica: **verdadero** $(\mathbf{a_i} \rightarrow \mathbf{b_i}) = \mathbf{min}(\mathbf{a_i}, \mathbf{b_j})$. El mínimo operador, no obstante, es un operador lógico simétrico. De modo que no generaliza adecuadamente la implicación clásica $\mathbf{P} \rightarrow \mathbf{Q}$, que es falsa si y solamente si el antecedente **P** es verdadero y el consecuente **Q** es falso, $\mathbf{t}(\mathbf{P}) = 1$ y $\mathbf{t}(\mathbf{Q}) = 0$. En contraste, esto es semejante a definir una matriz de puntos "condición-posibilidad" con implicación de

valores continuos, que llevo a Zadeh[2] a escoger el operador de implicación Lukasiewicz: m_{ij} = **verdadero**($a_i \rightarrow b_j$) = **min**(1,1 $-a_i + b_j$). Lamentablemente el operador Lukasiewicz es generalmente igual o aproximadamente unitario, para el **min**(1,1 $-a_i+b_j$)< 1 **si** $a_i > b_j$, la mayoría de las entradas de la resultante matriz **M** son unitarias o aproximadamente unitarias. Esto ignora la información en la asociación (**A**,**B**). Así **A`** o **M** tiende a igualarse al mayor valor de $a_k$ para cualquier entrada del sistema **A`**. Se construye una autoasociativa matriz fuzzy **FAM Hebb** para codificar el redundante par (**A**,**A**) en (3-8) como la matriz fuzzy de autocorrelación:

$$M = A^T \circ A \qquad (3\text{-}9)$$

En el ejemplo previo la matriz **M** fue tal que la entrada **A** = (.3 .4 .8 1) del vector recordado **B** = (.8 .4 .5) sobre la composición **máxima-mínima**: **A** o **M** = **B**. Se dispone de **A** todavía para recordar **B** si reemplazáramos la original matriz **M** con la matriz fuzzy **Hebb** encontrada con (3-8). Sustituyendo **A** y **B** en (3-8) da:

$$M = A^T \circ B = \begin{vmatrix} .3 \\ .4 \\ .8 \\ .1 \end{vmatrix} \circ (.8 \ .4 \ .5) = \begin{vmatrix} .3 & .3 & .3 \\ .4 & .4 & .4 \\ .8 & .4 & .5 \\ .8 & .4 & .5 \end{vmatrix}$$

Esta matriz **M** fuzzy **hebb** ilustra dos propiedades claves. Primero, las **i** filas de **M** son iguales al mínimo par de a_i y la asociación de salida **B**. Simétricamente, las **j** columnas de **M** son igual al mínimo par de b_j y la asociación de entrada **A**:

$$M = \begin{vmatrix} a_1 \hat{} B \\ \vdots \\ a_n \hat{} B \end{vmatrix} \qquad (3\text{-}10)$$

$$= [b_1 \hat{} A^T \ ^3 \ldots ^3 b_m \hat{} A^T] \qquad (3\text{-}11)$$

Donde el operador de gorro denota el mínimo par: $a_i \hat{} b_j$ = **min**(a_i,b_j). El término $a_i \hat{} B$ indica el mínimo componente:

2 Zadeh L. A. (1983), "A computation Approach to Fuzzy Quantifiers in Natural Languages", Computer and Mathematics, vol., 9, pp., 149-184.

$$a_i \wedge B = (a_i \wedge b_1, \ldots, a_i \wedge b_n) \qquad (3\text{-}12)$$

Por lo tanto si algún $a_k = 1$, entonces la **k** fila de **M** es igual a **B**. Si alguna $b_i = 1$, la **l** columna de **M** es igual a **A**. Más generalmente, si alguna a_k es al menos tan grande como cada b_j, entonces la **k** fila de la matriz **M** fuzzy **hebb** es igual a **B**. Segundo, las terceras y cuartas filas de **M** son igual al correspondiente vector **B**. Sin embargo no es igual a las columnas de **A**. Esto permite un perfecto recuerdo en la dirección hacia delante, **A o M = B**, pero no en la dirección hacia atrás, $B \circ M^T \neq A$:

$$A \circ M = (8. \ .4 \ .5) = B$$
$$B \circ M^T = (.3 \ .4 \ .8 \ .8) = A' \subset A$$

A' es un adecuado subconjunto de **A**: $A' \neq A$ y $S(A', A) = 1$, donde **S** mide el grado del subconjunto **A'** en **A**. En otras palabras, $a'_i \leq a_i$ para cada **i** y $a'_k < a_k$ para al menos una **k**. Si $B' = A \circ M$ difiere de **B**, entonces **B'** es un subconjunto adecuado de **B**. Entonces se proyectan subconjuntos fuzzy a subconjuntos fuzzy.

3.6.- EL TEOREMA DEL FAM BIDIRECCIONAL PARA LA CODIFICACION DE LA CORRELACION-MINIMA.

El análisis del recuerdo **FAM** utiliza la tradicional altura del conjunto fuzzy y la perpendicularidad del conjunto. La altura **H(A)** del conjunto **A** es el valor máximo correspondiente de **A**:

$$H(A) = \max_{1 \leq i \leq n} a_i \qquad (3\text{-}13)$$

El conjunto fuzzy **A** es perpendicular si **H(A) = 1**, si al menos un valor de a_k es máximo: $a_k = 1$. En la práctica los conjuntos fuzzy son generalmente perpendiculares. Podemos extender un conjunto fuzzy no perpendicular a un conjunto fuzzy perpendicular añadiendo una dimensión ficticia con el correspondiente valor $a_{n+1} = 1$.

La precisión del cuerdo en los fuzzy **Hebb** construidos con la codificación de la **correlación-mínima** depende de las alturas **H(A)** y **H(B)**. El conjunto fuzzy perpendicular exhibe el perfecto recuerdo. Sin duda **(A,B)** es un punto fijo bidireccional, **A o M = B**, y $B \circ M^T = A$, si y solamente **H(A) = H(B)**, que se mantienen siempre que **A** y **B** sean perpendiculares.

El teorema del **FAN** bidireccional para la **correlación-mínima** es:

Si $M = A^T \circ B$, entonces:

$$\begin{array}{lll} \text{(i)} & A \circ M & = B \text{ si } H(A) \geq H(B) \\ \text{(ii)} & B \circ M^T & = A \text{ si } H(B) \geq H(A) \\ \text{(iii)} & A` \circ M & \subset B \text{ para cualquier } A` \\ \text{(iv)} & B` \circ M^T & \subset A \text{ para cualquier } B` \end{array}$$

Prueba. Observe que la altura **H(A)** es igual al fuzzy perpendicular de **A**:

$$A \circ A^T = \max_i a_i \wedge a_i = \max_i a_i = H(A)$$

Entonces:

$$\begin{aligned} A \circ M &= A \circ (A^T \circ B) \\ &= (A \circ A^T) \circ B \\ &= H(A) \circ B \\ &= H(A) \wedge B \end{aligned}$$

Por lo tanto $H(A) \wedge B = B$, si $H(A) \geq H(B)$, establecido por (i). Ahora suponga que $A`$ es un vector arbitrario en I^n. Entonces:

$$\begin{aligned} A` \circ M &= (A` \circ A^T) \circ B \\ &= (A` \circ A^T) \wedge B \end{aligned}$$

Lo cual establece (iii) desde que $A` \circ A^T \leq H(A)$. Un argumento similar es utilizando para $M^T = B^T \circ A$ establecido por (ii) y (iv).
La igualdad $A \circ A^T = H(A)$ implica un inmediato colorario del teorema del **FAM** bidireccional. Un superconjunto $A` \supset A$ se comporta de la misma forma que la asociación de entrada codificada **A**: $A` \circ M = B$ si $A \circ M = B$. El fuzzy **hebb** ignora la información en la diferencia $A` - A$, cuando $A` \subset A`$.

3.6.1.- CODIFICACION CORRELACION-PRODUCTO.

La codificación **correlación-producto** provee un alternativo esquema de codificación fuzzy **Hebbian.** El producto exterior matemático estándar de los vectores **A** y **B** forman la matriz **M.** Entonces:

$$m_{ij} = a_i \, b_j \qquad (3\text{-}14)$$

y en notación de matrices:

$$M = A^T \, B \qquad (3\text{-}15)$$

de modo que las **i** filas de **M** son iguales a la amplitud del conjunto fuzzy $\mathbf{a_i B}$ y las **j** columnas de **M** son iguales a $\mathbf{b_j A^T}$:

$$M = \begin{vmatrix} a_1 B \\ \cdot \\ \cdot \\ \cdot \\ a_n B \end{vmatrix} \qquad (3\text{-}16)$$

$$= [\, b_1 A^T \,|\, \ldots \,|\, b_m A^T] \qquad (3\text{-}17)$$

Si **A** = (.3 .4 .8) y **B** = (.8 .4 .5) como antes y se codifica la regla **FAM** (**A**,**B**) con la **correlación-producto** en la siguiente matriz **M:**

$$M = \begin{vmatrix} .24 & .12 & .15 \\ .32 & .16 & .2 \\ .64 & .32 & .4 \\ .8 & .4 & .5 \end{vmatrix}$$

Se observa que si $\mathbf{A'}$ = (0 0 0 1), entonces $\mathbf{A' \circ M = B}$. El sistema **FAM** recuerda la asociación de salida **B** al máximo grado. Si $\mathbf{A'}$ = (1 0 0 0), entonces $\mathbf{A' \circ M}$ = (.24 ,12 ,15). El sistema recordara la salida **B** solamente a el grado .3. La codificación **correlación-mínima** produce una matriz del conjunto **B** abreviada, mientras la codificación **correlación-producto** produce una matriz del conjunto **B** ampliada. En el argumento calidad-función, todos los conjuntos fuzzy $\mathbf{a_i B}$ ampliados tienen la misma forma que **B**. EL conjunto fuzzy $\mathbf{a_i \hat{} B}$ abreviado es flotante en o sobre el valor de $\mathbf{a_i}$. En este

sentido la codificación **correlación-producto** preserva más información que la codificación **correlación-mínima**, en la práctica y en el desarrollo de aplicaciones fuzzy, el conjunto de entrada fuzzy $\mathbf{A}'$ es un vector binario con un **1** y todos los demás elementos **0**, una fila de la matriz identidad de **n** por **n** (o un pulso delta en el caso continuo). $\mathbf{A}'$ representa la ocurrencia del rizo medido del dato $\mathbf{x_i}$, como una densidad de tráfico con valor de 30. Cuando se aplica la codificación de la regla **(A,B)**, el valor medido de $\mathbf{x_i}$ activa **A** a el grado $\mathbf{a_i}$. Esto es parte del proceso del recuerdo de la composición **máxima-mínima**, para $\mathbf{A}' \circ \mathbf{M} = (\mathbf{A}' \circ \mathbf{A}^T) \circ \mathbf{B} = \mathbf{a_i} \wedge \mathbf{B} \circ \mathbf{a_i B}$, que depende de si codificamos **(A,B)** en **M** con la codificación **correlación-producto** o **correlación-mínima.** Activamos o "encendemos" la asociación de salida **B** a el grado $\mathbf{a_i}$. Por lo tanto los valores de $\mathbf{a_i}$ son binarios, $\mathbf{a_i m_{ij}} = \mathbf{a_i} \wedge \mathbf{m_{ij}}$, de modo que los operadores de la composición **máxima-mínima** y **máximo-producto** coinciden. Evitamos esta confusión refiriendonos a ambos, en el proceso del recuerdo y el esquema de la codificación de la correlación como la inferencia **correlación-mínima** cuando combinamos la codificación **correlación-mínima** con la composición **máxima-mínima** y como la inferencia **correlación-producto** cuando combinamos la codificación **correlación-producto** con la composición **máxima-mínima.** Ahora se muestra la versión del teorema del **FAM** bidireccional para la **correlación-producto.** Si $\mathbf{M} = \mathbf{A}^T\mathbf{B}$, **A** y **B** son vectores no nulos, entonces:

$$
\begin{array}{lllll}
\text{(i)} & A \circ M & = & B & \text{si } H(A) = 1 \\
\text{(ii)} & B \circ M^T & = & B & \text{si } H(B) = 1 \\
\text{(iii)} & A' \circ M & \subset & B & \text{para alguna } A' \\
\text{(iv)} & B' \circ M^T & \subset & A & \text{para alguna } B'
\end{array}
$$

Prueba:

$$
\begin{aligned}
A \circ M &= A \circ (A^T B) \\
&= (A \circ A^T) B \\
&= H(A) B
\end{aligned}
$$

Ya que **B** no es un conjunto vació, $\mathbf{H(A)B = B}$, si $\mathbf{H(A) = 1}$, establecido por (i), ($\mathbf{A \circ M = B}$ se mantiene trivialmente si **B** es el conjunto vació). Para un vector arbitrario $\mathbf{A}'$ en $\mathbf{I^n}$:

$$
\begin{aligned}
A' \circ M &= (A' \circ A^T) B \\
&\subset H(A) B \\
&\subset B
\end{aligned}
$$

Por lo tanto $A^{`}$ o $A \leq H(A)$, establecido por (iii), (ii) y (iv), y se sigue utilizando la igualdad $M^T = B^T A$.

3.7.- RECORDAR SALIDAS Y LA DEFUZZIFICACION.

El recuerdo del vector de salida **B** es igual a una suma de cargas del individual vector recordado:

$$B = \sum_{k=1}^{m} w_k B'_k \qquad (3\text{-}18)$$

Donde la carga no negativa w_k aumenta la credibilidad o fortaleza de la **K** regla (A_k, B_k). La credibilidad de las cargas w_k son candidatas inmediatas para la modificación adaptable. En la práctica se escoge $w_1 = \ldots = w_m = 1$ como un valor por omisión. En principio, aunque no en la práctica, el recuerdo del vector de salida es igual a una suma normalizada de los vectores $B^{`}_k$. Estos mantienen los componentes de **B** en un intervalo unitario. No se utiliza la normalización en la práctica porque invariablemente se **"defuzzifica"** la distribución de salida **B** para producir una salida numérica única, un valor único en el universo de salidas tratado $Y = \{y_1, \ldots, y_p\}$. La información de salida **B** reside ampliamente en los valores relativos de la calidad de los rangos. El más simple esquema de la defuzzificacion escoge este elemento y_{max} que tiene la máxima calidad en el conjunto de salida **B**:

$$m_B(y_{max}) = \max_{1 \leq j \leq k} m_B(y_j) \qquad (3\text{-}19)$$

Los métodos probabilísticos populares de estimación de parámetros de **máxima-probabilidad** y **máximos-posteriores** motiva este esquema de defuzzificación de la **máxima-calidad**. El esquema de defuzzificación de la **máxima-calidad** tiene dos problemas fundamentales. Primero, el modo de la distribución de **B** no es única. Este problema afecta la codificación **correlación-mínima**. Como muestra la representación (3-10), más que afectar la codificación **correlación-producto**. Por lo tanto el operador mínimo se une fuera de los principales vectores adaptados B_k, la aditiva combinación del vector de salida adaptado **B** que tiende a estar de bajo de muchas regiones del universo tratado **Y**. Para la calidad de funciones esto conduce a muchos modos infinitos. Hasta para cuantificar el conjunto fuzzy, pueden existir muchos modos. En la práctica podemos calcular modos múltiples, para muchos bancos **FAM** de reglas "independientes", alguna forma del teorema del límite central (cuya prueba depende finalmente de la transformada de Fourier, que no es probable). La forma de la

señal de B tiende a ser semejante a la calidad de una función Gaussiana. Por lo tanto un modo único tiende a emerger. Tiende a emerger con menos muestras si utilizáramos la codificación **correlación-producto**. Segundo, el esquema de la **máxima-calidad** ignora la información en gran parte de la forma de la señal de B, la codificación **correlación-mínima** arregla el problema. En la práctica B es frecuentemente altamente asimétrico, aunque es unimodal. Muchas distribuciones de salida pueden compartir el mismo modo, la alternativa natural es el esquema fuzzy del **centroide** de la defuzzificación. Se calcula directamente el valor real de la salida como una (normalizada) combinación convexa de los valores, el **centroide** fuzzy del vector B respecto al espacio de salida **Y**:

$$\bar{B} = \frac{\sum_{j=1}^{p} y_j \; m_B(y_j)}{\sum_{j=1}^{p} m_B(y_j)} \qquad (3\text{-}20)$$

El **centroide** fuzzy es único y utiliza toda la información en la distribución de salida. Para las simétricas distribuciones unimodales el modo y el **centroide** fuzzy coincide. En muchos casos tenemos que reemplazar las sumas discretas en (3-20) con integrales sobre espacios continuos infinitos, se piensa, que para bibliotecas de conjunto fuzzy de valores trapezoidales podemos reemplazar tal razón de integrales con una razón de sumas discretas simples. Calculando el **centroide** con (3-20) es el único paso en el procedimiento de inferencia **FAM** que requiere la división. Todas las demás operaciones son productos interiores, mínimo par y adiciones.

