UNIVERSIDAD DE GRANADA
E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación e
Inteligencia Artificial

MEMORIAS ASOCIATIVAS DIFUSAS: DISEÑO E IMPLEMENTACIÓN

TESIS DOCTORAL

Antonio Bautista Bailón Morillas

Granada, Abril 2001

MEMORIAS ASOCIATIVAS DIFUSAS: DISEÑO E IMPLEMENTACIÓN

Antonio Bautista Bailón Morillas

TESIS DOCTORAL

Directores
Dr. D. Miguel Delgado Calvo-Flores
Dr. D. Waldo Fajardo Contreras

Abril 2001

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

La memoria *Memorias Asociativas Difusas: Diseño e Implementación*, que presenta Antonio Bautista Bailón Morillas, para optar al grado de Doctor en Informática, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, bajo la dirección de los profesores Dr.D.Miguel Delgado Calvo-Flores y Dr.D.Waldo Fajardo Contreras.

Granada, Abril de 2001

Firmado: Dr.D.Miguel Delgado Calvo-Flores

Firmado: Dr.D.Waldo Fajardo Contreras

Firmado: D.Antonio Bautista Bailón Morillas

A Pepe y Carmen

MEMORIAS ASOCIATIVAS DIFUSAS: DISEÑO E IMPLEMENTACIÓN

Antonio Bautista Bailón Morillas

AGRADECIMIENTOS

Quisiera mostrar mi agradecimiento a mis directores Miguel Delgado y Waldo Fajardo, que no sólo han guiado el desarrollo de esta Tesis Doctoral sino que además me han demostrado una paciencia y apoyo por los que estaré siempre agradecido.

A mis padres por todo su amor y confianza y porque siempre se sacrificaron para que yo lograra mis objetivos.

A mis hermanos David y Ana que siempre se sintieron orgullosos de mí, consiguiendo que me superara día a día.

A toda mi familia por

A mis compañeros y amigos María José, Nacho y Antonio, que día a día me han demostrado su amistad con palabras de ánimo, valiosos consejos y con la ayuda desinteresada de quienes se preocupan por mí.

A mis compañeros de departamento por formar un excelente grupo de personas con quien trabajar.

A todos mis amigos porque siempre estuvieron conmigo cuando los necesité.

Y, sobre todo, quiero dar gracias a Bernarda, mi esposa, por todo lo que ella significa para mí.

Índice General

Índice de Figuras

Introducción

El hombre tiene la capacidad de recordar información a partir de asociaciones. Por ejemplo, unas pocas notas de una melodía pueden evocar la canción completa; el olor a rosas puede evocar la imagen de dichas flores. Además, la memoria humana puede realizar asociaciones a partir de datos incorrectos o incompletos. Por ello, al ver el fragmento de un jarrón evocamos el aspecto que tenía antes de romperse o al cambiarnos el color del pelo la gente nos sigue reconociendo.

Cuando el ser humano memoriza no necesita indicarle a su memoria en qué lugar debe almacenar la información. De igual modo, para recordar no hay que indicarle el lugar en que se encuentra lo que buscamos. Basta con "mostrarle" a nuestra memoria algo que pueda evocar el recuerdo de lo que buscamos. Por ejemplo, el nombre de una persona puede hacer que nuestra memoria recuerde el aspecto de dicha persona.

A lo largo de la historia el hombre ha tratado de construir máquinas capaces de mostrar un comportamiento inteligente. Esto ha conducido al intento de emular el comportamiento del cerebro humano. Más particularmente se ha intentado construir una memoria artificial que, al igual que la memoria humana, sea capaz de almacenar y recuperar asociaciones.

Llamamos memorias asociativas a las memorias artificiales que almacenan información mediante asociaciones de patrones[1]. Su capacidad para trabajar con información incompleta e incorrecta

[1]Aunque la memoria humana es una memoria asociativa, en adelante, al hablar de memorias asociativas nos referiremos a las memorias asociativas artificiales.

las dota de mayor robustez para usarlas en sistemas automáticos de tratamiento de la información.

Existen diversos modelos de memorias asociativas empleadas en sistemas automáticos de tratamiento de información. Dichos modelos permiten almacenar y recuperar información de un modo similar al empleado por el cerebro humano.

Si reflexionamos sobre el tipo de información que el ser humano suele memorizar o emplear en sus razonamientos nos encontramos con que maneja gran cantidad de conceptos imprecisos. Esto, que podría parecer una limitación, le permite afrontar con éxito tareas de muy alta complejidad en las que fracasan computadores mucho más precisos que el hombre. Esta relación inversa entre la precisión y la complejidad está expresada en el principio de incompatibilidad de Zadeh[59].

Si queremos modelar sistemas complejos necesitamos reducir la precisión con que deben ser abordados. De este modo se hacen necesarias memorias que sean capaces de almacenar información expresada de forma imprecisa.

Las memorias asociativas difusas son memorias asociativas que nos permiten almacenar y recuperar información imprecisa. Constituyen un elemento muy importante de los sistemas automáticos de tratamiento de la información debido a su comportamiento similar al presentado por la memoria humana.

Frente a las virtudes señaladas existe un grave problema en las memorias asociativas: la baja capacidad de almacenamiento. Cuando hablamos de la capacidad de una memoria nos referimos al número de patrones que pueden ser memorizados para ser después perfectamente recuperados. Cuando se alcanza el límite de la capacidad de una memoria asociativa, los nuevos patrones que se memoricen interferirán con los ya almacenados provocando que se recuperen patrones espurios, es decir, patrones que no corresponden a ninguno de los almacenados.

Existe otro problema que afecta a la capacidad de almacenamiento de las memorias asociativas que es la dependencia del tipo de patrones que se podrán memorizar. Cualquier limitación que se imponga sobre los patrones que se podrán memorizar implica una limitación a la capacidad de almacenamiento. Si se memorizara un

patrón que no cumple los requisitos impuestos, otros patrones almacenados se verán afectados y se recuperarán patrones espurios.

La solución al problema de la capacidad se logra evitando los factores que la limitan: los estados espurios y la dependencia de los datos. Para ello hemos desarrollado en esta tesis un modelo de memoria asociativa que no pone limitación al tipo de datos que se podrán almacenar al mismo tiempo que impide que los patrones almacenado interfieran entre sí dando lugar a patrones espurios.

Una vez obtenida una memoria asociativa "crisp" con una buena capacidad de almacenamiento la hemos adaptado para que sea capaz de memorizar información imprecisa. Esta memoria asociativa difusa constituirá un elemento muy útil en cualquier sistema de tratamiento de información empleado en razonamiento aproximado por su alta capacidad y por su habilidad para trabajar con información imprecisa.

Con estos objetivos generales esta memoria se estructura del siguiente modo:

- En el capítulo 1 se van a repasar los modelos más importantes de memorias asociativas analizando su topología, su funcionamiento en las fases de almacenamiento y recuperación de la información, sus ventajas e inconvenientes. El capítulo está dividido en dos apartados principales. En primer lugar se estudian los modelos existentes de memorias asociativas discretas y a continuación los modelos de memorias asociativas difusas.

- El capítulo 2 se dedica a las variables lingüísticas, un elemento muy importante para el razonamiento aproximado y la representación de información imprecisa. Además se estudia un método de codificación que nos va a permitir representar de forma discreta la información difusa representada por las variables lingüísticas. Este método denominado codificación por discretización incremental será empleado en el capítulo 4 para la obtención de un modelo de memoria asociativa difusa a partir de un modelo discreto.

- En el capítulo 3 se presenta la Memoria Asociativa Clasificadora. Éste es un modelo discreto de memoria asociativa que,

al evitar la interferencia entre los patrones sin imponerles limitaciones, logra una elevada capacidad de almacenamiento. En el capítulo se analizan diversos problemas:

  - Veremos el modo en que se resuelven los problemas de estados espurios y dependencia de los datos para lograr una alta capacidad de almacenamiento.
  - Se estudiará su topología justificando la elección de los distintos tipos de elementos que la componen.
  - Se explicarán los procesos de almacenamiento y recuperación junto con un ejemplo que facilitará su comprensión.
  - Probaremos que todos los patrones almacenados se pueden recuperar perfectamente y que el proceso de recuperación obtiene una clasificación del patrón de entrada por el vecino más cercano.
  - Analizaremos la eficiencia de la memoria comparándola con la memoria asociativa bidireccional.
  - Por último mostramos diversas modificaciones que se pueden realizar al modelo para mejorar su comportamiento tanto en la fase de memorización como en la de recuperación.