3.8.- LA ARQUITECTURA DEL SISTEMA FAM.

La figura 3.3 esquematiza la arquitectura del sistema **FAM** no lineal **F**. Observe que **F** proyecta conjuntos fuzzy a conjuntos fuzzy: **F(A) = B**. Por lo tanto define un sistema fuzzy de transformación $\mathbf{F}: I^n \rightarrow I^p$. En la práctica **A** es igual a un vector de un bit con un valor unitario, $a_i = 1$ y todos los demás valores son cero, $\mathbf{a_j = 0}$ o un pulso delta. Se defuzzifica el conjunto de salida **B** con la técnica del **centroide** para producir un elemento exacto $\mathbf{y_j}$ en el universo **Y** de salida. En efecto la defuzzificación produce un vector binario de salida **O**, de nuevo con un elemento **1** y el resto ceros. A este nivel el sistema **FAM F** proyecta conjuntos fuzzy a conjuntos fuzzy, reduciendo el sistema fuzzy a una proyección entre cubos Booleanos, $\mathbf{F}: \{0,1\}^n \rightarrow \{0,1\}^p$. En muchas aplicaciones se modela **X** e **Y** como universos continuos. Por lo tanto **n** y **p** son bastante grandes.

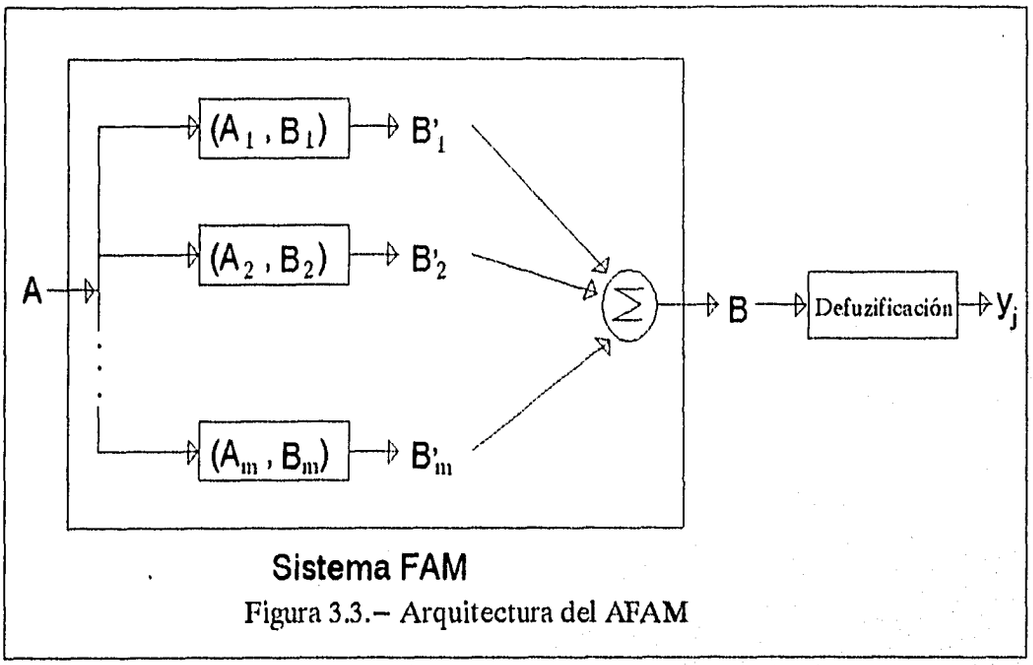


Figura 3.3.– Arquitectura del AFAM

3.9.- SISTEMAS DE CONTROL FUZZY Y NEURONALES.

En esta sección se muestra un sistema de control fuzzy y neuronal simulando el retroceso de un camión y de un camión con remolque, en un muelle de carga en una parte del estacionamiento. Se utiliza la diferencial del aprendizaje competido y la técnica de agrupación del **producto-espacio**, se genera una adaptable memoria asociativa (**FAM**), se forman reglas **FAM** de los datos para las simulaciones fuzzy y neuronales. Se desarrollan los sistemas neuronales del camión del diseño propuesto por Nguyen y Widrow[3]. Se forma el sistema neuronal del camión con el algoritmo de aprendizaje **backpropagation**. En principio la agrupación **producto-espacio** pudo convertir algunos sistemas neuronales de caja negro en una colección representativa de reglas **FAM**.

3.9.1.- PARTE SUPERIOR DEL CAMION.

La figura 3.4 muestra el camión y la zona de carga. El camión corresponde al camión neuronal en el sistema Nguyen-Widrow. Las tres variables declaradas ϕ, **x** y **y** exactamente determinan la posición del camión, ϕ especifican el ángulo del camión con la horizontal. El par de coordenadas (**x**,**y**) especifican la

3 Naguyen D., and Widrow B. (1989), "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks", vol., II, p.p. 357-363, June.

posición central del camión en el plano. La meta es marcar la llegada del camión al muelle de carga a un ángulo recto de ($\phi_f = 90^\circ$) y alinear la posición (**x**,**y**) del camión con la entrada del deseado muelle de carga (x_f,y_f). Se considera solamente la parte superior del camión. El camión se mueve hacia atrás a unas distancias fijas a cada etapa. La zona de carga corresponde al plano [0,100] x [0,100], y (x_f,y_f) se igualaron a (50,100). A cada etapa los controladores fuzzy y neuronal tienen que producir el ángulo θ de la dirección del camión al muelle de carga de alguna posición inicial y de algún ángulo en la zona de carga.

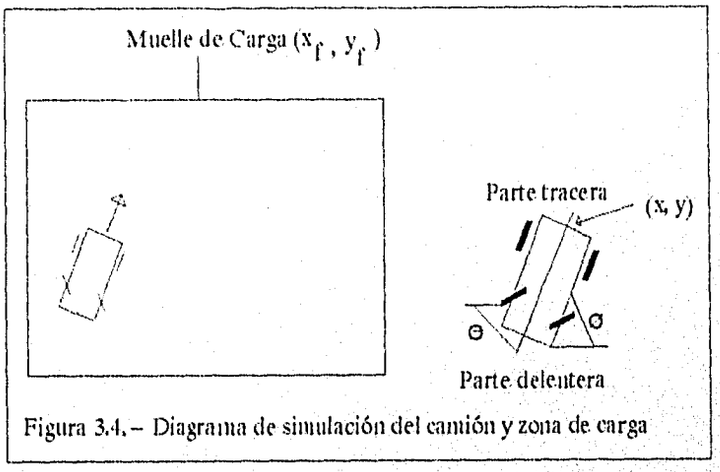


Figura 3.4.– Diagrama de simulación del camión y zona de carga

3.9.2.- SISTEMA CONTROLADOR FUZZY DEL CAMION.

Primero se especifica cada entrada del controlador y las variables de salida. Las variables de entrada son el ángulo del camión ϕ y la **x** posición, la coordenada **x**. La variable de salida es el ángulo de la dirección, la señal θ. Se supuso bastante espacio libre entre el camión y el muelle de carga tanto que podemos desconocer la posición de la coordenada **y**. El rango de las variables es como sigue:

$$\begin{aligned} 0 &\leq x \leq 100 \\ -90^\circ &\leq \phi \leq 270^\circ \\ -30^\circ &\leq \theta \leq 30^\circ \end{aligned}$$

Los valores positivos de θ representan las rotaciones del volante según las manecillas del reloj. Los valores negativos representan rotaciones contrarias a las manecillas del reloj. Se discretizan todos los valores reduciendo los cálculos. La resolución de ϕ y θ es a cada grado. La resolución de **x** es a 0.1. Después se especifica el conjunto de valores de las variables fuzzy de la entrada y salida. Los conjuntos fuzzy numéricamente representaron términos lingüísticos, la clase de los términos lingüísticos pueden ser utilizados para

describir la conducta del sistema controlador. Se eligieron los valores del conjunto fuzzy como sigue:

Angulo ϕ	Posición x	Angulo θ
RB:Bajo Derecho	LE:Izquierda	NB:Negaivo Grande
RU:Superior Derecha	LC:Centro Izquierda	NM:Negativo Mediano
RV:Vertical Derecha	CE:Centro	NS:Negativo Pequeño
VE:Vertical	RC:Centro Derecha	ZE:Cero
LV:Vertical Izquierda	RI:Derecho	PS:Positivo Pequeño
LU:Superior Izquierda		PM:Positivo Mediano
LB:Bajo Izquierdo		PB:Positivo Grande

El subconjunto fuzzy contiene los elementos con las calidades de los rangos. **A** es una función fuzzy m_A: **z** $\rightarrow$ [0,1], que le asigna un número real entre 0 y 1 a cada elemento **z** en el universo **Z** tratado. Este número $m_A(\mathbf{z})$ indica el grado a que el objeto o dato **z** pertenecen al conjunto fuzzy **A**. Equivalentemente, $m_A(\mathbf{z})$ define el apropiado valor (unidad fuzzy)[4] del elemento **z** en **A**. La función fuzzy puede tener formas diferentes dependiendo de la preferencia del diseñador o experiencia. En la práctica en construcciones fuzzy se ha encontrado que han funcionado mejor las formas triangulares y trapezoidales en la captura del modelo en el sentido de números fuzzy y simplifican los cálculos. La figura 3.5 muestra la gráfica de las funciones de los subconjunto.

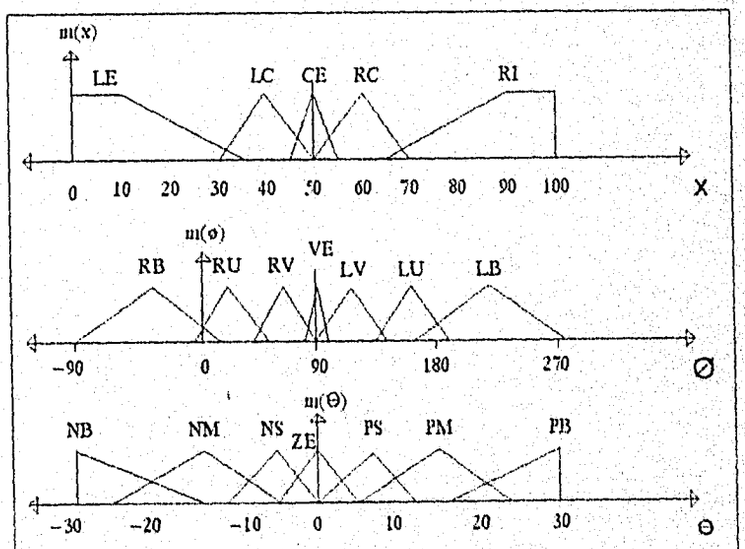


Figura 3.5.- Funciones para los conjuntos fuzzy.

4 Kosko B., (1986), "Fuzzy Entropy and Conditioning", Information Sciences, vol. 40, p.p. 165-174.

En la tercera gráfica, por ejemplo, $\theta = 20$ es un Medio Positivo el grado es 0.5, pero solamente es Positivo Grande a el grado 0.3. En la figura 3.5 los conjuntos fuzy **CE**, **VE** y **ZE** son más estrechos que los otros conjuntos fuzzy. Estos estrechos conjuntos fuzzy permiten un control fino cerca del muelle de carga. Se utilizan conjuntos fuzzy más amplios para describir los puntos finales para el rango de las variables fuzzy ϕ, **x**, y θ. Los conjunto más amplios permitieron un control más accidentado del muelle de carga. Después se especifica las "reglas" fuzzy o banco de reglas fuzzy, la memoria asociativa (**FAM**). Las asociaciones fuzzy o "reglas" (**A**,**B**), asocian el conjunto fuzzy **B** de salida, los valores de control con el conjunto fuzzy **A** de entrada, los valores de las variables de entrada. Se pueden escribir asociaciones fuzzy como un par antecedente-consecuente o declaraciones **SI_ENTONCES.** En el caso del camión, el banco **FAM** contuvo las 33 reglas de la figura 3.6.

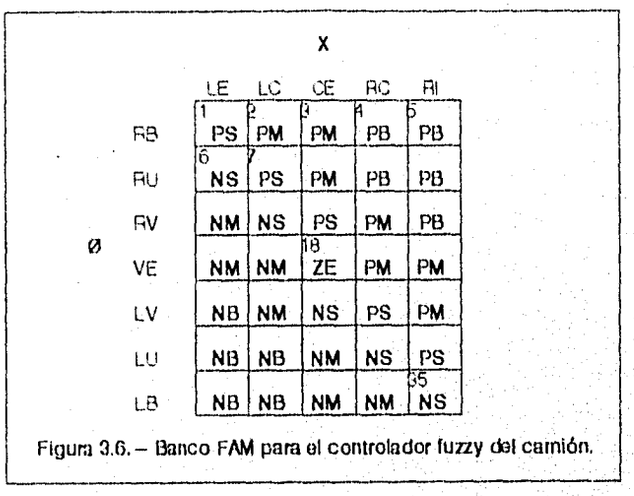


Figura 3.6.– Banco FAM para el controlador fuzzy del camión.