- En el capítulo 4 se va a obtener un modelo de Memoria Asociativa Clasificadora que pueda almacenar información imprecisa. Para ello se presentan dos posibilidades. Por un lado se usará un método que permita usar la CLAM para almacenar información difusa codificada de forma discreta. Alternativamente, se estudiará cómo modificar la CLAM para que pueda almacenar información difusa codificada como tal.

  - En primer lugar veremos cómo emplear el método de codificación por discretización incremental para poder usar una Memoria Asociativa Clasificadora para almacenar información imprecisa. El empleo del método a modo de filtro nos permitirá usar la memoria sin realizar ningún

tipo de modificación. Con un ejemplo se comprobará la efectividad de la solución.

- A continuación abordaremos la alternativa de realizar las modificaciones oportunas a una CLAM para adaptarla a la memorización de información difusa. Estas importantes modificaciones afectarán a la topología del modelo así como a los procesos de almacenamiento y recuperación. Veremos cómo la memoria asociativa difusa obtenida puede emplearse como elemento estructural de agrupaciones complejas que aumentan la funcionalidad de modelo. Se presentarán ejemplos que muestren el almacenamiento de patrones y reglas difusas.

- La memoria termina con la presentación de conclusiones y la exposición de las principales lineas de investigación futuras.

Capítulo 1

Memorias Asociativas

En un sentido amplio, el aprendizaje es el proceso de creación de asociaciones entre patrones que pueden ser del mismo tipo (asociación entre la imagen de un automóvil y el logotipo del fabricante) o de tipos distintos (asociación del olor y de la imagen de una rosa).

Por ejemplo, cuando aprendemos a leer realizamos asociaciones entre las letras y sus fonemas. Somos capaces de reaccionar ante imágenes que, aunque las consideramos iguales, no son idénticas; si la letra es de distinto tamaño o ha sido impresa usando otra fuente o estilo, somos capaces de asociarle el sonido correcto. Del mismo modo, si oímos un fonema pronunciado por distintas personas, a pesar de que los sonidos son diferentes, reconocemos no sólo el fonema sino la grafía asociada al mismo.

Estas asociaciones tienen la importante característica de que ante un estímulo similar a un patrón ya aprendido se invoca el patrón asociado al mismo. Este modo de almacenar y recuperar la información en una memoria mediante asociación de patrones, define el comportamiento de las memorias asociativas.

Las memorias asociativas son memorias en que la información se almacena mediante correlación o asociación entre patrones de tal modo que la recuperación de dicha información se realiza a partir de datos. De este modo las relaciones almacenadas determinan cuál es el patrón asociado al patrón presentado.

Las características de las memorias asociativas hacen que éstas sean especialmente útiles en el tratamiento de información inco-

rrecta e incompleta ya que las asociaciones almacenadas ayudan a encontrar un patrón almacenado similar y su correspondiente patrón asociado.

Cuando tratamos de redes de neuronas artificiales podemos distinguir entre memorias asociativas y redes neuronales.

Las redes neuronales son modelos que, entrenados con un conjunto de patrones de entrenamiento, ante un estímulo de entrada dan una respuesta interpolada a la función definida por los patroneas almacenados.

Las memorias asociativas son aquellos modelos que permiten el almacenamiento de pares de patrones asociados y la recuperación de los mismos a partir de información incompleta e incorrecta. De este modo, la respuesta que producen es siempre el patrón almacenado que se halla más próximo a la respuesta provocada por el estímulo de entrada.

Las redes neuronales se diferencian de las memorias asociativas en que su respuesta es interpolada, es decir, no se busca el patrón almacenado más cercano. De una memoria asociativa y una red neuronal entrenadas con los mismos patrones esperaremos distintas respuestas ante un mismo estímulo de entrada.

Por ejemplo, entrenamos una red neuronal y una memoria asociativa con los siguientes patrones:

0000 (0)	000000 (0)
0001 (1)	000011 (3)
1111 (15)	101101 (45)

De una memoria asociativa se espera que ante un patrón de entrada distinto a los almacenados recupere aquél con el que guarda mayor similitud. Es decir, ante el patrón de entrada 1001 (9) respondería 0001 (1) 000011 (3) que es el patrón almacenado más parecido.

Sin embargo una red neuronal entrenada con los mismos patrones encuentra la relación existente entre los mismos (cada pareja la componen un número y su triple). De este modo, con el mismo patrón de entrada 1001 (9) la respuesta de la red neuronal sería 1001 (9) 011011 (27).

A continuación vamos a repasar algunos modelos importantes de memorias asociativas. En cada caso se estudiará:

- La topología del modelo, indicando el tipo de elemento de proceso (EP), el tipo de las conexiones y si se trata de un modelo autoasociativo o heteroasociativo.
- El proceso de aprendizaje en el que se indicará el tipo de patrones que pueden ser almacenados en la memoria y el tipo de aprendizaje de que se trata.
- El proceso de recuperación de patrones.
- Un análisis de las ventajas e inconvenientes que presenta el modelo.

1.1 Memorias asociativas discretas

1.1.1 Red de Hopfield

La red de Hopfield es un modelo de memoria asociativa introducido por J.J.Hopfield en 1982 [21] en que cada EP está conectado a todos los demás excepto a sí mismo. Los pesos de las conexiones son simétricos ($w_{ij} = w_{ji}$). Cada EP recibe una señal externa además de las procedentes de los demás elementos.

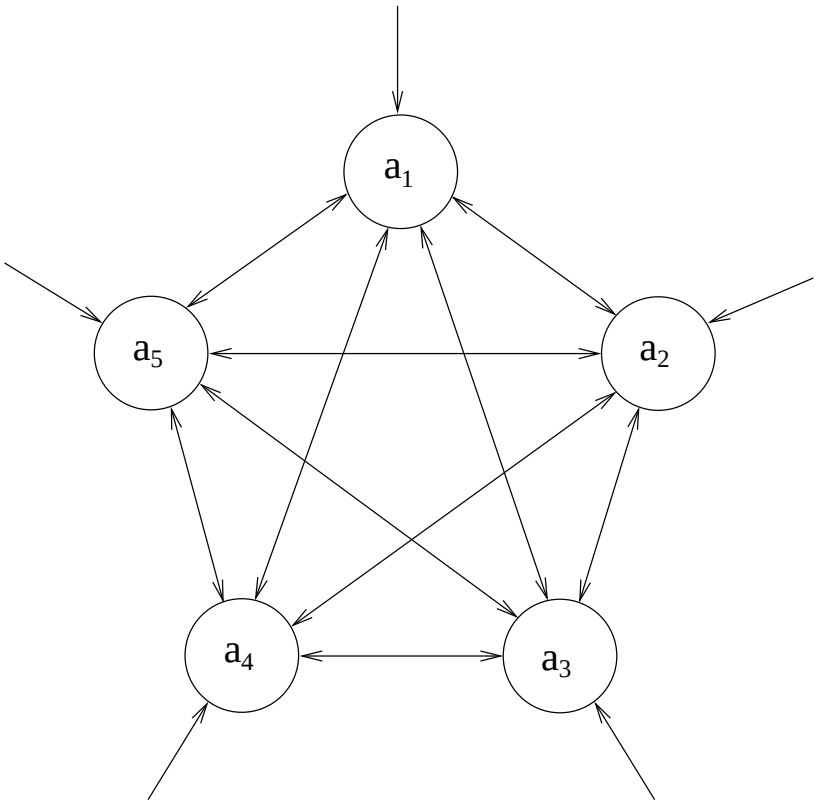
La actualización de los estados de activación de los EP se realiza de forma asíncrona por lo que en un momento dado sólo uno de los elementos podrá cambiar su estado. Este modo asíncrono de actualización permite la obtención de una función de energía que permite probar la convergencia a un estado estable de las señales de activación, es decir, la estabilidad del modelo.

1.1.1.1 Aprendizaje

Hay varias versiones del proceso de aprendizaje para la red de Hopfield.