Por ejemplo, la regla **FAM** del bloque izquierdo (regla 1) corresponde a la siguiente asociación fuzzy.

$$\text{SI } x = \text{LE Y } \phi = \text{RB, ENTONCES } \theta = \text{PS}$$

La regla **FAM** 18 indica que si el camión esta cerca de la posición de equilibrio, entonces el controlador no tiene que producir una señal de ángulo positivo o negativo. Las reglas **FAM** en la matriz del banco **FAM** refleja la simetría del sistema controlador.
Para la condición inicial **x** = 50 y ϕ = 270, el camión fuzzy no opero bien. La simetría de las reglas **FAM** y los conjuntos fuzzy cancelaron la salida del controlador fuzzy en un raro punto del puerto. Para esta condición inicial, el controlador neuronal (camión-remolque) también funciono mal. Alguna perturbación quiebra la simetría. Por ejemplo, la regla (**SI x = 50 Y ϕ = 270, ENTONCES θ = 5**) corrigió el problema. En esta simulación el procedimiento **FAM** de la **correlación-mínima**, determinó la superficie de control. Si la superficie de

control cambia con los valores de las variables, el sistema se comporta como un adaptable controlador fuzzy. Después se muestra un controlador adaptable no supervisado del camión y del camión-remolque. Finalmente, se determina la acción de la salida, dada la condición de entrada. Se utilizo el método de inferencia de la **correlación-mínima** ilustrado en la figura 3.7.

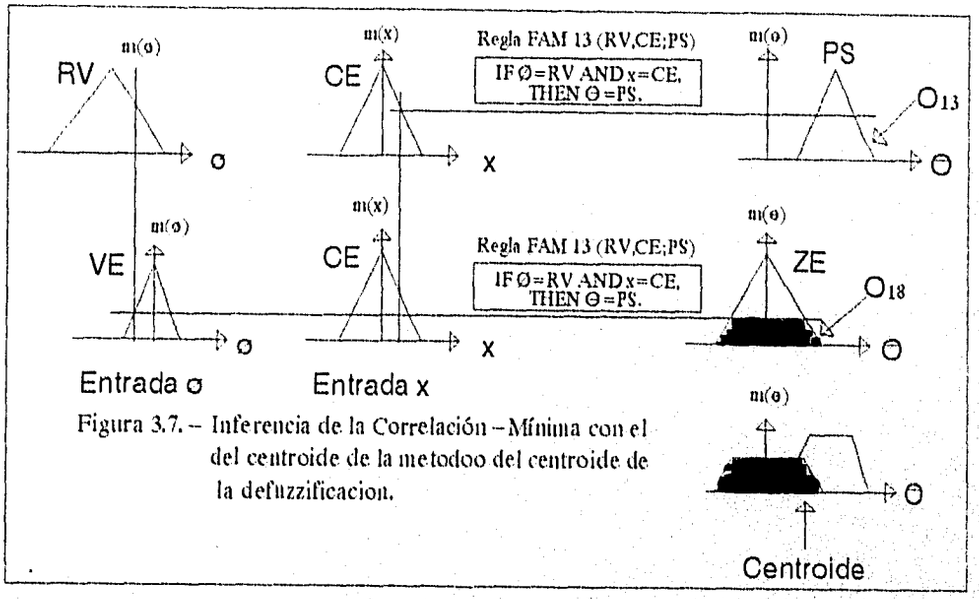


Figura 3.7.– Inferencia de la Correlación –Mínima con el del centroide de la metodoo del centroide de la defuzzificacion.

Cada regla **FAM** produjo el conjunto fuzzy de salida, que asegura el grado de ingreso determinado por la condición de entrada y la regla **FAM**. Alternativamente, la inferencia **correlación-producto** combinaría las reglas **FAM** multiplicandolas. Cada regla **FAM** emite una apropiada carga de salida del conjunto fuzzy O_i a cada iteración. La salida total O agregó estas cargas a la salida:

$$O = \sum_i O_i \qquad (3\text{-}21)$$

$$= \sum_i \min(f_i, S_i) \qquad (3\text{-}22)$$

donde f_i denota el valor del antecedente y S_i representa el consecuente conjunto fuzzy del valor del ángulo de la dirección en las **i** reglas **FAM**. Primero el sistema fuzzy combinara el conjunto de salida O_i con el máximo par. Pero esto tiende a producir un uniforme conjunto de salida **O** como el número de reglas **FAM** crezca. Se agrego el conjunto de salida O_i que invoca la versión fuzzy del teorema del límite central. Esto tiende a producir un simétrico, unimodal conjunto de salida fuzzy **O** del valor del ángulo de la dirección. La proyección del sistema fuzzy, es conjuntos fuzzy a conjuntos fuzzy. La salida del sistema fuzzy define el conjunto **O** el valor del ángulo a cada iteración. Tenemos

que "defuzzificar" el conjunto fuzzy O para producir un numérico (punto estimado) del valor del ángulo θ de salida. La simple defuzzificación proyecta el valor elegido correspondiente al apropiado valor máximo en el conjunto fuzzy. Este modo de selección de aproximación desconoce la mayor parte de la información del conjunto fuzzy de salida y requiere un algoritmo adicional de decisión cuando modos múltiples ocurren. El **centroide** de la defuzzificación provee un procedimiento más eficaz. Este método utiliza el **centroide** fuzzy como la salida:

$$\bar{\theta} = \frac{\sum_{j=1}^{p} \theta_j \, m_O(\theta_j)}{\sum_{j=1}^{p} m_O(\theta_j)} \qquad (3\text{-}23)$$

donde O define un subconjunto fuzzy del universo del ángulo, $\theta = \{\theta_1,\ldots,\theta_p\}$. El teorema del limite central en efecto es producido al agregar el conjunto de salida fuzzy O_i que beneficia a ambos, al modo máximo y el **centroide** de la defuzzificación. La Figura 3.7 muestra la inferencia de la **correlación-mínima** y el **centroide** de la defuzzificación aplicado a las reglas **FAM** 13 y 18. Se utiliza el **centroide** de la defuzzificación en toda la simulación. Con 35 reglas **FAM**, el controlador fuzzy del camión produjo exitosamente las trayectorias del camión de alguna posición inicial. La figura 3.8 muestra un ejemplo típico de las trayectorias del camión de diferentes posiciones iniciales. El sistema controlador fuzzy no utilizo todas las reglas **FAM** a cada iteración. Equivalentemente la mayoría de colecciones de salida consiguientes son vacías. En la mayoría de casos el sistema usó solamente una o dos reglas **FAM** a cada iteración. La mayoría de sistemas utilizan 4 reglas **FAM** algunas veces.

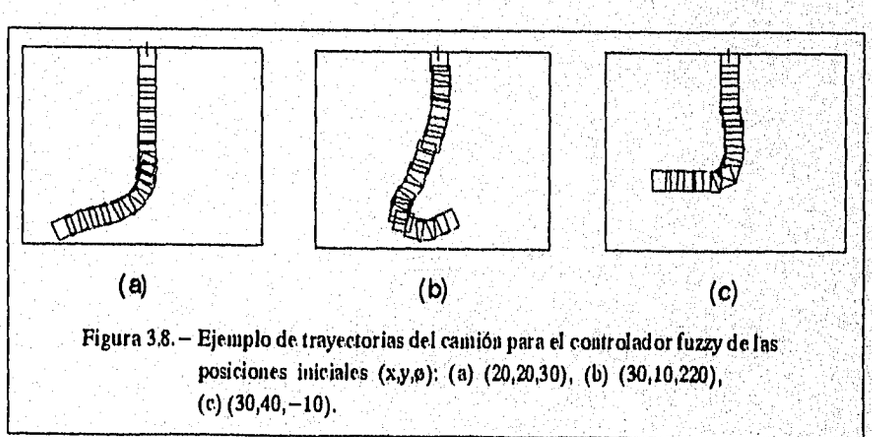


Figura 3.8.– Ejemplo de trayectorias del camión para el controlador fuzzy de las posiciones iniciales (x,y,ø): (a) (20,20,30), (b) (30,10,220), (c) (30,40,–10).

3.9.3.- SISTEMA NEURONAL DEL CAMION.

El sistema neuronal del camión de Nguyen y Windrow consistió de una red neuronal de múltiples capas no realimentada con el algoritmo **backpropagation**. El sistema neuronal de control consistió de dos redes neuronales: la red del controlador y la red de emulación del camión. La red del controlador produjo un ángulo apropiado de la señal de salida dada alguna coordenada (**x**,**y**) del estacionamiento y el ángulo ϕ. La red emuladora calculo la próxima posición del camión. La red emuladora llevó como entrada la posición anterior del camión y el ángulo de salida de la dirección calculado por la red controladora. La red emuladora no pudo obtener las "universales" cargas sinapticas de conexión. El algoritmo **backpropagation** no convergió para algunos conjuntos de muestras. El número de muestras para la red emuladora puede exceder las 3000. Por ejemplo, las combinaciones de un ángulo de prueba ϕ, **x**-posición, **y**-posición, y el ángulo θ de la dirección pueden corresponder a 3150 (18 x 5 x 5 x 7) muestras, dependiendo de la división del producto del espacio de la entrada-salida. Además, las muestras de instrucciones eran numéricamente semejantes, ya que para las señales neuronales se supuso valores escalares en [0,1] o [-1,1]. Por ejemplo, se trataron valores cercanos, tales como 0.40 y 0.41, como muestras de distintos valores. Sencillas ecuaciones cinemáticas reemplazaron la red emuladora del camión. Si el camión se movió hacia atrás de (x,y) a (x',y') a cada iteración, entonces:

$$x' = x + r\cos(\acute{Y}') \qquad (3\text{-}24)$$

$$y' = y + r\sin(\acute{Y}') \qquad (3\text{-}25)$$

$$\phi' = \phi + \theta \qquad (3\text{-}26)$$

r denota la distancia fija de conducción del camión para todos los movimientos. Se utilizaron las ecuaciones (3-24)-(3-26) en vez de la red emuladora. Esto no afectó la ejecución de la post-instrución del camión neuronal, ya que la red emuladora del camión solamente backpropagated errores. Se formo solamente la red del controlador con el **backpropagation**. La red del controlador utilizo 24 neuronas "ocultas" con la función sigmoide. En la formación del controlador del camión, se estima el ángulo ideal de la señal a cada etapa antes de formar la red del controlador. En la simulación, se utilizan las trayectorias del camión formadas por el controlador fuzzy como la trayectoria ideal. El controlador fuzzy genero cada prueba formando (x, y, ϕ, θ) cada iteración del proceso. Se utilizaron 35 muestras del vector de instrucciones y se necesitaron más que 100,000 iteraciones para adiestrar la red del controlador. La superficie del controlador neuronal tiene una estructura menor que la correspondiente superficie del controlador fuzzy. Esto refleja la naturaleza antiestructural, de caja-

negra del aprendizaje supervisado. La figura 3.9 muestra la conexión topologica de la red para nuestro controlador neuronal del camión.

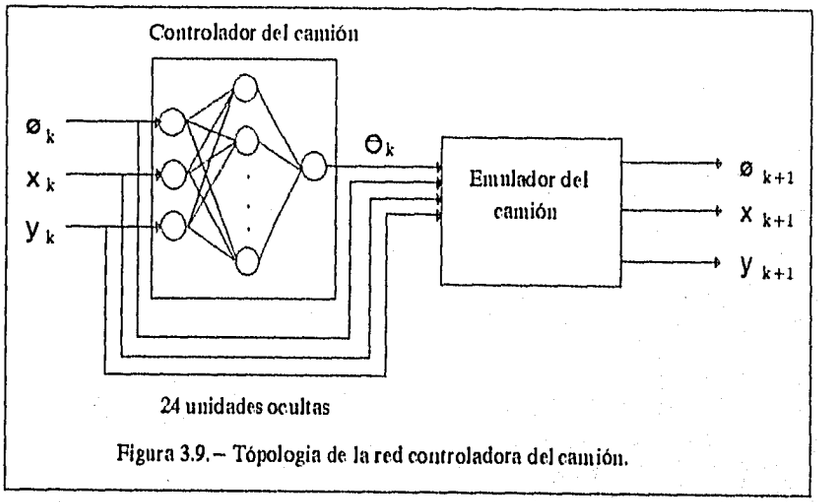


Figura 3.9.– Tópologia de la red controladora del camión.

La figura 3.10 muestra un ejemplo típico de las trayectorias del controlador neuronal del camión de varias posiciones iniciales. Aunque se enseño a la red neuronal a seguir la senda formada por el arco, algunas trayectorias del camión no eran óptimas.

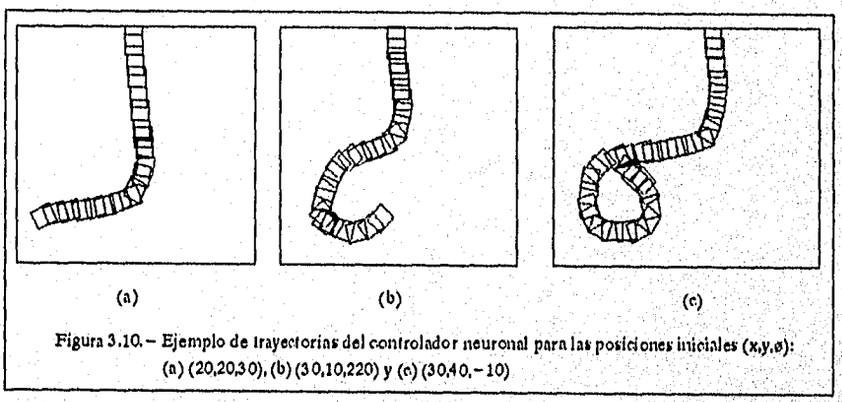


Figura 3.10.– Ejemplo de trayectorias del controlador neuronal para las posiciones iniciales (x,y,ø): (a) (20,20,30), (b) (30,10,220) y (c) (30,40,–10)

3.9.4.- COMPARACION DE LOS SISTEMAS FUZZY Y NEURONAL.

Como muestran las figuras 3.8 y 3.10, el controlador fuzzy siempre sin novedad funciono para el camión, pero el controlador neuronal no lo hizo. El controlador neuronal del camión a veces siguió una senda irregular. El sistema neuronal consumio mas tiempo. El algoritmo backpropagation requirió miles de iteraciones en la red del controlador, en unos casos, el algoritmo no convergió. Se "adiestro" el controlador fuzzy por la codificación de nuestro propio sentido común, reglas **FAM**. Una vez que se desarrollo el banco de reglas **FAM**, se puede calcular la salida del controlador de la resultante matriz del banco **FAM** o superficie de control.

El controlador fuzzy no necesitó un camión emulador y no se requirió hacer un modelo matemático de cómo las salidas dependen de las entradas. El controlador fuzzy realizo cálculos más sencillos que el controlador neuronal. La mayoría de operaciones de cálculo en el controlador neuronal involucraron la multiplicación, adición, o logaritmo de dos números reales. En el controlador fuzzy, la mayoría de operaciones de cálculos involucraron el comparar y agregar dos números reales.

3.9.5.- ANALISIS DE SENSIBILIDAD.

Se estudia la sensibilidad del controlador fuzzy en dos formas. Se reemplazan las reglas **FAM** con destructivas o "erróneas" reglas **FAM** y aleatoriamente se quitaron reglas **FAM**. Deliberadamente se eligen reglas **FAM** erróneas para confundir al sistema. La figura 3.11 muestra las trayectorias cuando dos reglas **FAM** erróneas reemplazaron la importante regla **FAM** del estado estable, la regla 18: el controlador fuzzy tiene que producir una salida de cero cuando el camión esta casi en la posición correcta del estacionamiento.