La primera descripción de Hopfield sólo contemplaba el almacenamiento de patrones binarios. Para almacenar el conjunto de

Figura 1.1: Red de Hopfield



patrones binarios $\{A_i\}_{i=1}^{n}\,,\ A_i = (a_{i1}, \ldots, a_{ip})$, la matriz de pesos se determina del siguiente modo:

$$\begin{gathered} M = \{m_{ij}\} \\ m_{ij} = \sum_{k=1}^{n} (2a_{ki} - 1)\,(2a_{kj} - 1) \qquad \forall i \neq j\ . \\ m_{ii} = 0 \end{gathered} \tag{1.1}$$

Más tarde Hopfield aportó una nueva descripción de la red en que se permitían patrones bipolares de modo que para el conjunto de patrones bipolares $\{A_i\}_{i=1}^{n}\,,\ A_i = (a_{i1}, \ldots, a_{ip})$ la matriz de pesos

se obtiene como:

$$\begin{array}{r} M = \{m_{ij}\} \\ m_{ij} = \sum_{k=1}^{n} a_{ki} a_{kj} \qquad \forall i \neq j \ . \\ m_{ii} = 0 \end{array} \tag{1.2}$$

1.1.1.2 Recuperación

El proceso de recuperación de la red de Hopfield implica el paso reiterado de señales de activación entre los EP hasta que se converge a un estado estable en que no se producen cambios. Los pasos del proceso son los siguientes:

1. Se presenta el patrón de entrada $X = (x_1, \ldots, x_p)$.
2. Los estados de activación de los EP de la red son $y_i = x_i$.
3. Para cada elemento y_i de la red, seleccionado de forma aleatoria (todos tienen la misma probabilidad de ser seleccionados), se repiten los siguientes pasos:
   (a) Se calcula la señal que recibe el elemento como la suma de la señal de entrada más las recibidas desde los demás elementos:
   $$e_i = x_i + \sum_{j \neq i} y_j m_{ji}. \tag{1.3}$$
   (b) Se determina el estado de activación del EP comparando la señal recibida con el umbral θ_i del elemento y_i:
   $$y_i = \begin{cases} 1 & \text{si } e_i > \theta_i \\ y_i & \text{si } e_i = \theta_i \\ -1 & \text{si } e_i < \theta_i. \end{cases} \tag{1.4}$$
4. Se realiza un test de convergencia para comprobar si el proceso de recuperación ha alcanzado un estado estable. Dicho estado será aquel que no sufre modificaciones tras el proceso detallado en el paso 3. Si no se ha alcanzado la estabilidad se vuelve al paso 3.

1.1.1.3 Ventajas e inconvenientes

El principal problema que presenta la red de Hopfield es que tiene una capacidad de almacenamiento muy limitada y esto reduce de manera importante su utilidad.

La mayor ventaja es la innovación que supuso frente a los modelos desarrollados hasta ese momento y la influencia que ejerció en los modelos surgidos posteriormente.

1.1.2 Memoria asociativa lineal

La memoria asociativa lineal (LAM), introducida por J.Anderson en 1968 [1] y refinada posteriormente por J.Anderson [2, 3, 4] y T.Kohonen [20, 23, 24]. Es un modelo de dos capas heteroasociativo e interpolativo que almacena patrones espaciales analógicos arbitrarios de la forma $(A_k, B_k), k = 1, \ldots, m$ usando aprendizaje hebbiano, siendo $A_k = (a_{k_1}, \ldots, a_{k_n}), B_k = (b_{k_1}, \ldots, b_{k_p})$.

1.1.2.1 Memorización

Los pares de patrones se almacenan utilizando la suma de su producto exterior para dar lugar a la expresión que define la matriz de pesos:

$$M = \sum_{k=1}^{m} A_k^T B_k. \tag{1.5}$$

La matriz de pesos tiene n filas y p columnas de modo que cada una de sus componentes vale:

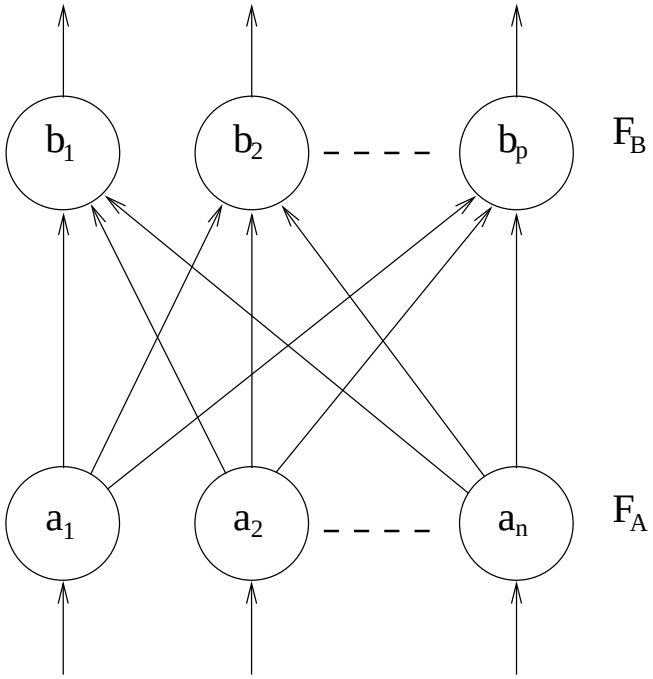
$$m_{ij} = \sum_{k=1}^{m} a_{ki} b_{kj}. \tag{1.6}$$

1.1.2.2 Recuperación

Cuando se presenta como entrada a la memoria el patrón $A = (a_1, \ldots, a_n)$ en la capa F_A, ésta responde devolviendo en la capa F_B el patrón B dado por:

$$B = AM. \tag{1.7}$$

Figura 1.2: Memoria Asociativa Lineal (LAM)



Las componentes del patrón B recuperado mediante la expresión anterior son:

$$b_j = \sum_{i=1}^{n} a_i m_{ij}. \tag{1.8}$$

La memoria asociativa lineal garantiza la perfecta recuperación de todos los patrones memorizados cuando dichos patrones son ortonormales entre sí. Para ello se deben cumplir las siguientes condiciones:

- Todos los A_k están normalizados, es decir:

$$A_i A_i^T = 1 \quad \forall i = 1, \ldots, m. \tag{1.9}$$

- Todos los A_k son ortogonales entre sí:

$$A_i A_j^T = 0 \quad \forall i \neq j. \tag{1.10}$$

Esto se puede comprobar si descomponemos la expresión de formación de la matriz de pesos en términos de señal y ruido. La señal es la excitación provocada por las parejas de patrones asociados mientras que el ruido son las interferencias provocadas por el resto de los patrones:

$$M = \underbrace{A_h^T B_h}_{señal} + \underbrace{\sum_{\substack{k=1 \\ k \neq h}}^{n} A_k^T B_k}_{ruido} . \tag{1.11}$$

Entonces el patrón recuperado cuando se presenta como entrada a la memoria el patrón A_h es:

$$A_h M = \underbrace{A_h A_h^T}_{1} B_h + \sum_{\substack{k=1 \\ k \neq h}}^{n} \underbrace{A_h A_k^T}_{0} B_k = B_h. \tag{1.12}$$

1.1.2.3 Ventajas e inconvenientes

El problema principal de la memoria asociativa lineal es el hecho de que sólo garantice la recuperación de los pares de patrones (A_i, B_i) cuando los patrones A_i o los B_i son ortonormales entre sí.

Entre las ventajas de la memoria asociativa lineal se encuentran:

- Responde en tiempo real
- Buena tolerancia a fallos (comportamiento aceptable cuando hay conexiones dañadas)
- Respuesta interpolada.

Los inconvenientes que presenta incluyen:

- Incapacidad para almacenar más patrones que el número de EP de la capa F_A

- Comportamiento impredecible cuando se sobrepasa dicha capacidad
- Incapacidad de almacenar relaciones no lineales.

1.1.3 Memoria asociativa lineal óptima

La memoria asociativa lineal óptima se debe al trabajo de W.Wee en 1968 [48] y de T.Kohonen y M.Ruohonen en 1973 [25]. Se trata de una memoria asociativa que presenta tres variantes con funcionamientos distintos:

- Codificador de patrones autoasociativo
- Acoplador de patrones heteroasociativo
- Filtro de novedad.

En todos los casos su comportamiento es siempre interpolativo.

El aprendizaje se realiza fuera de línea, es decir, se debe llevar a cabo el aprendizaje del conjunto de patrones antes de poder usar la memoria de modo que si se necesita memorizar un nuevo patrón será necesario aprender de nuevo el conjunto entero de patrones. OLAM utiliza un procedimiento de actualización síncrono basado en multiplicaciones vector-matriz y opera en tiempo discreto.

A continuación se van a exponer los procesos de codificación y de recuperación de la memoria asociativa lineal óptima en cada uno de los tres modos de actuar: acoplador heteroasociativo, codificador autoasociativo y filtro de novedad.

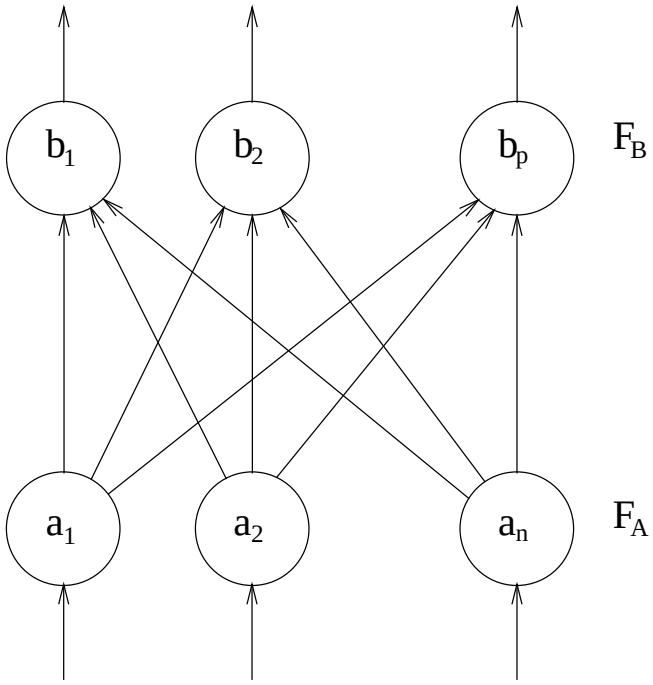
1.1.3.1 Acoplador de patrones heteroasociativo

Como acoplador de patrones heteroasociativo, la OLAM almacena pares de patrones espaciales analógicos arbitrarios $(A_k, B_k) \quad , \quad k = 1, \ldots, m$, $A_k = \left(a_1^k, \ldots, a_n^k\right)$, $B_k = \left(b_1^k, \ldots, b_p^k\right)$, usando una matriz pseudoinversa en el método de codificación.

La topología que presenta la OLAM cuando actúa como acoplador de patrones heteroasociativo está compuesta de dos capas de EP sin retroalimentación en la que los EP de una capa se corresponden con las componentes de A_k y los EP de la otra capa se

corresponden con las componentes de B_k. La topología se muestra en la figura 1.3.

Figura 1.3: Acoplador de patrones heteroasociativo



Codificación

La OLAM se puede representar mediante una matriz M que se obtiene a partir del conjunto de pares de patrones (A_k, B_k), $k = 1, \ldots, m$. Sea $\mathcal{A}$ la matriz resultante de concatenar los vectores A_k de tal modo que cada uno de ellos de lugar a una fila de la matriz:

$$\mathcal{A} = \left[A_1^T \mid A_2^T \mid \ldots \mid A_m^T\right]^T . \tag{1.13}$$

Del mismo modo, la matriz $\mathcal{B}$ la definimos como:

$$\mathcal{B} = \left[B_1^T \mid B_2^T \mid \ldots \mid B_m^T\right]^T. \tag{1.14}$$

Este grupo de patrones, codificado en una memoria asociativa lineal LAM, daría lugar a la siguiente matriz de pesos:

$$M = \sum_{k=1}^{m} A_k^T B_k = \mathcal{A}^T \mathcal{B}.$$

Puesto que la eficiencia de la LAM depende mucho de la ortonormalidad de los pares de patrones a almacenar, Wee y más tarde Kohonen y Ruohonen diseñaron un proceso de codificación mejorado en que se hacía uso de la pseudoinversa en lugar de la operación de trasposición matricial para minimizar el error cuadrático medio del paso hacia delante (*forward recall*).

Para minimizar el error cuadrático medio del paso hacia delante necesitamos encontrar la matriz $\widetilde{M}$ tal que:

$$\left\|\mathcal{B} - \mathcal{A}\widetilde{M}\right\| \leq \|\mathcal{B} - \mathcal{A}M\| \ \forall M. \tag{1.15}$$

Supongamos que las matrices $\mathcal{A}$ y $\mathcal{B}$ son cuadradas y que el número de patrones a almacenar es igual al número de EP de la capa F_A y de la capa F_B, es decir, $m = n = p$. Si existe la inversa de la matriz $\mathcal{A}$ entonces:

$$0 = \|\mathcal{B} - \mathcal{B}\| = \left\|\mathcal{B} - \mathcal{A}\mathcal{A}^{-1}\mathcal{B}\right\|. \tag{1.16}$$

Por tanto, en este caso la matriz buscada es $\widetilde{M} = \mathcal{A}^{-1}\mathcal{B}$.

Para el caso general en que no se cumple $m = n = p$, o en que no existe la inversa de la matriz $\mathcal{A}$, en lugar de hacer uso de la inversa utilizaremos la pseudoinversa $\mathcal{A}^*$ que siempre existe y es única para cada matriz.

De este modo, la matriz que representa la memoria se obtiene mediante:

$$M = \mathcal{A}^*\mathcal{B}. \tag{1.17}$$

Recuperación

La recuperación de patrones en OLAM cuando funciona como acoplador de patrones heteroasociativo usa una multiplicación simple de vector por matriz.

Sea M la matriz que representa la memoria y A un vector cuyas componentes son las señales de activación de los EP de la capa F_A. El vector B obtenido en la capa F_B viene dado por la expresión:

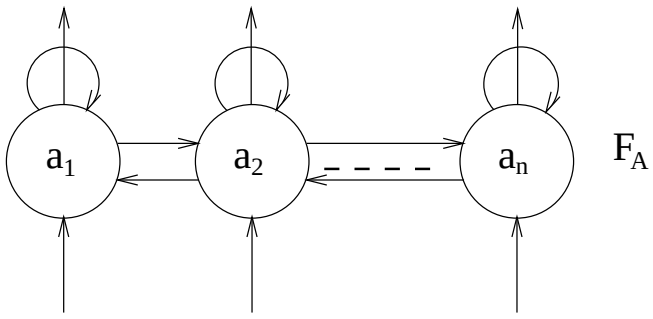
$$B = AM. \tag{1.18}$$

1.1.3.2 Codificador de patrones autoasociativo

Como codificador de patrones autoasociativo, la OLAM almacena patrones espaciales analógicos arbitrarios $A_k = \left(a_1^k, \ldots, a_n^k\right),\ k = 1, \ldots, m$.

La topología que presenta la OLAM cuando actúa como codificador de patrones autoasociativo está compuesta por una capa de EP sin retroalimentación, en que el número de EP es igual al número de componentes de cada patrón A_k a almacenar. Su topología se muestra en la figura 1.4.

Figura 1.4: Codificador de patrones autoasociativo



Codificación

El comportamiento de la OLAM como codificador de patrones autoasociativo puede verse como un caso especial del comportamiento heteroasociativo en que los pares de patrones que se almacenan son (A_k, A_k), siendo $A_k = \left(a_1^k, \ldots, a_n^k\right),\ k = 1, \ldots, m$.

Sea $\mathcal{A} = \left[A_1^T \mid A_2^T \mid \ldots \mid A_m^T\right]^T$. La matriz que representa la memoria se obtiene como:

$$M = \mathcal{A}^* \mathcal{A}. \tag{1.19}$$

En caso que el número de patrones a almacenar m y el número de componentes de cada patrón n sean iguales y que todos los patrones sean linealmente independientes entonces $\mathcal{A}^* = \mathcal{A}^{-1}$ y, por tanto, $M = \mathcal{A}^* \mathcal{A} = \mathcal{A}^{-1} \mathcal{A} = I$.

Recuperación

La recuperación de patrones en OLAM cuando funciona como codificador de patrones autoasociativo usa una multiplicación simple de vector por matriz.