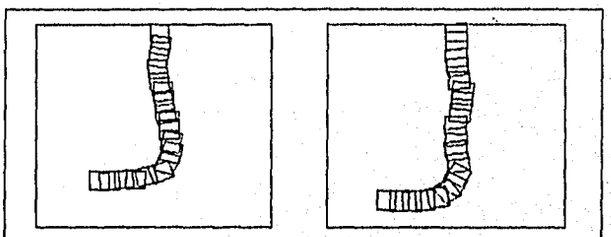


Figura 3.11.– Trayectorias despues de remplazar la regla FAM 18 del estado estable por las 2 reglas erroneas:
(a) IF x = CE AND ø=VE, THEN O=PB;
(b) IF x=CE AND ø=VE, THEN O=NB.

La figura 3.12 muestra las trayectorias del camión después de quitar cuatro reglas **FAM** elegidas aleatoriamente (7, 13, 18, y 23).

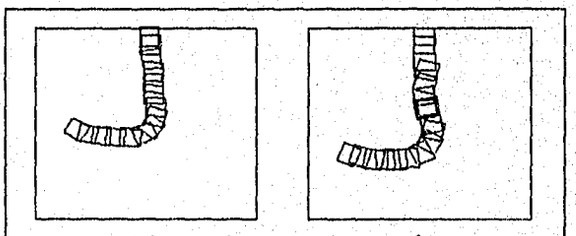


Figura 3.12.– Trayectorias Fuzzy cuando:
(a) no son removidas reglas FAM
(b) son removidas las reglas 7,13 y 18.

Estas perturbaciones no afectaron significativamente la ejecución del controlador fuzzy. Se estudia vigorosamente cada controlador para examinar el valor del fracaso. Para el controlador fuzzy se elimino un porcentaje aleatorio de selectas reglas del sistema **FAM**. Para el controlador neuronal se quitaron datos. La figura 3.13 muestra el promedio del calculo de los errores de ejecución sobre diez típicas reglas **FAM** perdidas para el controlador fuzzy y datos perdidos para el controlador neuronal. Las reglas **FAM** perdidas y los datos están en el rango de 0 a 100 por ciento del total.

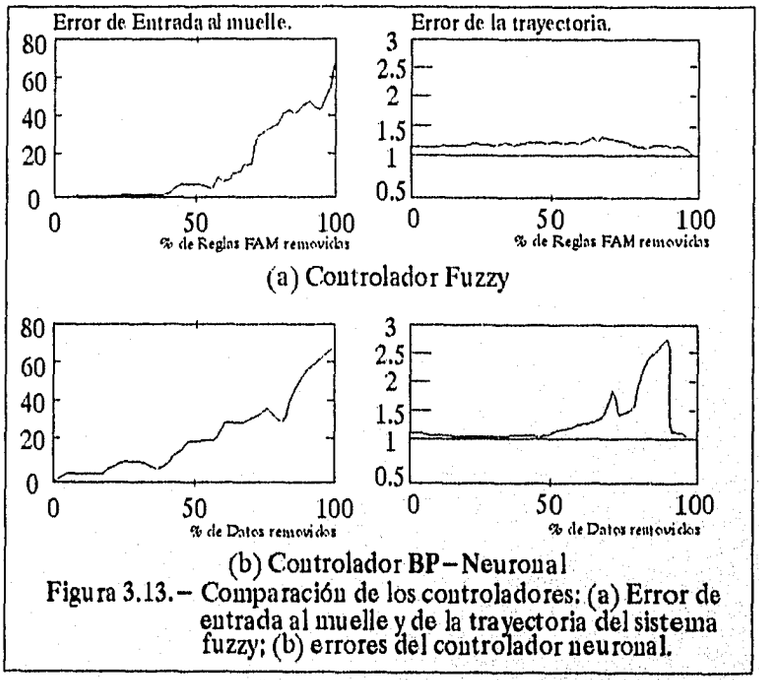


Figura 3.13.– Comparación de los controladores: (a) Error de entrada al muelle y de la trayectoria del sistema fuzzy; (b) errores del controlador neuronal.

En la figura 3.13(a), el error de entrada al muelle es igual a la distancia Euclidiana de la posición final actual (ϕ, x,y), a la posición final deseada (ϕ_f, x_f, y_f):

$$\text{error de entrada al muelle} = \sqrt{(\phi_f-\phi)^2+(x_f-x)^2+(y_f-y)^2} \quad (3\text{-}27)$$

En la figura 3.13(b), el error de la trayectoria es igual a la razón de la longitud de la trayectoria del camión dividido por la distancia de la linea recta al muelle de carga:

$$\text{error de la trayectoria} = \frac{\text{Longitud de la trayectoria}}{\text{distancia(posición inicial, posición final deseada)}} \quad (3\text{-}28)$$

3.9.6.- FUZZY ADAPTABLE DEL CAMION.

Los sistemas **FAM** adaptables (**AFAM**) generan reglas directamente de datos de instrucciones. **A** es un sistema **FAM** de una dimensión, $S: I^n \rightarrow I^p$, que define una regla **FAM**, una sola asociación de la forma **(A,B)**. En este caso el espacio del producto entrada-salida es igual a I^n x I^p, una regla (A_i,B_i) que define un grupo o esfera de puntos en el producto del espacio cubico I^n **x** I^p concentrando los puntos (A_i,B_i). El algoritmo de adaptabilidad de agrupaciones puede estimar la desconocida regla (A_i,B_i) de las muestras formadas en R^2, se utiliza la diferencial del aprendizaje competido (**DCL**) para recobrar el banco de reglas **FAM** que generan los datos de aprendizaje del camión. Se generaron 2230 pruebas del camión de siete diferentes posiciones iniciales y varios ángulos. Se eligieron las posiciones iniciales (20,20), (30,20), (45,20), (50,20), (55,20), (70,20), y (80,20). Se cambio el ángulo de -60° a 240° a cada posición inicial. A cada paso, el controlador fuzzy produjo el ángulo θ de salida de la dirección del camión. El vector de instrucciones (x, ϕ, θ) definió puntos en un espacio producido de tres dimensiones, **x** tiene cinco valores del conjunto fuzzy: **LE**, **LC**, **CE**, **RC**, y **RI**, ϕ tiene siete valores del conjunto fuzzy: **RB**, **RU**, **RV**, **VE**, **LV**, **LU** y **LB**, θ tiene siete valores del conjunto fuzzy: **NB**, **NM**, **NS**, **ZE**, **PS**, **PM** y **PB**. Por lo tanto se tienen 245 (5 x 7 x 7) posibles celdas **FAM**. Se definieron las celdas **FAM** por las particiones del real espacio producido como sigue. Se dividio el espacio $0 \leq x \leq 100$ en cinco intervalos no uniformes [0,32.5], [32.5,47.5], [47.5,52.5], [52.5,67.5] y [67.5,100]. Cada intervalo representa los cinco valores del conjunto fuzzy **LE**, **LC**, **CE**, **RC** y **RI**. Esta selección correspondió a los intervalos no traslapados de la función fuzzy, las gráficas **m(x)** en la figura 3.5. Semejantemente, se dividio el espacio $-90^\circ \leq \phi \leq 270^\circ$ en siete intervalos no uniformes $[-90^\circ,0^\circ]$, $[0^\circ,66.5^\circ]$, $[66.5^\circ,86^\circ]$, $[86^\circ,94^\circ]$, $[94^\circ,113.5^\circ]$, $[113.5^\circ,182.5^\circ]$ y $[82.5^\circ,270^\circ]$, que corresponden respectivamente a **RB**, **RU**, **RV**, **VE**, **LV**, **LU** y **LB**. Se dividio el espacio $-30^\circ \leq \theta \leq 30^\circ$ en siete intervalos no uniformes $[-30^\circ,-20^\circ]$, $[-20^\circ,-7.5^\circ]$, $[-7.5^\circ,-2.5^\circ]$, $[-2.5^\circ,2.5^\circ]$, $[2.5^\circ,7.5^\circ]$, $[7.5^\circ,20^\circ]$ y $[20^\circ,30^\circ]$, que corresponden a **NB**, **NM**, **NS**, **ZE**, **PS**, **PM**, y **PB** las celdas **FAM** cerca del centro eran más pequeño que las celdas exteriores porque se eligió restringir la función cerca de la celda del estado estable. Particiones uniformes del espacio produce cálculos pobres de las muestras originales. Como en la figura 3.5, esto refleja la necesidad de definir juiciosamente los valores del conjunto fuzzy de las variables del sistema. Si una celda **FAM** contiene por lo menos uno de los 245 vectores sinapticos, se introdujo la correspondiente regla **FAM** en la matriz. En el caso de vincular la celda **FAM** elegida con la mayor densidad del grupo de datos. La figura 3.14(a) muestra la distribución de las entradas de prueba (x,ϕ), no se incluye la variable θ

en la figura. El grupo de datos cerca de la posición del estado estable (x = 50 y ϕ = 90°).

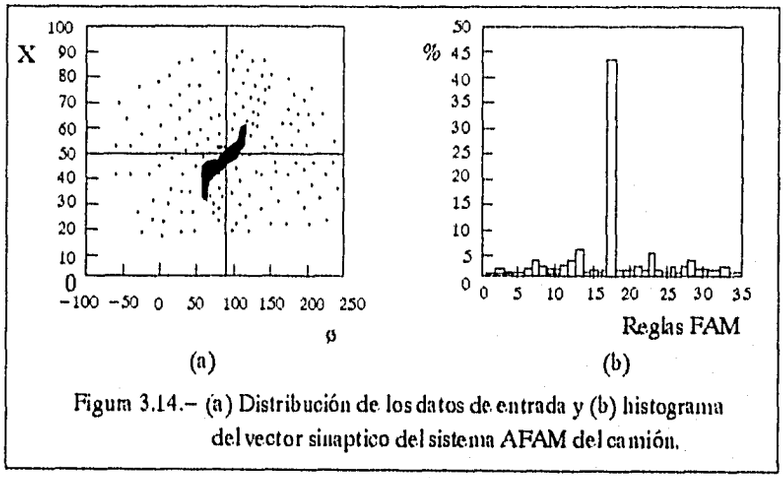


Figura 3.14.– (a) Distribución de los datos de entrada y (b) histograma del vector sinaptico del sistema AFAM del camión.

La figura 3.14(b) muestra el histograma del vector sinaptico **DCL** después de clasificar 2230 vectores de instrucciones para 35 reglas **FAM**. Ya que exitosamente el sistema genero la muestra de la instrucción, en la mayoría de pruebas y así la mayoría de vectores sinapticos, se agruparon en la celda **FAM** del estado estable. el agrupamiento **DCL** del espacio producido estimo 35 nuevas reglas **FAM**. La figura 3.15 muestra la estimación **DCL** del banco **FAM**.

Ø \ X	LE	LC	CE	RC	RI
RB	PS	PM	PM	PB	PB
RU	ZE	PM	PM	PB	PB
RV	NS	ZE	PS	PM	PB
VE	NM	NM	ZE	PM	PB
LV	NB	NB	NS	PS	PS
LU	NB	NB	NB	NS	PS
LB	NB	NB	NM	NM	NS

Figura 3.15.– Banco FAM de la estimación DCL.

Los dos sistemas producen una conducta casi equivalente del camión. Esto sugiere que el adaptable espacio-producto puede agrupar las estimadas reglas **FAM** fundamentales en muchos casos de la conducta (experta), aún cuando la construcción experta o fuzzy no puede articular las reglas **FAM**. También se utiliza la superficie neuronal de control para estimar las reglas **FAM**. Se dividio el rectángulo **[0,100] x [-90,270]** en 35 cuadros no uniformes con las mismas divisiones como en el caso del controlador fuzzy. Entonces se agregan y calculan

los valores de la superficie de control en el cuadro. Se agrega una regla **FAM** al banco, si el valor medio corresponde a una de las siete celdas. La figura 3.16 muestra el resultante banco **FAM**.

Ø \ X	LE	LC	CE	RC	RI
RB	PS	PB	PB	PB	PB
RU	NM	ZE	PM	PB	PB
RV	NM	NM	NS	PS	PB
VE	NM	NM	NM	ZE	PB
LV	NM	NM	NM	NS	PB
LU	NM	NM	NM	NM	PM
LB	NM	NM	NM	NM	NM

Figura 3.16.– Banco FAM generado por el controlador neuronal

Como se esperó, el sistema **DCL-AFAM** produjo el menor error absoluto que el producido por el sistema **BP-AFAM**. La figura 3.17 muestra los errores de entrada al muelle y de las trayectorias de los dos sistemas **AFAM** de control.

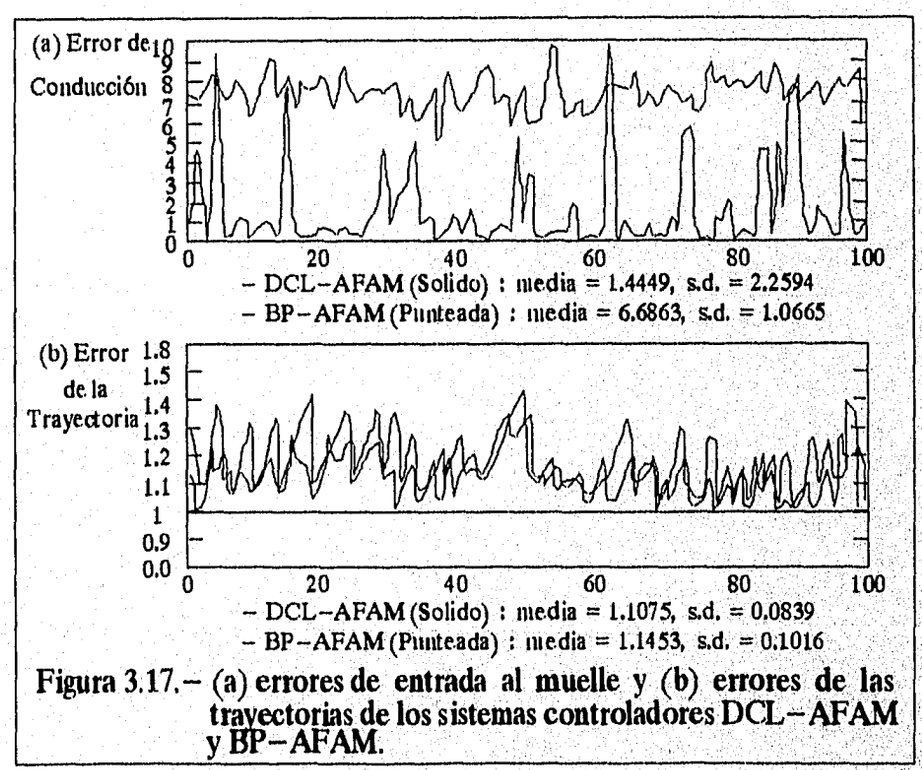


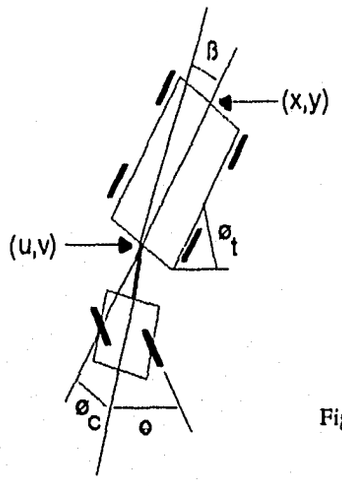
Figura 3.17.– (a) errores de entrada al muelle y (b) errores de las trayectorias de los sistemas controladores DCL–AFAM y BP–AFAM.