Sea M la matriz que representa la memoria y A un vector cuyas componentes son las señales de activación de los EP de la capa F_A. El vector A' obtenido viene dado por la expresión:

$$A' = AM. \tag{1.20}$$

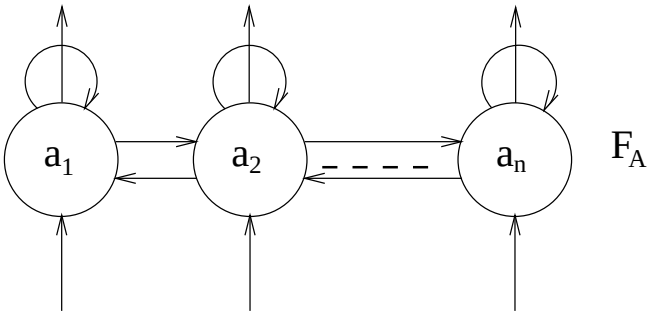
1.1.3.3 Filtro de novedad

Cuando se comporta como filtro de novedad, la OLAM almacena patrones espaciales analógicos arbitrarios $A_k = \left(a_1^k, \ldots, a_n^k\right),\ k = 1, \ldots, m$ y es capaz de determinar la diferencia existente entre las componentes de dichos patrones almacenados y las de un patrón que se presenta a la memoria. Tras el proceso de recuperación, en los EP se obtendrán valores pequeños si el patrón presentado es similar a los almacenados y se obtendrán valores mayores cuanto más diferentes sean.

La topología de la OLAM cuando actúa como filtro de novedad está compuesta por una capa de EP sin retroalimentación, en que

el número de EP es igual al número de componentes de cada patrón A_k a almacenar. Su topología se muestra en la figura 1.5.

Figura 1.5: Filtro de novedad



Codificación

El comportamiento de la OLAM como filtro de novedad hace uso de una matriz de pesos que se obtiene a partir de la diferencia entre la matriz identidad y la matriz de pesos que resulta de almacenar los patrones en la OLAM actuando como codificador autoasociativo. Sea $\mathcal{A} = \left[A_1^T \mid A_2^T \mid \ldots \mid A_m^T\right]^T$, la matriz de pesos se obtiene como:

$$M = I - \mathcal{A}^* \mathcal{A}. \tag{1.21}$$

Recuperación

En el proceso de recuperación el resultado que se obtiene es un vector Δ que contiene las diferencias encontradas entre el patrón presentado y los patrones almacenados. Cuando el patrón presentado a la memoria es A, la respuesta viene dada por la ecuación:

$$\Delta = AM. \tag{1.22}$$

1.1.3.4 Ventajas e inconvenientes

Las ventajas de la memoria asociativa lineal óptima son:

- La capacidad que muestra para dar respuestas interpolativas en tiempo real
- Buen comportamiento en la aproximación de funciones
- Suave degradación del almacenamiento cuando el número de patrones es superior al número de componentes.

Entre los inconvenientes destacan:

- El elevado tiempo de computación necesario para realizar el cálculo de la pseudoinversa de una matriz
- La amplificación lineal de la señal y el ruido
- La incapacidad que presenta para trabajar con relaciones no lineales.

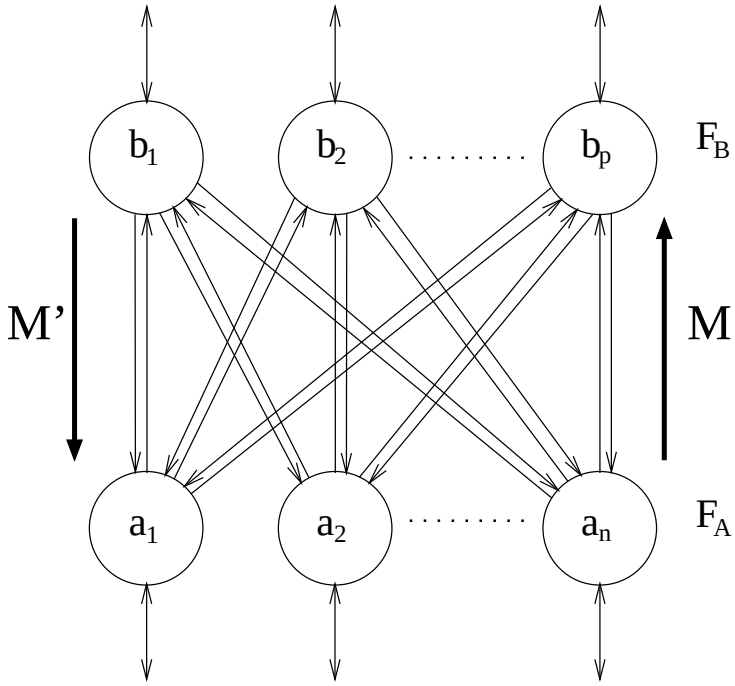
1.1.4 Memoria asociativa bidireccional

La memoria asociativa bidireccional (BAM) introducida por B.Kosko en 1988 [32] es un modelo heteroasociativo de dos capas que clasifica pares de patrones bipolares o binarios por el criterio del vecino más próximo usando aprendizaje hebbiano. Es más común el uso de EP bipolares por lo que a partir de ahora no consideraremos aquí el caso binario; sólo cabe indicar que las diferencias de funcionamiento de la memoria asociativa bidireccional son mínimas al trabajar con EP bipolares o binarios.

La BAM presenta una topología formada por dos capas de EP interconectadas entre sí por medio de conexiones ponderadas. Dichas capas se envían mutuamente unas señales de activación que dependen de los pesos de las conexiones en un proceso cíclico que termina cuando se alcanza un punto estable en que no se producen cambios. Se puede demostrar que dicho punto de estabilidad es siempre alcanzable de modo que un estado estable corresponde a un patrón almacenado.

Para almacenar el conjunto de m pares de patrones de n y p componentes $(A_k, B_k), k = 1, \ldots, m$, $A_k = (a_{k_1}, \ldots, a_{k_n}), B_k = (b_{k_1}, \ldots, b_{k_p})$ tendremos una BAM en que la capa F_A tendrá n EP y la capa F_B tendrá p.

Figura 1.6: Memoria Asociativa Bidireccional (BAM)



1.1.4.1 Memorización

Cualquier método de aprendizaje, sea supervisado o no, puede producir las matrices de pesos M y M' cuyos elementos ponderan el valor de las señales de activación recibidas por los EP.

El método más común para construir las matrices de pesos es el del aprendizaje hebbiano en el que los pares de patrones se almacenan utilizando la suma de su producto exterior para dar lugar a

la expresión que define la matriz de pesos del paso hacia adelante:

$$M = \sum_{k=1}^{m} A_k^T B_k. \tag{1.23}$$

La matriz de pesos tiene n filas y p columnas de modo que cada una de sus componentes vale:

$$m_{ij} = \sum_{k=1}^{m} a_{k_i} b_{k_j}. \tag{1.24}$$

La matriz de pesos del paso hacia atrás con aprendizaje hebbiano viene definida por la expresión:

$$M' = \sum_{k=1}^{m} B_k^T A_k = M^T. \tag{1.25}$$

1.1.4.2 Recuperación

Para comenzar se presenta como entrada a la memoria el patrón A en la capa F_A y/o el patrón B en la capa F_B. La memoria inicia entonces un proceso cíclico que termina cuando se llega a un estado estacionario en el que en las capas F_A y F_B está el par recuperado. El proceso de recuperación consta de los siguientes pasos:

1. Se presenta el patrón (A,B) en las capas F_A y F_B. Dicho patrón puede estar parcialmente deteriorado o incompleto debido al ruido o a la falta de información.

2. Se propagan las señales de activación de la capa F_A hacia la capa F_B síncronamente (o asíncronamente) mediante la matriz de pesos M. La capa F_B recibe:

$$B = AM. \tag{1.26}$$

3. Se calculan las señales de activación de los EP de la capa F_B. Para ello se convierten en bipolares las señales recibidas

desde la capa F_A mediante la función ϕ:

$$B' = \phi(B) = (b'_1, \ldots, b'_p) \quad b'_i = \begin{cases} +1 & \textbf{si} \quad b_i > 0 \\ 0 & \textbf{si} \quad b_i = 0 \\ -1 & \textbf{si} \quad b_i < 0. \end{cases} \tag{1.27}$$

4. Las señales de activación de la capa F_B se propagan hacia la capa F_A síncronamente (o asíncronamente) mediante la matriz de pesos:
$$A' = B'M'. \tag{1.28}$$

5. Se calculan las señales de activación de los EP de la capa F_A. Para ello se convierten en bipolares las señales recibidas desde la capa F_B mediante la función ϕ:

$$A'' = \phi(A') = (a''_1, \ldots, a''_n) \quad a''_i = \begin{cases} +1 & \textbf{si} \quad a'_i > 0 \\ 0 & \textbf{si} \quad a'_i = 0 \\ -1 & \textbf{si} \quad a'_i < 0. \end{cases} \tag{1.29}$$

6. Se repiten los pasos 2-5 hasta que se alcanza un estado de equilibrio en que no se producen cambios (se ha demostrado que dicho estado de estabilidad siempre se alcanza).

1.1.4.3 Estabilidad

La estabilidad global se define como la eventual estabilización de las señales de activación de todos los EP a partir de cualquier activación inicial. Es decir, en algún momento se alcanzará un estado en que no cambian las señales de activación de los EP. A dicho estado de las señales de activación se le denomina punto fijo. En una memoria asociativa, los puntos fijos determinan los patrones almacenados de modo que lo deseable es que dichos puntos fijos se correspondan con los patrones que queremos memorizar.

Para probar la estabilidad del sistema se puede usar el método directo de Lyapunov [13] que nos dará una condición necesaria (no suficiente) para que se cumpla dicha estabilidad.

1.1.4.4 Ventajas e inconvenientes

El problema principal de la BAM es el hecho de que no garantice la recuperación de los pares de patrones almacenados. Diversos autores han propuesto soluciones para este problema pero el incremento de la capacidad que se consigue es mínimo de modo que la mejora sólo merece la pena cuando los patrones a memorizar cumplen determinadas condiciones.

La capacidad de la memoria asociativa bidireccional estimada por Haines & Hecht-Nielsen [18] es:

$$\frac{q}{4\lg(q)} \quad q = \min(n, p), \tag{1.30}$$

siendo n y p el número de elementos de proceso de las capas F_A y F_B.

Los inconvenientes que presenta la BAM incluyen:

- Su incapacidad para almacenar muchos patrones.
- Está limitada a patrones bipolares y binarios.

Entre las ventajas de la memoria asociativa bidireccional se encuentran:

- Su probada estabilidad
- La facilidad con que se pueden añadir nuevos patrones rápidamente.

1.1.4.5 Métodos de entrenamiento

Entre los distintos métodos aparecidos para incrementar la capacidad de la memoria asociativa bidireccional se encuentran la codificación extendida, el entrenamiento múltiple secuencial y el entrenamiento múltiple por programación lineal.

Codificación extendida

El método de codificación extendida [47] garantiza la perfecta recuperación de todos los pares de patrones. Para ello los patrones se aumentan con unas componentes que eliminan el ruido provocado por la interferencia entre unos patrones y otros.

Definición 1. Un conjunto de pares de patrones $\{(A_i, B_i)\}_{i=1...n}$ está libre de ruido si se cumple:

$$\sum_{j \neq i} \left(A_i A_j^T\right) B_j = 0 \qquad \forall i \tag{1.31}$$

$$\sum_{j \neq i} \left(B_i B_j^T\right) A_j = 0 \qquad \forall i. \tag{1.32}$$

Definición 2. El conjunto de pares de patrones $\{(A_i, B_i)\}_{i=1...n}$ está estrictamente libre de ruido si los patrones son ortogonales, es decir, si se cumple:

$$A_i A_j^T = 0 \qquad \forall i, j, i \neq j \tag{1.33}$$

$$B_i B_j^T = 0 \qquad \forall i, j, i \neq j. \tag{1.34}$$

Dado un conjunto de entrenamiento $\{(A_i, B_i)\}_{i=1...n}$, la matriz de pesos M que se obtiene es:

$$M = \sum_{k=1}^{n} A_k^T B_k. \tag{1.35}$$

Si el conjunto de patrones está libre de ruido, la señal propagada de la capa F_A a la F_B a partir del patrón A_i es:

$$A_i M = \sum_{k=1}^{n} A_i A_k^T B_k = A_i A_i^T B_i + \underbrace{\sum_{k \neq i}^{n} A_i A_k^T B_k}_{0} = A_i A_i^T B_i = s B_i. \tag{1.36}$$

donde s es un número entero positivo.

En el proceso de recuperación, el estado de activación en la capa F_B al presentar en la capa F_A el patrón A_i es:

$$\phi\left(A_i M\right) = \phi\left(A_i A_i^T B_i\right) = \phi\left(s B_i\right) = B_i. \tag{1.37}$$

Del mismo modo, la señal propagada desde F_B hasta F_A es:

$$B_i M^T = \sum_{k=1}^{n} B_i B_k^T A_k = B_i B_i^T A_i + \underbrace{\sum_{k \neq i}^{n} B_i B_k^T A_k}_{0} = B_i B_i^T A_i = t A_i. \tag{1.38}$$

donde t es un número entero positivo.

El estado de activación en la capa F_A al presentar en la capa F_B el patrón B_i es:

$$\phi\left(B_i M^T\right) = \phi\left(B_i B_i^T A_i\right) = \phi\left(t A_i\right) = A_i. \tag{1.39}$$

Como se puede observar, con un conjunto de patrones libre de ruido se consigue que los pares de patrones formen mínimos energéticos locales y, por tanto, todos pueden ser recuperados.

Sea $\{D_i\}_{i=1}^n$, $D_i = (d_{i1}, \ldots, d_{im})$, un conjunto de elementos libre de ruido.

Para el conjunto de pares de entrenamiento $\{(A_i, B_i)\}_{i=1}^n$ construimos un nuevo conjunto de entrenamiento:

$$\left\{ \left(\left[A_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K} \right], \left[B_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K'} \right] \right) \right\}_{i=1.}^{n} \tag{1.40}$$

La matriz de pesos resultante es:

$$M = \sum_{i=1}^{n} \left[A_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K} \right]^T \left[B_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K'} \right]. \tag{1.41}$$

Al presentar en la capa F_A el patrón $\left[A_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K} \right]$ la señal

recibida por la capa F_B es:

$$\left[A_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K}\right] M = \left(A_i A_i^T + K D_i D_i^T\right) \left[B_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K'}\right] + \\ + \sum_{j \neq i} \left(A_i A_j^T\right) \left[B_j \mid \underbrace{D_j \mid \ldots \mid D_j}_{K'}\right]. \tag{1.42}$$

Haciendo una selección adecuada de D_i y de K se puede conseguir que se cumpla la siguiente condición:

$$K D_i D_i^T > \sum_{j \neq i} \left| A_i A_j^T \right|. \tag{1.43}$$

Con esto nos aseguramos que el patrón obtenido en la capa F_B sea el correcto, es decir:

$$\phi\left(\left[A_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K}\right] M\right) = \left[B_i \mid \underbrace{D_i \mid \ldots \mid D_i}_{K'}\right]. \tag{1.44}$$

Se puede realizar un razonamiento similar para obtener el valor adecuado de K'.

Entrenamiento múltiple

El entrenamiento múltiple fue propuesto por Wang, Cruz y Mulligan [46]. Consiste en realizar el proceso de aprendizaje varias veces con cada uno de los patrones con la intención de lograr que todos los patrones almacenados formen mínimos energéticos locales.

Cuantas más veces se repita el entrenamiento de un patrón menor será la energía asociada al mismo y, por tanto, mayor es la probabilidad de que forme un mínimo energético local (sólo se recuperan los patrones que forman mínimos energéticos locales).

Por otro lado, el entrenamiento de un patrón afecta al resto de patrones ya almacenados por lo que hay que encontrar el número

adecuado de veces que hay que entrenar con cada patrón para que todos puedan ser recuperados.

Para el cálculo del número de veces que se entrena cada uno de los patrones se proponen los dos métodos expuestos a continuación.

Entrenamiento múltiple secuencial El entrenamiento múltiple secuencial consiste en la repetición de la memorización de determinados patrones hasta conseguir que se almacenen correctamente patrones que no eran memorizados bien por la memoria. Como no se puede asegurar que el proceso termine, se hace necesario el establecer un límite del tiempo máximo que esperamos para la memorización de los patrones. El proceso se puede sintetizar en el esquema mostrado en la figura 1.7.

Este método tiene como ventaja su simplicidad mientras que entre sus inconvenientes destaca el hecho de que, si se alcanza el límite de tiempo establecido, no se puede extraer ninguna conclusión acerca de la posibilidad de memorizar todos los patrones (nada nos asegura que poco segundos después de la parada no se habrían almacenado todos).