El sistema **DCL-AFAM** produjo un menor error de entrada al muelle que el sistema **BP-AFAM** produjo para 100 arbitrarias pruebas. Los dos sistemas **AFAM** generaron semejantes trayectorias. Esto sugiere que la estimación neuronal puede definir la apariencia final de la estructura del sistema **FAM**. En principio se puede utilizar esta técnica para generar la estructura de las reglas **FAM** por alguna aplicación neuronal. Podemos entonces inspeccionar y refinar estas reglas y quizás reemplazar el original sistema neuronal con el adaptado sistema **FAM**.

3.9.7.- CONTROLADOR FUZZY DEL CAMION Y EL REMOLQUE.

Agregamos un remolque al sistema del camión. La figura 3.18 muestra el sistema de simulación del camión y el remolque. Se agrego una variable más (el ángulo del coche, ϕ_c) la tercera variable declarada del remolque del camión. En este caso la regla **FAM** lleva la forma:

SI x = LE y ϕ_t = RB y ϕ_c = PO, entonces ß = NS



(x,y): Coordenadas cartesianas de la parte posterior, [0,100]
(u,v): Coordenadasde union.
($ø_t$): Angulo del remolque con la horizontal
($ø_c$): Angulo relativo del coche con el remolque, [-90,90]
(O): Angulo de la dirección, [-30,30]
(ß): Angulo del remolque a actualizar a cada movimiento, [-30,30]

Figura 3.18.– Diagrama para el sistema camión y remolque.

Las cuatro variables declaradas x, y, ϕ_t y ϕ_c determinaron la posición del camión y del remolque en el plano. La variable fuzzy ϕ_t correspondió a ϕ para el remolque del camión. La variable fuzzy ϕ_c, especifica el ángulo relativo del camión con respecto a la linea del centro del remolque. ϕ_c esta en el rango de -90° a 90°. Los ángulos extremos del coche 90° y -90° corresponden a las dos posiciones de "coleo" del coche con respecto a el remolque. El valor positivo de ϕ_c indicó que el coche permanece en el lado izquierdo del remolque. El valor negativo indica que permanece en el lado derecho del remolque. La figura 3.18 muestra un valor positivo del ángulo

ϕ_c. Las variables fuzzy x, ϕ_t, y ϕ_c se definieron como las variables de entrada. La variable fuzzy ß se definió como la salida. La variable ß indica el ángulo que necesitamos actualizar del remolque a cada iteración. Calculamos el ángulo θ de salida de la dirección con la siguiente relación geométrica. Con el valor calculado ß, la posición del remolque (x, y) se movió a la nueva posición (x', y'):

$$x' = x + r\cos(\phi_t + ß) \quad (3\text{-}29)$$
$$y' = y + r\sin(\phi_t + ß) \quad (3\text{-}30)$$

donde **r** denota una distancia fija de apoyo, entonces la unión del coche y el remolque (**u**, **v**) se movió a la nueva posición (**u'**, **v'**):

$$u' = x' - l\cos(\phi_t + ß) \quad (3\text{-}31)$$
$$v' = y' - l\sin(\phi_t + ß) \quad (3\text{-}32)$$

donde **l** denota lo largo del remolque. Actualizamos el vector de la dirección (**dirU**, **dirV**), que define el ángulo del coche, por:

$$dirU' = dirU + \Delta u \quad (3\text{-}33)$$
$$dirV' = dirV + \Delta v \quad (3\text{-}34)$$

donde **$\Delta u = u' - u$ y $\Delta v = v' - v$.** El nuevo vector de la dirección (**dirU'**, **dirV'**) define el nuevo ángulo ϕ'_c del coche. Entonces obtenemos el valor del ángulo de la dirección como $\theta = \phi'_{c,h} - \phi_{c,h}$, donde $\phi_{c,h}$ denota el ángulo del coche con la horizontal. Se eligieron los mismos valores del conjunto fuzzy y la función ß como se eligieron para θ, el rango de ß es de -30° a 30°. Se eligieron los valores del conjunto fuzzy para ϕ_c como **NE**, **ZR** y **PO** como en la figura 3-19.

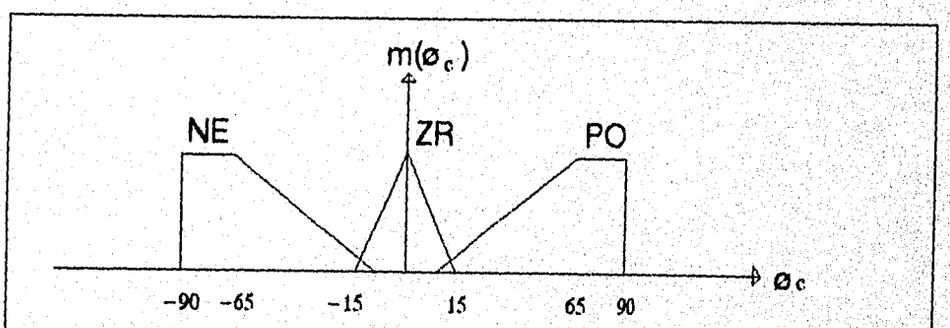
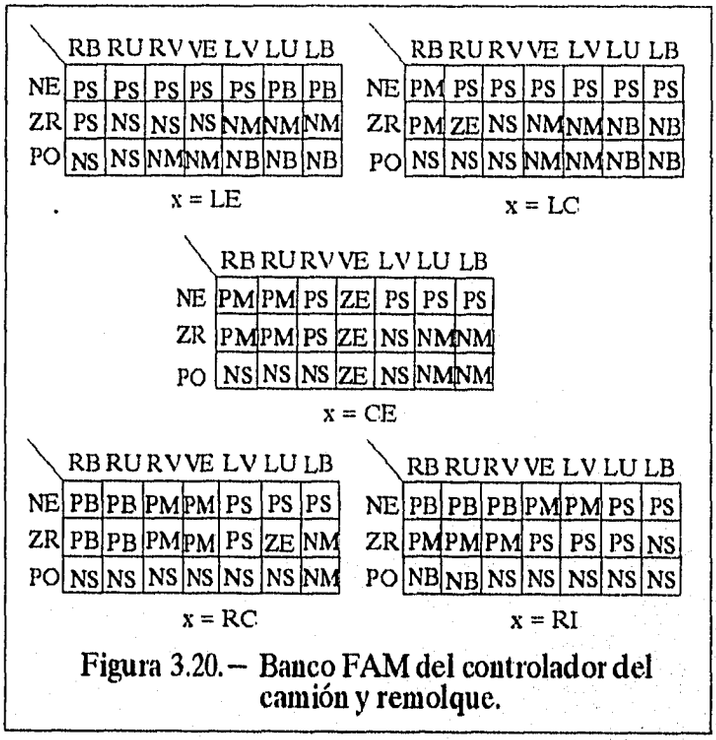


Figura 3.19.– Grafica de los tres valores del conjunto fuzzy para la variable ø c.

La figura 3.20 muestra las cinco matrices de reglas del banco **FAM** del sistema fuzzy camión-remolque. En la figura 3.20 se arreglaron las variables fuzzy **x** como **LE**, **LC**, **CE**, **RC** y **RI**. Había 735 (7 x 5 x 7 x 3) posibles reglas **FAM** y solamente 105 reglas **FAM** reales.



	RB	RU	RV	VE	LV	LU	LB
NE	PS	PS	PS	PS	PS	PB	PB
ZR	PS	NS	NS	NS	NM	NM	NM
PO	NS	NS	NM	NM	NB	NB	NB

x = LE

	RB	RU	RV	VE	LV	LU	LB
NE	PM	PS	PS	PS	PS	PS	PS
ZR	PM	ZE	NS	NM	NM	NB	NB
PO	NS	NS	NS	NM	NM	NB	NB

x = LC

	RB	RU	RV	VE	LV	LU	LB
NE	PM	PM	PS	ZE	PS	PS	PS
ZR	PM	PM	PS	ZE	NS	NM	NM
PO	NS	NS	NS	ZE	NS	NM	NM

x = CE

	RB	RU	RV	VE	LV	LU	LB
NE	PB	PB	PM	PM	PS	PS	PS
ZR	PB	PB	PM	PM	PS	ZE	NM
PO	NS	NS	NS	NS	NS	NS	NM

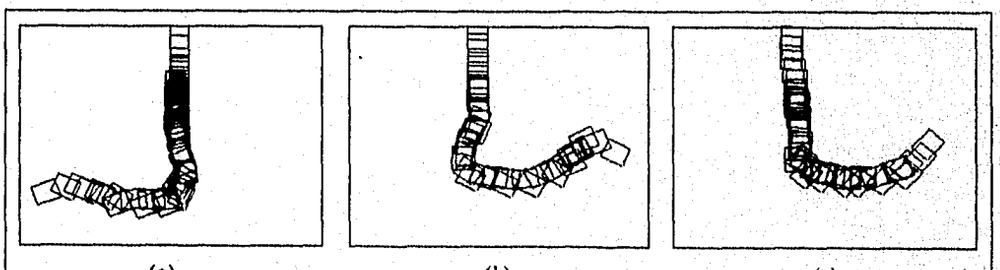
x = RC

	RB	RU	RV	VE	LV	LU	LB
NE	PB	PB	PB	PM	PM	PS	PS
ZR	PM	PM	PM	PS	PS	PS	NS
PO	NB	NB	NS	NS	NS	NS	NS

x = RI

Figura 3.20.– Banco FAM del controlador del camión y remolque.

La figura 3.21 muestra las típicas trayectorias del controlador fuzzy del sistema camión-remolque de diferentes posiciones iniciales. Las diferentes direcciones del camión y remolque, dependen de la posición relativa del camión con respecto al remolque. El sistema de control fuzzy exitosamente controlo el camión y el remolque en las posiciones de coleo.



(a) (b) (c)

Figura 3.21.– Ejemplo de trayectorias del controlador Fuzzy camión–remolque para las posiciones iniciales (x,y,ϕ_t,ϕ_c): (a) (25,30,–20,30), (b) (80,30,210,–40) y (c) (70,30,200,30).

3.9.8.- SISTEMA CONTROLADOR BP DEL CAMION Y REMOLQUE.

Se agrego la variable ϕ_C del ángulo del camión al controlador neuronal **backpropagation** como una entrada. La red del controlador contiene 24 neuronas ocultas con la variable de salida ß. La muestra de instrucciones se compone de un espacio de cinco dimensiones de la forma (x, y, ϕ_t, ϕ_C, ß) se formo la red del controlador con 52 muestras de instrucciones del controlador fuzzy: 26 muestras para la mitad izquierda del plano, 26 muestras para la mitad derecha del plano. Se utilizaron las ecuaciones (3-29)-(3-34) en vez de la red emuladora. Se requirió realizar más de 200,000 iteraciones. Unas series de instrucciones no convergieron. El controlador **BP** formado opero bien excepto en algunos casos, la figura 3.22 muestra una típica trayectoria del sistema controlador **BP** del camión-remolque de las mismas posiciones iniciales utilizadas en la figura 3-21.

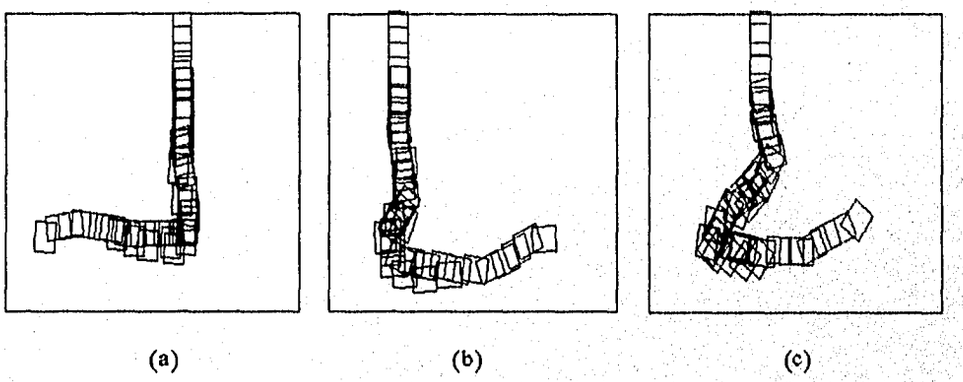


Figura 3.22.– Ejemplo de trayectorias del camión y remolque del sistema BP de la posición inicial (x,y,ϕ_t,ϕ_c): (a) (25,30,–20,30), (b) (80,30,210,–40) y (c) (70,30,200,30).

Se realizaron las mismas pruebas para el controlador fuzzy y el **BP** del camión-remolque como en el caso del camión. La figura 3.23 muestra el promedio de errores de ejecución calculados sobre diez típicos camiones de diez diferentes posiciones iniciales. Estas gráficas de ejecución se parecen a las gráficas de ejecución para los sistemas del camión de la figura 3.13.

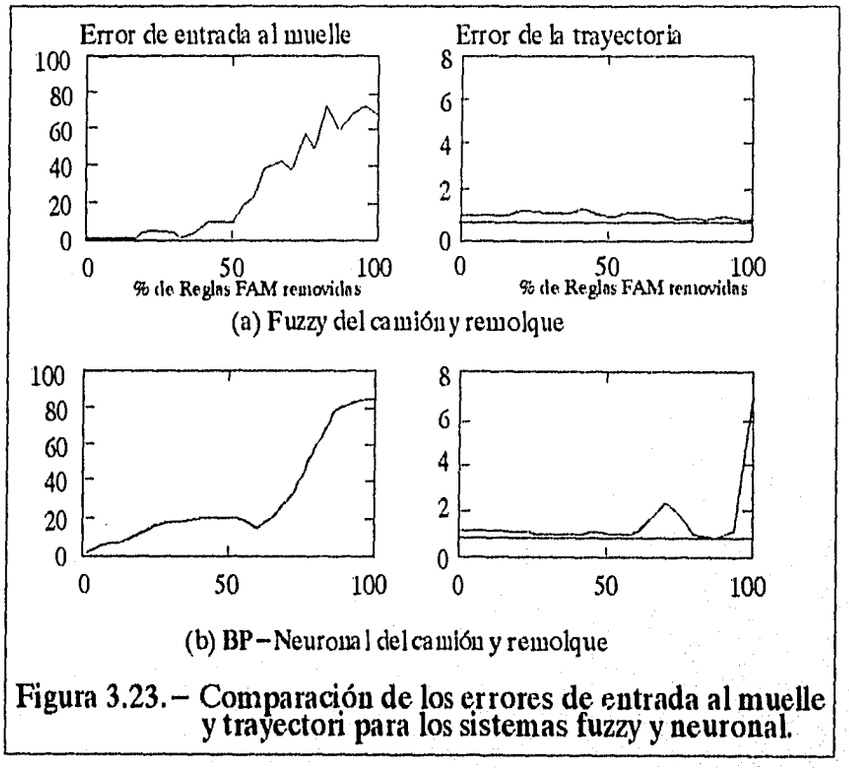


Figura 3.23.– Comparación de los errores de entrada al muelle y trayectori para los sistemas fuzzy y neuronal.