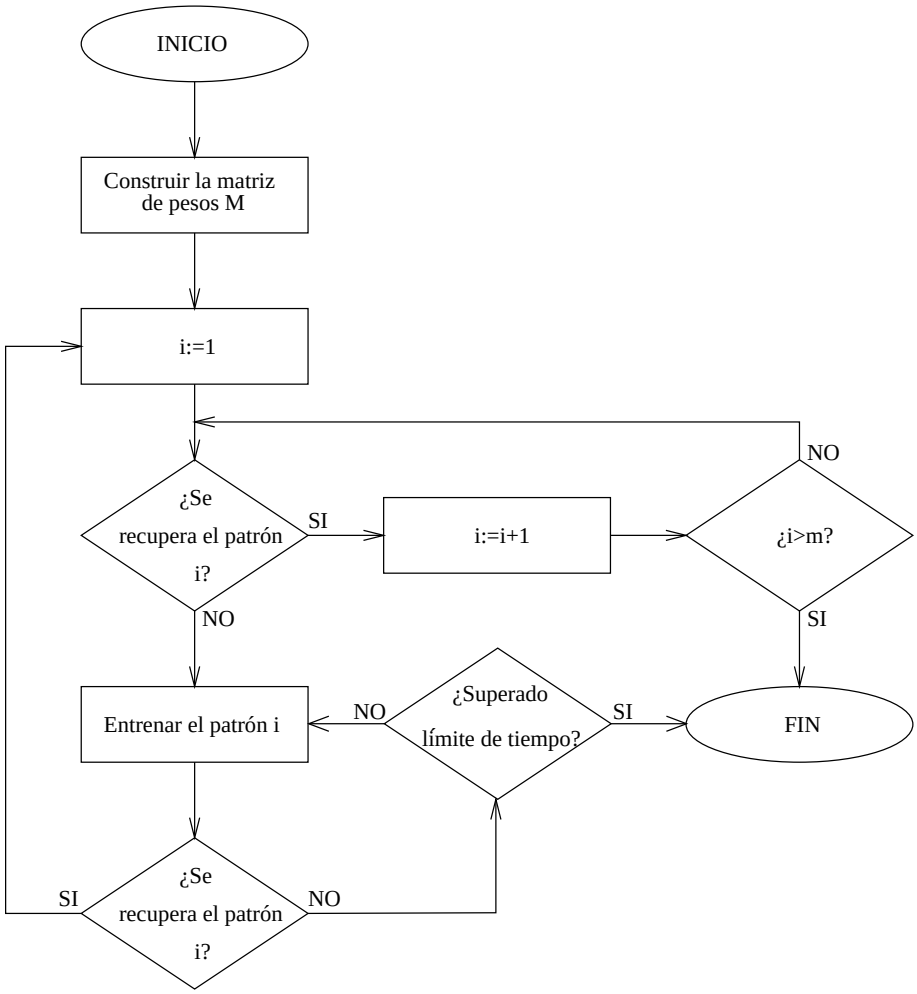
Entrenamiento múltiple por programación lineal Este método es una generalización del entrenamiento múltiple. Consiste en establecer una matriz de pesos generalizada en la que se añade un factor que se podría interpretar como el número de veces que se realiza el entrenamiento con cada uno de los distintos patrones:

$$M = \sum_{k=1}^{m} q_k A_k^T B_k. \tag{1.45}$$

Para que un patrón pueda recuperarse sin problema, éste debe constituir un mínimo energético local, es decir, cualquier patrón vecino (distancia de Hamming igual a uno) debe tener mayor energía.

Los autores desarrollan la idea de los mínimos energéticos locales para dar lugar a un complejo sistema de inecuaciones cuya

Figura 1.7: Entrenamiento múltiple secuencial



solución nos proporciona los q_i correspondientes a cada uno de los patrones a memorizar. Si el sistema no tiene solución es que el conjunto de patrones no puede ser memorizado.

La ventaja evidente del método la constituye el hecho de que, tras resolver el sistema de inecuaciones, podemos saber si los pa-

trones pueden ser memorizados y los pesos asignados a cada uno de los patrones en caso de que así sea.

Los inconvenientes superan con creces a las ventajas ya que:

- El sistema de inecuaciones tiene un número de incógnitas y de inecuaciones cuyo número crece con el tamaño de los patrones y con el número de patrones a memorizar de modo que un pequeño conjunto de patrones de ejemplo da lugar a un complejo sistema muy difícil de resolver.
- La ganancia en capacidad es mínima.
- Sólo determinados tipos de patrones dan resultados algo más aceptables. Entre esos tipos de patrones se encuentran los que tienen un número semejante de componentes positivas y negativas. De nuevo nos encontramos con el problema de que la memoria no es capaz de almacenar cualquier tipo de patrones que necesitemos.

1.2 Memorias asociativas difusas

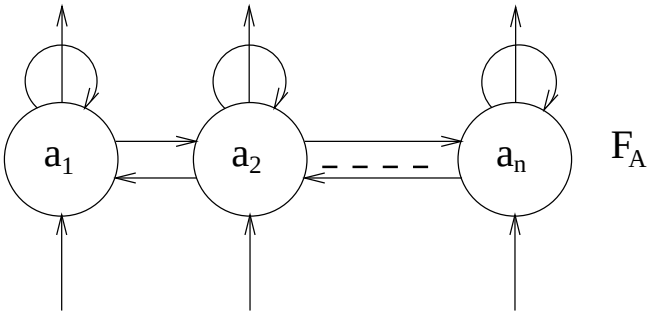
1.2.1 Mapas cognitivos difusos

Los mapas cognitivos difusos fueron introducidos por B.Kosko en 1985 [26, 27] inspirado por el trabajo de R.Axelrod de 1976 [5] sobre mapas cognitivos. Se trata de un modelo de una capa de EP que codifica patrones arbitrarios difusos de la forma $A_k = (a_{k1}, \ldots, a_{kn})\ k = 1, \ldots, m$. El entrenamiento se realiza usando aprendizaje hebbiano diferencial cuando es no supervisado o mediante la codificación directa de los pesos de las conexiones cuando se trata de aprendizaje supervisado.

La topología del modelo es la mostrada en la figura 1.8 en que los EP se corresponden con las componentes de los patrones.

Los EP del mapa cognitivo difuso representan conceptos relacionados entre sí por conexiones causa-efecto que indican la medida en que un concepto es o no causante de otro. Si el peso de la

Figura 1.8: Mapa Cognitivo Difuso



conexión desde el elemento S hasta el T es positivo indica que S es causante de se produzca T en un grado proporcional al peso de la conexión; si es peso es negativo indica que S es causante de que no se produzca T de forma proporcional al peso; si el peso de la conexión es cero indica que no existe relación causal entre los conceptos S y T.

1.2.1.1 Aprendizaje

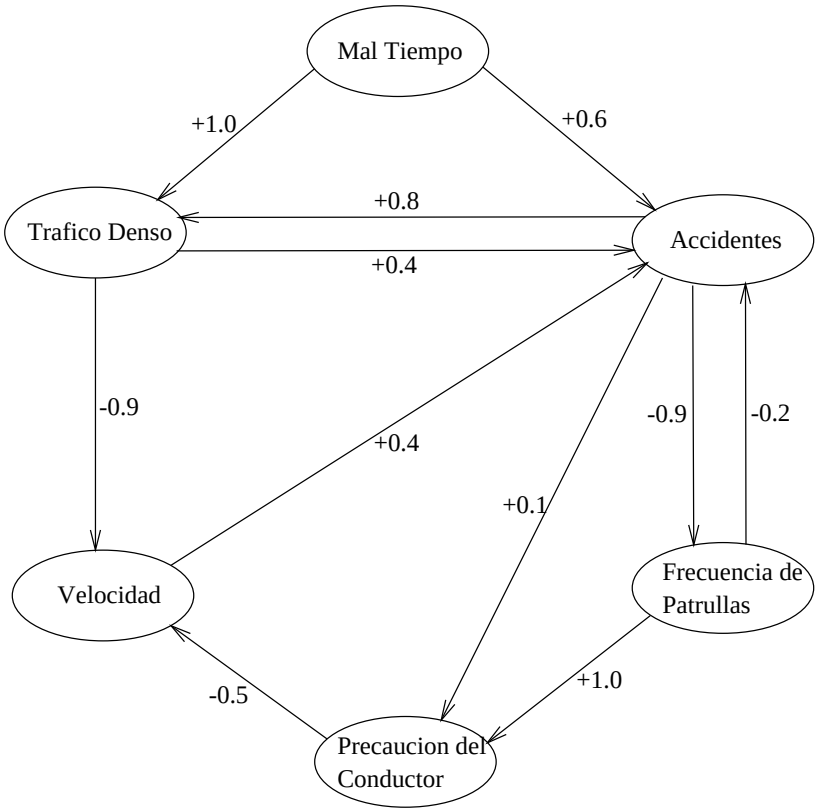
Como se ha comentado, el aprendizaje puede realizarse de forma supervisada o no supervisada. En el primero de los casos el ajuste de los pesos de las conexiones se realizará asignando directamente a las conexiones el peso causal correspondiente. Cuando se trata de aprendizaje no supervisado se emplea el aprendizaje hebbiano diferencial.