3.9.9.- SISTEMA CONTROLADOR AFAM DEL CAMION Y REMOLQUE.

Se generaron 6250 datos del camión y remolque utilizando el sistema **FAM** original de la figura 3.20. Se retrocedio el camión y el remolque de las mismas posiciones iniciales. El ángulo del remolque ϕ_t esta en el rango de -60° a 240° y para el ángulo del camión ϕ_c se supuso solamente tres valores -45°, 0° y 45°. El vector de instrucciones (x, ϕ_t, ϕ_c, β) definió puntos en el producto del espacio de cuatro dimensiones de la entrada-salida. No se particiono uniformemente el espacio de las celdas **FAM** permitiendo más estrechos valores del conjunto fuzzy cerca de la celda del estado estable. Se utiliza el **DCL** para el entrenamiento del controlador **AFAM** del camión-remolque. El número total de celdas **FAM** es igual a 733 (7 x 5 x 7 x 3). Se utilizaron 735 vectores sinápticos. El algoritmo **DCL** clasificó los 6250 datos en las 105 celdas **FAM**. Se trataron las variables fuzzy del camión-remolque como antes, para el ángulo ϕ_c, se dividio el espacio $-90° \leq \phi_c \leq 90°$ en tres intervalos [-90°,-12.5°], [-12.5°,12.5°] y [12.5°,90°], que corresponden a **NE**, **ZR**, y **PO**. Había 735 celdas **FAM** y 735 posibles reglas **FAM**, de la forma (x, ϕ_t, ϕ_c, β). La figura 3.24 muestra el histograma del vector sináptico correspondiente a las 105 reglas **FAM**. La figura 3.25 muestra el estimado banco **FAM** por el algoritmo **DCL**. La figura 3.26 muestra las trayectorias del original **FAM** y la estimación **DCL** del controlador **AFAM** del camión-remolque. Las figuras 3.26(a) y (b) muestran las dos trayectorias de la

posición inicial (x, y, ϕ_t, ϕ_c) = (30, 30, 10°, 45°). Las figuras 3.26(c) y (d) muestran las trayectorias de la posición inicial (60, 30, 210°, -60°). El original **FAM** y la estimación **DCL** del sistema **AFAM** exhibieron una ejecución comparable del controlador excepto en algunos casos, donde la estimación **DCL** de las trayectorias del **AFAM** eran irregulares.

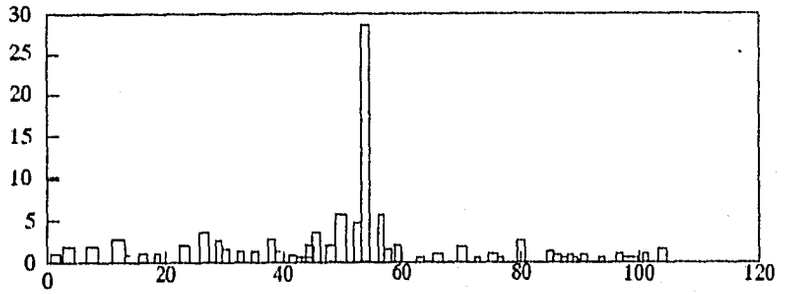


Figura 3.24.– Histograma del vector sinaptico para el sistema AFAM del camión y remolque

$\emptyset_t$

$\emptyset_c$	RB	RU	RV	VE	LV	LU	LB
NE	PS	PS	PS	PS	PB	PB	PB
ZR	PS	ZE	NS	NS	NM	NB	NB
PO	ZE	NS	NM	NM	NB	NB	NB

x = LE

	RB	RU	RV	VE	LV	LU	LB
NE	PM	PS	PS	PS	PB	PS	PS
ZR	PM	PS	ZE	NM	NS	NM	NM
PO	NS	NS	NS	NM	NB	NB	NB

x = LC

	RB	RU	RV	VE	LV	LU	LB
NE	PM	PM	PM	PS	PS	PS	PS
ZR	PM	PM	PS	ZE	NS	NM	NM
PO	NS	NS	NS	NM	NM	NB	NM

x = CE

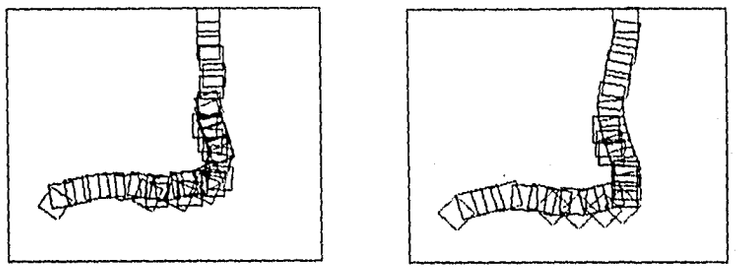
	RB	RU	RV	VE	LV	LU	LB
NE	PB	PB	PM	PM	PS	PS	PS
ZR	PB	PB	PB	PS	PS	ZE	NM
PO	NB	NB	NB	NS	NS	NS	NM

x = RC

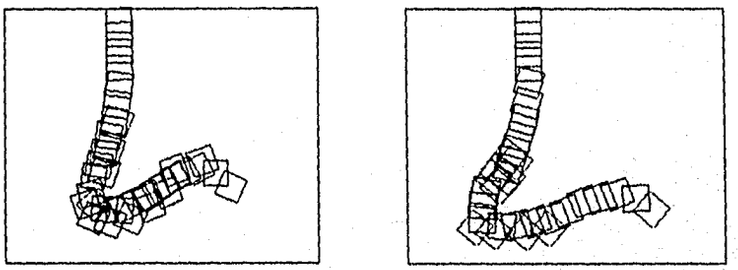
	RB	RU	RV	VE	LV	LU	LB
NE	PB	PB	PB	PM	PS	PS	ZE
ZR	PB	PB	PM	PS	PS	PS	ZE
PO	NB	NB	NB	NS	NS	NS	NS

x = RI

Figura 3.25.– Banco FAM de la estimación DCL para el sistema AFAM del camión y remolque.



(a) FAM Original (b) Estimación DCL del AFAM



(c) FAM Original (d) Estimación DCL del AFAM

Figara 3.26.– Ejemplo de la trayectori del camión y remolque del sistema original y la estimación DCL para la posición inicial $(x,y,\phi,\phi) = (30,30,10,45)$ y $(60,30,210,-60)$.

C O N C L U S I O N

En el análisis realizado con la aplicación de sistema Fuzzy, podemos apreciar el gran avance que se ha tenido en cuanto al desarrollo de redes, y aunque en nuestro estudio podemos apreciar que se tiene una mayor precisión al utilizar sistemas Fuzzy, que al utilizar Redes Neuronales, no debemos descartar la posibilidad de que en un futuro no lejano puedan acoplarse mejor los sistemas Fuzzy y las Redes Neuronales para poder conseguir mayores avances y en forma mas rápida.

Los avances científicos y tecnológicos que se dan día con día, nos permiten en este momento determinar que en un tiempo no muy lejano, se podra conseguir, no solo la aplicación cada día masa real y practica de las Redes Neuronales Artificiales junto con los sistemas Fuzzy Logic debido a su gran precisión en la salida por su relación entre memorias asociativas (FAM).

Y lo que es as, gracias a las investigaciones y estudios hechos por científicos de la Gran Bretaña, según un futurólogo en el año 2020 podra ser posible realizar la conexión de un cerebro humano con una computadora ya que en la actualidad se esta estudiando cada vez mas a detalle el funcionamiento biológico de las neuronas para, mas adelante lograr tener un funcionamiento microelectrónico muy similar y así poder pensar en una posible conexión cerebro-maquina.

Esperamos que el estudio en este campo pronto se de en nuestro país para poder adquirir un conocimiento mas amplio de dichos sistemas y también poder generar nuestros propios sistemas de redes de vanguardia.

Apéndice A

Neuronas Biológicas

INTRODUCCION.

Hace tres siglos Rene Descartes describio la mente como una entidad extracorporal, esto fue expresado por la glándula **"Pineal"**. Descartes se equivocó sobre la **Pineal**, pero estimuló un fuerte debate en cuanto a la relación entre mente y cerebro. Cómo la mente no material influye en el cerebro, y viceversa?. Al realizar estudios sobre esta relación, Descartes tuvo un error, no se dio cuenta que el cerebro humano es la estructura más compleja en el universo, compleja y bastante coordinada como los dedos de un pianista o como crear un paisaje tridimensional tan ligero que para la vista parece uno de dos-dimensiones. No supo como se construye la maquinaria del cerebro y mantuvo la conjetura que era por genes y por experiencias, y él sin duda no supo que la versión corriente es el resultado de millones de años de evolución. Es difícil entender el cerebro y porque no se puede construir una computadora similar a él, con propósitos específicos o principios de diseño de la mente. La selección natural, es la maquinaria de la evolución, que es la responsable de la constitución del cerebro. Si Descartes hubiera sabido estas cosas, él se hubiera preguntado al igual que los modernos neurobiologistas, si el cerebro es bastante complejo de explicar como el misterio de la imaginación humana, de la memoria y el humor. Preguntas filosóficas que tienen que suplementarse por experimentos que ahora entre la mayoría es un reto emocionante para la ciencia. Nuestra supervivencia y probablemente la supervivencia de este planeta depende en gran parte de entender por completo la mente humana. Si convenimos pensar que la mente es como una colección de bastantes procesos mentales como una substancia o espíritu, es más fácil quedarse con los anteriores estudios empíricos. Lo urgente es buscar las bases neuronales del fenómeno mental. Estos acontecimientos mentales pueden ser correlacionados con modelos de impulsos nerviosos en el cerebro. Para apreciar completamente el sentido de esta suposición, se tiene que considerar cómo trabaja una celda neuronal; cómo se comunica con otras?, cómo se organizan en forma local? o se distribuyen en redes?, y cómo la conexión entre neuronas cambia con la experiencia?. Es también

importante definir claramente los fenómenos mentales pues es necesario que se puedan explicar. Avances notables han sido realizados a cada uno de estos niveles. Intrigantes correlaciones se tienen en realidad y empiezan a salir entre los atributos mentales y los modelos de impulsos nerviosos que brillan y se destiñen en tiempo y espacio, en alguna parte interior del cerebro.

A.1.- MENTE Y CEREBRO.

Para la mayoría de los neurobiologos el cerebro humano requiere de una gran atención especial, pues es enorme, aparentemente el hemisferio central es simétrico y el corazón esta colocado a horcajadas en la parte central que se extiende abajo de la medula ósea: los hemisferios arrugados son cubiertos por una celda, la corteza es una lámina de dos milímetros de espesor. La corteza cerebral puede ser subdividida por un criterio morfológico y funcional en numerosas áreas sensoriales, en áreas motocontroladoras y áreas bien definidas en donde los acontecimientos asociativos tienen lugar, muchos observadores suponen que es aquí en la interface entre la entrada y salida donde ocurre la gran síntesis de la vida mental.

El cerebro humano pesa solamente de 900 a 1200 gramos pero contiene como 100 billones de neuronas. Aunque ese número es extraordinario está del mismo orden de magnitud como el número de estrellas en la Vía Láctea, no se puede explicar la complejidad del cerebro. El hígado probablemente contiene 100 millones de celdas, pero 1,000 hígados no suman una rica vida interior.

Santiago Ramón y Cajal, el padre de la ciencia moderna del cerebro, descubrió como las celdas de neuronas polarizadas reciben señales en grandes extensiones atraves de las ramificación de sus cuerpos, que llamó dendritas, y manda la información por unas extensiones de las divisiones, que llamó axóns. El método Golgi revelo una gran variedad de formas en el cuerpo de las celdas, ramificaciones dendriticas y axón largos. Cajal discernió una distinción fundamental entre celdas al tener axóns corto, que se comunican con otras celdas que tienen axóns largo y estos se proyectan a otras regiones.

Estas formas no son la única fuente de variación entre neuronas. Estas diversidades son aún más grandes si se consideran las diferencias moleculares. Toda celda contiene la misma colección de genes, individualmente las celdas activan solamente un subconjunto pequeño. En el cerebro, la expresión de selección de genes ha sido hallada aparentemente dentro de poblaciones homogéneas como el amacrine en la retina, las Purkinje en la medula ósea. Más allá estructuralmente existen diferencias moleculares, aún pueden

existir distinciones mas refinadas que entre neuronas pueden ser marcadas si se toman en cuenta sus entradas y proyecciones.

En la presentación de esta diversidad asombrosa, es un alivio aprender que esta simplificación puede ser marcada. Otra simplificación es que todas las neuronas conducen información en muchas rutas iguales. La información viaja por axóns en la forma de breves impulsos eléctricos llamados acciones potenciales, la acción potencial, es aproximadamente de 100 milivoltios en amplitud y un milisegundo en duración, resultado del movimiento de un ion de sodio cargado positivamente a través de la superficie de la membrana, un fluido extracelular en el interior de la celda, o citoplasma.

Aunque los axóns se ven como alambres aislados, no conducen impulsos en el mismo camino: no son buenos cables; la resistencia por el eje es muy alta y la resistencia de la membrana es muy baja. La carga positiva que entra en el axón durante la acción potencial es disipada en uno o dos milímetros. Al viajar distancias de muchos centímetros, la acción potencial tiene que ser frecuentemente regenerada por el camino. La necesidad de levantar repetidamente los límites de corriente se debe a la velocidad máxima a que un impulso viaja a 100 metros por segundo. Esto es menos que una millonésima de la velocidad a que una señal eléctrica se mueve en un cable telegráfico. Así, la acción potencial es con frecuencia relativamente baja. Pensamientos fugaces tienen que depender de la aparente sincronización que conduce al impulso sobre muchos axóns en paralelo y a las miles de conexiones hechas por cada uno.

El estudio del cerebro no es sencillo. En la mayoría de los casos, su comunicación entre neuronas es mediada por transmisiones químicas, estas son liberadas a través de contactos especiales llamados sinapsis. Cuando las acciones potenciales llega a la terminal del axón, son transmitidas liberando pequeños paquetes de 20 nanómetros de anchura que separan las membranas presinapticas y postsinápticas. Los efectos son excitantes cuando el movimiento de carga trae a la membrana más cercana al umbral por la acción potencial generada. Es inhibitorio cuando se estabiliza la membrana cerca de su valor de descanso. Cada sinapsis produce solamente un efecto pequeño. Colocando la intensidad (frecuencia-acción-potencial) de su salida, a cada neurona se tienen que integrar continuamente mas de 1,000 entradas sinapticas, que no se suman de una manera sencilla. Cada neurona es una computadora sofisticada.

Por buenas razones, el cerebro humano es tratado en la mayoría de casos, como el objeto mas complejo en el universo. Este comprende trillones de celdas, 100 billones de estas neuronas están vinculadas en redes que aumentan la inteligencia, habilidad, facultad creadora, emoción,

conciencia y memoria. Anatómicamente grandes, subdivisiones en el cerebro ofrece un mapa áspero de sus capacidades. A un nivel muy grueso, el cerebro es simétrico bilateralmente, sus hemisferios izquierdo y derecho se unen por los cuerpo callosum y otros puentes axónales. Su base consiste de estructuras tales como la medular, que regula las funciones autónomas (incluso respiración, circulación y digestión), y el cerebelo, que tiene a su cargo los movimientos coordinados. Para las mentiras el sistema límbico (azul), una colección de estructuras involucraron el comportamiento sensible de la memoria a largo plazo y otras funciones.

Muchos tipos diferentes de transmisiones han sido identificadas en el cerebro, y esta variedad tiene inferencias enormes para el funcionamiento del cerebro. Desde el primer neurotrasmisor que fue identificado en 1921, la lista de candidatos ha crecido a grandes pasos, el numero es aproximadamente de cincuenta. Hemos aprendido grandes cosas sobre cómo se sintetizan los transmisores, cómo son activados al igual que los receptores en la membrana postsináptica. Este nivel de análisis es particularmente singular para los psiquiatras, por los desordenes neurobilogicos que arrojan los trabajos sobre la mente[1]. La nicotina activa la recepción de la acetilcolina, que es distribuida por todo la corteza cerebral. Una visión más profunda en las bases químicas del pensamiento y conducta depende de la obtención de más datos precisos en cuanto a los sitios de acción de estos potentes agentes y el descubrimiento de más selectores de moléculas, que unen a los receptores. Los transmisores receptores pueden ser agrupados en dos grandes superfamilias de acuerdo a su base de aminoácido y de acuerdo a la forma en que se supone como las moléculas son incrustadas en alguna parte de la membrana de la celda.

El primer gen receptor de dopamina fue aislado en 1988. Se basó en la suposición de que el receptor se parecería a otros receptores que fueron conocidos al unirse proteínas **G**. Esta poderosa "homología" proyecta la estrategia que condujo a la identificación de cuatro receptores más de dopamina. Una de las recientes adiciones, imaginariamente nombrada **D4**, ha atraído considerablemente la atención. El receptor toma dopamina y clozapina con afinidad extraordinariamente alta, de igual importancia, el gen **D4** aparentemente no esta expresado en la ganglia, se encuentra una mayor explicación en la ausencia de la disquinesia. La localización precisa del receptor **D4** dentro de la corteza prefrontal puede revelar el origen de alucinaciones o por lo menos un componente de la maquinaria nerviosa que esta en forma oblicua en la esquizofrenia. La lenta velocidad a que se activan las drogas presenta un enigma. Las interacciones del receptor de la droga son inmediatas, sin embargo los síntomas de

1 Elliot S. Gershon y Ronald O. Rieder, "Mayores Desordenes de Mente y Cerebro", p.p. 88.

esquizofrenia, depresión y otros desordenes no se resuelven en varias semanas. Investigaciones de la sinapsis con dopamina han proporcionado información de los daños y efectos adicionales de la droga. La cocaína, retiene e inhibe una proteína que transporta dopamina lejos de su sitio de acción, es uno de los más poderosos narcóticos que se conocen. Estudios recientes apuntan a una "trayectoria" neuronal que es el mayor blanco de todas las substancias adictivas como las anfetaminas, la nicotina, el alcohol y opio.

Las rutas en el cerebro han sido trazadas por medio de una variedad de moléculas que son transportadas por axóns. Tal reporte de moléculas puede ser visualizado una vez en el tejido fino propiamente preparado. La conexiones también han sido trazadas por microelectrodos posicionados que encierran un cuerpo de la celda del nervio o un axón descubriendo las pequeñas corrientes generadas como una acción potencial. Cada técnica ha revelado un orden, en los mapas topográficos de la corteza cerebral. Se representa la superficie del cuerpo en el espacio cerebral post-central de la corteza cerebral aunque las neuronas corticales forman tres sinapsis lejos de los receptores sensoriales de la piel. Igual que un punto de un mapa del mundo al visualizarlo, es evidente que en la primera corteza visual del hueso occipital este regresa al cerebro. Este orden es evidente a cada una de las rutas relevantes hacia la corteza, y en el orden topográfico también se han encontrado proyecciones de los cortices primarios hacia centros más alto.

El modelo de flujo de información en el cerebro durante la ejecución de tareas mentales no puede fácilmente determinarse por estudios anatómicos del diagrama de circuito o por estudios de plasticidad. La correlación neuronal de más alto funcionamiento mental hasta este momento se a buscado directamente en los primates enseñando los a hacer tareas que requieren juicio, planeación o de memoria, o las tres capacidades. Esta demanda requiere sofisticados instrumentos, diseño experimental y meses de enseñanza hasta que el mono piense los mismos pensamientos como el investigador. Todas las sesiones se invierte en escuchar las amplificaciones de las acciones potenciales, generadas por una o algunas neuronas, seguidas por días de análisis tomando datos como reglas. El progreso es lento, pero importantes generalizaciones han salido. Uno de los principios más importantes es que el sistemas sensorial esta arreglado de una manera jerárquica. Esto es, las neuronas responden progresivamente en los aspectos abstractos de estímulos complejos. El hecho es que las neuronas responden a líneas preferentemente que hacen manchas. Otro principio importante, discutido por Semir Zeki, es que la información no viaja por

un solo camino. Preferentemente, fracciones diferentes de una sola percepción son procesos con caminos paralelos[2].

En suma, podemos esperar avances a una velocidad creciente en todos los niveles de investigación referentes a la mente. Pronto sabremos exactamente cuántos transmisores y receptores están en el cerebro y donde se concentra cada uno. También tendremos un mayor y completo cuadro de acciones de neurotrasmisor, incluso interacciones múltiples conjuntamente relacionadas con moduladores. Y aprenderemos mucho más de las moléculas que afectan la diferenciación y degeneración neuronal. Muchos de los neurotransmisores son comúnmente aminoácidos encontrados por todo el cuerpo. Asimismo, existen principios moleculares específicos del cerebro que han resultado del estudio de regulación de hormonas o de factores que influyen en la supervivencia y diferenciación de neuronas. El desafío es grande para determinar cómo estas moléculas modulan el diagrama funcional telegráfico del cerebro. Ultimamente, será esencial especificar qué exactamente se deben mencionar que estos acontecimientos mentales están correlacionados con señales eléctricas.

A.2.- MICROCIRCUITOS EN EL SISTEMA NERVIOSO.

Se analizan usualmente circuitos de neuronas en términos del axón, la fibra más larga de la neurona. Ahora parece que hay muchos circuitos que involucran neuronas y sus extensiones más cortas, las dendritas.

Por casi un siglo la base de la investigación de la fisiología de la conducta ha sido dominada por el concepto que el sistema nervioso es tranquilo en centros y rutas. Dentro de los centros se agrupan las neuronas, celdas de nervio, que realizan funciones específicas tal como el proceso de la información sensorial y el control de movimientos. Unas de estas celdas largas y delgadas las llamaron axóns, que transmiten información de un centro a otro en la forma de impulsos electroquímicos. La conducta por lo tanto involucra la transmisión continua de impulsos en los axóns en diferentes rutas. Esta vista tradicional naturalmente ha enfocado su atención sobre el impulso del nervio y el axón. Los neurofisiologístas han comprobado cómo el impulso del nervio es generado y los neuroanatomistas han trazado la trayectoria de los axóns, tales investigaciones han producido capítulos brillantes en el desarrollo de la neurociencia.

---

2 Patricia S. Goldman-Rakic, "Working Memory and the Mind", Scientific American, Sep. 1992, p.p. 73-79.

A.2.1.- DENDRITAS Y SINAPSIS.

Dentro de cada centro, en las celdas de neuronas son ascendentes sus dendritas: cortas, ramificandose los hilos que terminan en la celda vecina. Los largos y ramificados padres de las dendritas son muchos y variados. Algunas dendritas son escasas de ramas, algunas tiene ramificaciones en todas direcciones, las dendritas pueden extenderse desde un décimo de milímetro hasta varios milímetros. Los variados modelos de ramificaciones dendriticas tienen características en las cuales cada una de las muchas regiones están especializadas dentro del sistema nervioso central. En 1950 el microscopio electrónico revelo las confluencias internas que conectan las celdas de nervio: las sinapsis. Esta información de confluencia es trasladada de una neurona a otra, usualmente por medio de los mensajeros químicos, se conoció como funcionan los neurotransmisores. Una sinapsis entre un axón y el soma (el cuerpo de la celda) es una neurona axomática; una entre un axón y una dendrita se llama axodendritica. Por muchos años se asumio que éstos eran los únicos tipos posibles de sinapsis y por lo tanto que la neurona era funcionalmente polarizada, recibía señales de sus dendritas y cuerpo de la celda y las transmitia por su axón. Como veremos, el sistema nervioso es mucho más entendible al colocar los circuitos juntos con la sinápsis.

El efecto de una entrada sinaptica es inducir un potencial post-sinaptico: un cambio lento en el voltaje a través de la membrana exterior recibido en la neurona. Dependiendo de los tipos de neurotrasmisores involucrados, el potencial post-sináptico es por cualquiera de dos medios excitación o inhibición. Donde un impulso de nervio viajando debajo de un axón es todo un fenómeno, la excitación e inhibición post-sináptico es graduada por sus potenciales: la amplitud de la respuesta depende de la intensidad de la entrada. En las neuronas que tiene un axón largo, con la suma de los potenciales excitatodores se logra un cierto umbral, la celda se mantiene "encendida" o manda un impulso bajo su axón. Si, en cambio, la entrada excitadora no alcanza el umbral, o si es opuesta por una entrada de inhibición, la celda no se encenderá. Así las actividades relativas de la excitación e inhibición de la sinapsis chocan en una neurona, donde se determina la frecuencia de los impulsos transmitidos por el axón de la celda a otras regiones. La investigación de la inhibición sinaptica enseñó que puede ser por medio de la proyección de axón's de neuronas a centros distantes o alternativamente los axón's de las celdas de neuronas que están en la vecindad inmediata. Por ejemplo, el axón de una neurona motor en la cuerda espinal de una rama colateral que hace una sinapsis exitatoria dentro de una pequeña celda de con un axón corto es llamada interneurona; la interneurona entonces hace la inhibición de la sinapsis detrás y debajo de la neurona motor.

A.2.2.- SINAPSIS DENDRODENDRITICA.

El mecanismo de acción de la celda de gránulo era todavía desconocido en 1962, la importancia de las dendritas en la integración de la actividad de la sinapsis neuronal, y lo más importante es que la corriente eléctrica através de la sinapsis activa las dendritas y fluye por todo el árbol dendritico.
Estas corrientes "electrónicas" son el eslabón funcional entre los puntos de entrada de la sinapsis en las dendritas y el punto de generación de impulsos. Se demostro que las corrientes eléctricas y los cambios potenciales asociados con ellas pueden rigurosamente describir como llevar la cuenta de las propiedades eléctricas de las dendritas y la geometría de sus ramas. Estos métodos tienden ahora a ser la base aceptada para analizar y entender la integración de la sinapsis en las dendritas para todo el sistema nervioso.

Se construyo un modelo de computadora de las celdas que de manera realista representen sus propiedades eléctricas y geométricas. Se empezó con el axón y el cuerpo de la celda, se modelo la propagación de la actividad en las dendritas. De acuerdo al modelo, a la llegada de un impulso se activo la celda, la excitación de la sinapsis y la dendrita secundaria de la celda y la parte posterior de otra celda. Esta entrada de excitación de la sinapsis despolarizo una celda (esta, descarga su voltaje a través de su membrana exterior) y activo la sinapsis de inhibición detrás de otra celda produciendo una observable y muy larga inhibición. Se conduce la despolarización eléctrica gradualmente por todo el árbol dendritico de la celda, se activan las celdas al estar cercanas a la sinapsis de inhibición para que estas sean activadas correctamente.

Este descubrimiento de tales sinapsis dendrodendriticas, contradijo la doctrina clásica que la celda puede solamente recibir señales con sus dendritas y cuerpo de la celda y las transmite por su axón, sugirió que las neuronas puede comunicarse entre si por sus dendritas sin la intervención de axón o un impulso de nervio.

Los científicos han realizado investigaciones sobre la existencia de circuitos dendrodendritico y han sido difícil de conciliar con las nociones tradicionales de la organización de neuronas, las basadas en circuitos formados por axóns. Los circuitos dendrodendriticos han sido llamados "primitivos" "no usuales" e "informales". Las evidencias disponibles indican, sin embargo, que estas sinapsis dendrodendriticas son semejantes con respeto a las sinapsis que se realizan por axóns. Ahora aparentemente son caminos lógicos y económicos para organizar interacciones de sinapsis en un espacio mínimo. La sinapsis recíproca entre dendritas en la mayoría de casos son circuitos compactos sinapticos. Podemos por lo tanto considerar que es un microcircuito, en

el extremo opuesto de los macrocircuitos entre centros de neuronas hechos por un largo camino axónal.

A.3.- FUNCIONES SENSORIALES.

¿Cómo es la operación de los microcircuitos en las dendritas relativo a la función sensorial del sistema olfativo?. Una propiedad importante de los sistemas sensoriales es su sensibilidad. Se ha encontrado que las moléculas de substancias olorosas pueden ser encontradas en concentraciones extremadamente bajas en el aire, al encontrar que se tiene un alto grado de sensibilidad en los circuitos del bulbo olfativo al transmitir bajo estas condiciones.
En realidad la situación es algo más compleja. Las simulaciones en computadora hechas recientemente sugiere que estos diferentes microcircuitos del bulbo olfativo difieren marcadamente en la sensibilidad electrónica de su flujo de corriente sinaptica. Por ejemplo, la sinapsis hecha por las dendritas secundarias de la celda de mitral parece tener una sensibilidad relativamente baja, estas son activadas por la propagación generada de impulsos en las celdas de dendritas primarias como entrada de las celdas olfativas receptoras. En contraste, las sinapsis hechas por las dendritas de celda de granulo parecen tener una sensibilidad relativamente alta, desde allí es activado gradualmente el potencial sinaptico. La alta sensitividad de la sinapsis también está presente en el glomérulo olfativo. Es necesario saber más de las propiedades electrónicas de las dendritas, sin embargo, la sensitividad anterior de las sinapsis dendrodendritica en el bulbo olfativo puede ser una característica cuantitativa.

Estos trabajos sobre las inhibiciones muestra que son importantes para formar las respuestas de las celdas de mitral, para identificar las contribuciones específicas de los circuitos periglomerular y las celdas de gránulo de esta inhibición. Parece difícil pero se debe tener habilidad para distinguir uno olor de otro pues depende de las interacciones entre la actividad de excitación e inhibición en los circuitos dendriticos del bulbo olfativo. Un acceso enteramente diferente ha sido correlacionar la actividad de los circuitos en diferentes partes del bulbo olfativo con estímulos olfativos diferentes. Por ejemplo como las celdas de nervio están fisiológicamente activas consumiendo glucosa como combustible, se puede identificar la actividad de estos circuitos, proporcionandoles un químico derivativo de glucosa. La glucosa derivada esta rotulada con átomos de un isótopo radioactivo, para que se revele su sitio al tener radioactividad. Se descubre la radiactividad al rebanar el tejido fino en secciones delgadas y colocarlo en una película fotográfica, después de la exposición de una semana las

celdas radioactivas han obscurecido la emulsión de la película en proporción a su actividad fisiológica.

Con la ayuda de esta técnica se ha encontrado que la estimulación del olor esta asociada con modelos de actividad especiales en el bulbo olfativo. El punto de actividad se encuentra precisamente en grupos de glomérulos, que tiene una alta densidad de dendritas y sinapsis. El método de la glucosa, es una técnica de la bioquímica disponible para los neurocientificos, y promete estar vigente durante mucho tiempo correlacionando y organizando la sinapsis con sitios y niveles de actividad funcional.

A.4.- DISTRIBUCION DE MICROCIRCUITOS.

Un tipo de interneuronas en la retina, se descubrio en la celda amacrine, por la falta de un axón, se encontro una sinapsis de entrada recíproca entre las dendritas de las otras celdas y estas neuronas que se les conoce como celdas bipolares. En el transcurso de los años el trabajo en el bulbo olfativo y la retina dan muchas evidencias de tener acumulaciones de tipos semejantes de circuitos sinapticos y dendritas en otras partes del sistema nervioso. La mayoría de los neuroanatomistas pensaron que la sinapsis que existia era hecha solamente por cuerpos de celda de axón o dendritas, y ellos tienen que retractarse (y en unos casos revisar conclusiones) e incluir la sinapsis hechas por dendritas. Ahora, sin embargo, la lista de circuitos de sinapsis hechas por dendritas es bastante larga, y se puede decir que estos circuitos han sido hallados en parte de los sistemas nerviosos.

Dentro del tálamo ascienden las terminales del axón que fabrican las sinapsis dentro de las dendritas de ambos axóns largos que retransmite a neuronas y entre neuronas; las neuronas están fabricando ambos sinapsis dendrodendritica y axodendriticas en las neuronas de relevo. Por lo tanto aparece mucha de la información fluyendo en la corteza cerebral esta es procesada por microcircuitos al nivel talámico. La sinapsis dendrodendritica también ha sido encontrada en regiones que regulan movimientos en el mono, tal como el área motor de la corteza cerebral y el ganglio basal del cerebro y el mesocéfalo, los cuales han estado implicados en el desarreglo de movimientos, estos ocurren en la enfermedad de Parkinson. Estas regiones fueron difíciles de estudiar, y la importancia cuantitativa de las dendrodendritas y otros tipos de circuitos encontrados todavía tienen que ser establecidos. Un ejemplo final es el núcleo supraquiasmatico, una región pequeña en la parte delantera del hipotálamo arriba del quiasma óptico (la región donde el nervio óptico de un ojo encuentra el nervio óptico del otro). Se sabe muy poco sobre el papel que esta región juega en la conducta cíclica, tal como los ciclos diarios

fisiológicos del cuerpo, para controlar la emisión de ciertas hormonas. Por lo tanto aparecen esas sinapsis dendrodendriticas y puede trasmitir solamente una rápida información procesada.

A.5.- OTRA INTERACCION LOCAL.

En esta cuenta de microcircuitos se han acentuado las interacciones dendrodendriticas, pero también puede existir un número de sitios donde interaccionen los axóns (sinapsis axonaxonica) que se sospecha serán encontrados. En forma semejantemente, se a enfocado la atención para graduar los potenciales en microcircuitos, pero todo - o - nada, de los impulsos de nervio se dan en los axóns, en racimos de axóns colaterales y en unas dendritas. En suma, aunque la mayoría de las sinapsis operan a través de neurotransmisores, hay un número de regiones especiales llamadas confluencias de división donde la transmisión es mediada por el flujo directo de corriente eléctrica. Finalmente, hay interacciones entre neuronas, estas tienen un lugar independiente de puntos específicos de contacto; incluye el flujo continuo de substancias dentro de las celdas de nervio y a través de sus membranas, y los campos eléctricos que son instalados cuando las poblaciones de celdas de nervio son activadas en conjunto. Todos estos fenómenos contribuyen en las operaciones funcionales dentro del micro medio ambiente del sistema nervioso.

BIBLIOGRAFIA

Ackey D., G. Hinton, and T. Sejnowski (1985), "A learning Algorithm for Boltzmann Machines", Cognitive Sci., vol. 9.

Ahalt S., A. Krishnamurth, P. Chen, and D. Melton (1990), "Competitive Learning Algorithms for Vector Quantization", Neural Networks, Vol. 3.

Amari S. (1971), "Characteristics of Randomly Connected Treshold Element Network and Network Systems", Proc. IEEE, vol. 59, Jan.

Amari S. (1972), "Learning Patterns and Patterns Sequences by Self-Organizing Nets of Threshold Elements", IEEE Trans. Computer, vol. C-21, Nov.

Amari S. (1983), "Field Thery of Self-Organizing Neural Nets", IEEE Trans. Systems Man, Cybernet, vol. SMC-13, Sept/Oct.

Anderson J. (1970), "Two Models for Memory Organization Using Interractives Traces", Math. Bioci., vol. 8.

Anderson J. (1990), "Knowledge Representation in Neural Networks", Al Expert, Fall.

Anderson J., J. Silverstein, S. Rotz, and R. Jones (1977), "Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model", Pych. REv., vol. 84.

Barto A. (1984), "Simulation Experiments with Goal-Seeking Adative Elements", Air Force Wright Aeronautical Laboratory, Technical Report AFWAL-TR-84-1022.

Barto A. (1985), "Learning by Statical Comparation of Self-Interested-nueral-like Computing Units", Human Neurobiol., vol. 44.

Barto A., R. Sutton, and C. Anderson (1983), Neuron-Like Adaptive Elements that Can Solve Difficult Learning Control Problems", IEEE Trans Systems, Cybernet, vol. SMC-15, Sept/Oct.

Carpenter G., and S. Grossberg (1987), "A massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", Computer Visions, Graphics,and Image Understanding, vol. 37.

Hopfield J. (1982), "Neural Networks and Physical Systems with Emergent Collective Computation Abilities", Proc. Nat. Acad. Sci. U.S.A., vol. 79.

Hopfield J., and D. Tank (1985), "Neural Computation of Decisions in Optimization Problems", Biol. Cybernet., vol. 52.

Kanerva P. (1988), Sparse Distributed Memory, Cambrige, MA:MIT Press.

Kirkpatick S., C. Gelatt, and M.Vecchi (1983), "Optimization by Simulated Annealing", Science, vol. 220.

Kohonen T. (1972), "Correlation Matrix Memories", IEEE Trans. Computer, vol. c-21, Apr.

Kohonen T. (1984), Self-Organization and Associative Memory, Berlin:Springer-Verlag.

Kohonen T. (19986), "Learning Vector Quantization for Pattern Recognition", Helsinki university of Technology, Technical Report No. TKK-F-A601.

Kosko B. (1986), "Differential Hebbian Learning", in AIP Conf. Proc. 151: Neural Networks for Computing, J. Denker, Ed., New York: American Institute of Physics.

Kosko B. (1986), "Fuzzy Entropy and Conditioning", Information Csi., vol. 40.

Kosko B. (1988), "Bodirectional Associative Memories", IEEE Trans. Systems, Man, Cybernet, vol., SMC-18, Jan/Feb.

Kosko B. (1991), Neural Network for Signal Processing, Prentice Hall, Engelwood Cliffs, NJ.

Lee S. and R. Kill (1989) "Bidirectional Continuous Associator Based on Gaussian Potential Function Network", in Proc. IEEE/INNS Int. Joint Conf. Neural Networks, vol. 1.

Linde Y., A. Buzo and R. M. Grray (1980), "An Algorithm for Vector Quantizer Desing", IEEE Trans. Communications, vol. 28, no. 1.

Malsburg C. v. d. (1973), "Self-Organization of Orientation Sensitive Cells in the Striate Cortex", Kybernetik, vol. 14.

Mandani E. (1987), "Application of Fuzzy Logic to Aproximate Reasoning Using Linguistic Synthesis", IEEE Transsactions on Computers, vol., C-26, Dec.

Maren A., C. Harston and R. Pap (1990), Handbook of Neural Computing Applications, San Diego, CA: Academic Press.

Carpenter G., and S. Grossberg (1987), "ART2: Self-Organization of Stable Category REcognition Codes for Analog Input Patters", Appl., Optics, vol. 26

Cohen M., and S. Grossberg (1983), Absolute Stability of Global Pattern Formation and Parallel Storage by Competitive Neural Networks", IEEE trans., Systems, Man. Cybernet, vol. SMC-13, Sept./Oct.

DeSieno D. (1988), "Adding a Conscience to Competitive Learning in Proc. 1988 Int. Conf. Neural Networks, vol. 1.

Dwayne Phillips (1992), "The Foundation of Neural Networks: The Adaline and Madaline", The C Users Journal, Sep.

Eberhart R. (1990), "Standardization of Neural Netrwork Terminoology", IEEE Tras., Neural Networks, vol. 1.

Elliot D. (1967), Handbook of Digital Signal Processing Engineering Applications, San Diego, CA: Academic Press.

Fkunuga K. (11986), "Statistical Pattern Classification", in Handbook of Pattern Recognition and Image Proc., T. Young and K. Fu. Eds., San Diego, CA: Academi Press.

Fukushima K. (1988), "Noecognitron: A Hierarchical Neural Netwrok Capable of Visual Pattern Recognition", Neural Network, vol. 1.

Geral D. Fischbach (1992), "Mind and Grain", Scientific American, Septiembre.

Gray R. (1984), "Vector Quantization", IEEE ASSP Mag., Vol. 1.

Grossberg S. (1969), "On the Serial Learning of Lists", Math., Biosci., vol. 4.

Grossberg S. (1970), "Neural Pattern Discrimination", J. Theoret. Biol., vbol. 27.

Grossberg S. (1982), Studies of Ming and Brain, Booston: Reidel.

Hebb, D. (1949), Organization of Behavior, New York: Wiley.

Hecht-Nielsen R. (1990), Neurocomputing, Reading, MA: Addison-Wesley.

Hertz J. (1990), Introduction to the Theory of Neural Computation, Reading, MA: Addison-Wesley.

McEliece R., E. POsner, E. Rodemich and S. Venkatesh (1987), "The capacity of the Hopfield Associative Memory", IEEE Trans. Information Theory, vol. IT-33, July.

Nguyen D., and Widrow B. (1989), "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks", Proccedings of International Joint Conference on Neural Networks, vol., 2.

Oja E. (1982), "A Simplified Neuron Model as a Principal Component Analyzer", J. Math. Biol., vol 15.

Oja E. (1989), "Neural Network, Principle Components, and Susspaces", Int. J. Neural Networks, vol. 1.
Pao Y. (1989), Adaptive Pattern Recognition and Neural Networks, REading, MA: Addison-Wesley.

Parker D. (1982), "Learning Logic", Stanford University, Dept. of Electrical Engineering, Invention Report 581-64, Oct.

Robbins H. and S. Monro (1951), "A Stochastic Approximation Method", Ann. Math. Statist., vol. 22.

Robinson A., M. Niranjan and F. Fallside (1988), "Generalizing the Nodes of the Error Propagation Network", Cambridge University Engineering Department, Technical Report CUED/F-INENG/TR.25.

Rosenblatt F. (1962), Principles of Neurodynamics, Washington, DC: Spartan Books.

Rozoner L. (1969), "Random Logic Networks I, II, III", In Automatic Remote Control, vols. 5-7.

Ruck D., S. Rogers, M. Kabrisky, U. Maybeck and M. Oxley (1990), "Comparative analysis of Backpropagation and the Extended Kalman Filter for TRaining Multilayer Perceptrons", IEEE Trans. Pattern Ana

Rumelhart D., G. Hinton and R. Williams (1986), "Learning Representations by Backpropagation Errors", Nature, vol. 323.

Sanger T. (1989), "Optimal Unsupervised Learning in a Single-Layer Linear feedfordward Neural Network", Neural Networks, vol. 2.

Sejnowski T. (1977), "Storing Covariance with Nonlinearly Interacting Neurons", J. Math. Biol., vol. 4.

Simpson P. (1990), Artificial Neural Systems Foundations, Paradigms, Applications and Implementations, Elmsfrod, NY: Pergamon Press.

Simpson P. (1990), "Fuzzy Adaptive Resonance Theory", Presented at Southern Illinois Neuroengineering Workshop, Sept., and Published as General Dynamics Technical Report GDE-ISG-PKS-010, Apr. (revis

Simpson P. (1990), "Higher-Ordered and Intraconnected Bidirectional Associative Memories", IEEE Trans. Systems, Man. Cybernet., vol. 20.

Simpson P. (1991), "Fuzzy Min-Max Classification with Neural Networks", Huristic, vol. 4.

Simpson P. (1991), "Fuzzy Min-Max Neural Networks", in Proc. 1991 Int. Joint Conf. Neural Networks (Singapore).

Simpson P. (1992), "Fuzzy Min-Max Neural Networks: 1. Classification", IEEE Trans. Neural Networks, in Press.

Singhal S. and L. Wu (1989), "Training Multi-Layer Perceptrons with the Extended Kalman Algorithm", in Advences in Neural Information Processing Systems 1, D. Touretzky, Ed., San Mateo, CA: Kaufmann

Specht D. (1990), "Probabilistic Neural Networks", Neural Networks, vol. 3.

Steinbuch K. and U. Piske (1963), "Learning Matrices and their Applications", IEEE Trans. Electronic Computers, vol. EC-12.

Sutton R. and A. Barto (1981), "Toward a Modern Theory of Adaptive Networks: Expectation and Prediction", Psych. Rev., vol. 88.

Taber W., and Siegel M. (1987), "Estimation of Expert Weights Using Fuzzy Cognitive Maps", Proceedings of the IEEE International Conference on Neural Networks, vol. 2, June.

Tank D. and J. Hopfield (1986), "Simple 'Neural' Optimization Networks: A/D Convertor, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Trans. Circuits Systems, vol. CAS-33, May.

Tesauro G. (1986), "Simple Neural Models of Classical Conditioning", Biol. Cybernet., vol. 55.

Werbos P. (1974), "Beyong Regression", Ph. D. Dissertation, Harvard University, Cambridge, MA.

White H. (1989), "Learning in Neural Networks: A Statistical Perspective", Neural Computation, vol. 1.

White H. (1990), "Neural Network Learning and Statistics", AI Expert, Fall.

Widrow B. and M. Hoff (1960), "Adaptive Switching Circuits", in 1960 WESCON Convention Record: Part IV.

Widrow B. and R. Winter (1988), "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", IEEE Computer Mag., Mar.

Widrow B. and S. Stearns (1985), Adaptive Signal Processing, Englewoood Cliffs, NJ: Prentice-Hall.

Widrow B., N. K. Gupta and S. Maitra (1973), "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems", IEEE Trans. Syst. Man, Cybernetics, vol. SMC-3.

Williams R. (1986), "Reinforced Learning in Connection to Networks: A Mathematical Analysis", University of California, Institute for Cognitive Science, Technical Report No. 8605.

Willshaw D. (1980), "Holograpy, Associative Memory, and Inductive Generalization", in Parallel Models of Associative Memory, J. Aderson and G. Hinton, Eds., Hillsdale, NJ: Lawrence Eribaum.

Young T. and K. Fu (1986), Handbook of Pattern Recognition and Image Processing, San Diego, CA: Academic Press.