Aprendizaje supervisado

Cuando conocemos las relaciones causales que existen entre los conceptos, trabajamos con aprendizaje supervisado y codificamos directamente en los pesos de las conexiones las relaciones causales. Los pesos han de ser asignados de modo que se tenga en cuenta:

- Un peso positivo en la conexión $S \rightarrow T$ indica que un incre-

Figura 1.9: Ejemplo de Mapa Cognitivo Difuso



mento del valor de S causa un incremento en T. Del mismo modo un decremento en S causa un decremento en T.

- El peso de las conexiones positivas ha de estar en el rango $(0, 1]$, siendo 1 el peso correspondiente al mayor grado en que se incrementa T ante un incremento de S.
- Un peso negativo en la conexión $S \rightarrow T$ indica que un incremento del valor de S causa un decremento en T. Igualmente, un decremento de S causa un incremento en T.
- El peso de las conexiones negativas ha de estar en el rango

$[-1, 0)$, siendo -1 el peso correspondiente al mayor grado en que se decrementa T ante un incremento de S.

El conocimiento que se tiene de las relaciones causales puede provenir de más de un experto. En este caso los mapas cognitivos creados por cada uno de ellos pueden ser combinados para dar lugar a uno que los unifica. Para que se pueda producir dicha combinación es necesario que todos los mapas contengan los mismos conceptos de modo que a cada uno de ellos se le añadirán aquellos conceptos que le falten con unas relaciones causales nulas con los demás. Una vez que se tienen todos los mapas con los mismos conceptos representados se pueden sumar (y normalizar) las matrices de pesos para obtener un mapa cognitivo difuso que codifique el conocimiento aportado por todos los expertos.

Es también posible que el grado de fiabilidad de los expertos no sea el mismo para todos, de modo que la agregación del conocimiento representado en los mapas debe ser ponderada por el grado de fiabilidad de cada experto.

Aprendizaje no supervisado

En ocasiones no se conocerán las relaciones causales existentes entre los distintos conceptos representados. Ante esta situación es necesario inferir dichas relaciones a partir de la información recibida.

Los pesos se calculan automáticamente usando aprendizaje hebbiano diferencial que los ajusta a partir de los cambios sufridos por los EP conectados. El peso de la conexión entre los elementos i y j se ajusta mediante la ecuación siguiente:

$$\frac{d}{dt}m_{ij} = -m_{ij} + \frac{d}{dt}S\left(a_{ki}\right)\frac{d}{dt}S\left(a_{kj}\right). \tag{1.46}$$

donde S es la función sigmoide.

Cuando el cambio sufrido por dos EP tiene el mismo signo (los dos se ven incrementados o decrementados) el peso de la conexión será incrementado de forma proporcional a dichos cambios. Del mismo modo, si los elementos tienen cambios opuestos el peso de la conexión se decrementará de forma proporcional.

1.2.1.2 Recuperación

El comportamiento del mapa cognitivo difuso en la fase de recuperación viene expresado por la siguiente ecuación:

$$\frac{d}{dt}a_i = -a_i + \sum_{j=1}^{n} S\left(a_j\right) m_{ji} + I_i. \tag{1.47}$$

I_i representa la i-ésima componente de un patrón de entrada inicial suministrado al sistema.

Otro modo en que se puede definir el comportamiento del sistema es mediante esta otra ecuación:

$$\frac{d}{dt}a_i = -\alpha_i\left(a_i\right)\left[\beta_i\left(a_i\right) - \sum_{j=1}^{p} m_{ji} S\left(a_j\right)\right]. \tag{1.48}$$

donde α_i es una función no decreciente, β_i es una función acotada arbitraria y S es una sigmoide.

1.2.1.3 Estabilidad

El mapa cognitivo difuso no presenta un comportamiento estable sino que por el contrario se ve afectado por un comportamiento oscilante en el que el sistema pasa cíclicamente por un conjunto de estados. Esto es un serio problema ya que no es fácil obtener conclusiones claras a partir de estos comportamientos oscilatorios.

1.2.1.4 Ventajas e inconvenientes

La principal ventaja que presentan los mapas cognitivos difusos es que las relaciones almacenadas son fácilmente comprensibles, alejándose de la característica de 'caja negra'asociada a las redes neuronales. También hay que destacar la capacidad de adaptarse sobre la marcha ajustando los pesos ante nuevos patrones de entrenamiento.

El claro inconveniente que presentan es el de la falta de estabilidad que provoca la aparición de ciclos de estados que son visitados de forma repetida y de los cuales no es fácil obtener conclusiones.

1.2.2 Memoria asociativa difusa

La memoria asociativa difusa introducida por B.Kosko en 1987 [30] es un modelo heteroasociativo formado por dos capas de EP que almacena parejas de patrones espaciales arbitrarios de la forma (A, B) donde $A = (a_1, \ldots, a_n)$ y $B = (b_1, \ldots, b_p)$. La FAM opera en tiempo discreto sin retroalimentación y el entrenamiento se realiza *offline* usando aprendizaje hebbiano difuso.

EL empleo del término "difuso" en la denominación de esta memoria se debe a la capacidad que posee para trabajar con información continua y al empleo del aprendiza hebbiano difuso. Este aprendizaje surge como una modificación del aprendizaje hebbiano clásico al sustituir la suma por la t-norma del máximo y el producto por la t-conorma del mínimo.

La topología de la memoria asociativa difusa es la mostrada en la figura 1.10 en la que los EP de la capa F_A corresponden a las componentes del patrón A y los EP de la capa F_B corresponden a las componentes del patrón B..

1.2.2.1 Aprendizaje

El aprendizaje hebbiano difuso usado para el almacenamiento en esta memoria, obtiene los pesos de las conexiones entre los EP de las dos capas mediante $m_{ij} = \min(a_i, b_j)$, donde m_{ij} es el peso de la conexión existente entre los EP a_i y b_j.

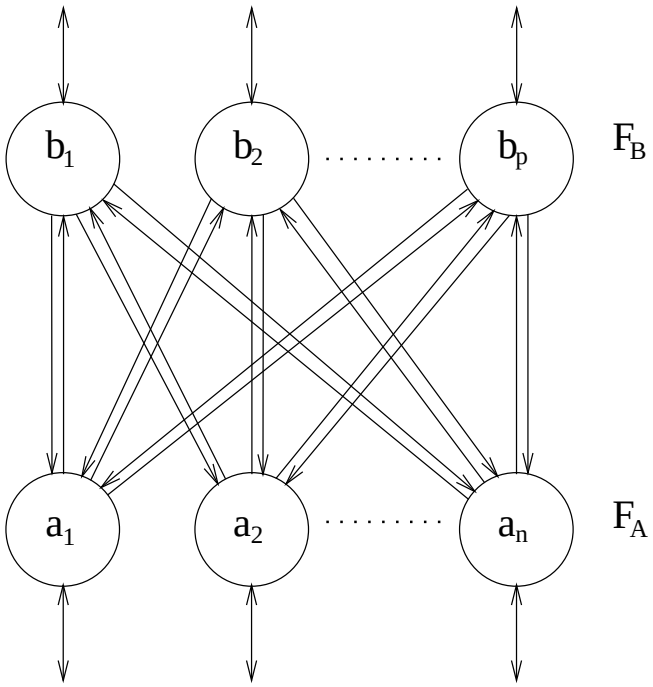
En notación matricial, la matriz de pesos M se obtiene mediante el producto exterior difuso $M = A^T \circ B$.

Este tipo de aprendizaje sólo permite el almacenamiento de una asociación de patrones.

1.2.2.2 Recuperación

En el proceso de recuperación, aunque las ecuaciones son bidireccionales, no hay retroalimentación, es decir, la información circula en un único sentido.

Figura 1.10: Memoria Asociativa Difusa (FAM)



En la capa F_B los EP recuperan la información mediante:

$$b_j = \max_{i=1}^{n} \left[\min \left(a_i, m_{ij} \right) \right]. \tag{1.49}$$

De un modo similar, la recuperación en la capa F_A se realiza mediante:

$$a_i = \max_{j=1}^{p} \left[\min \left(b_j, m_{ji} \right) \right]. \tag{1.50}$$

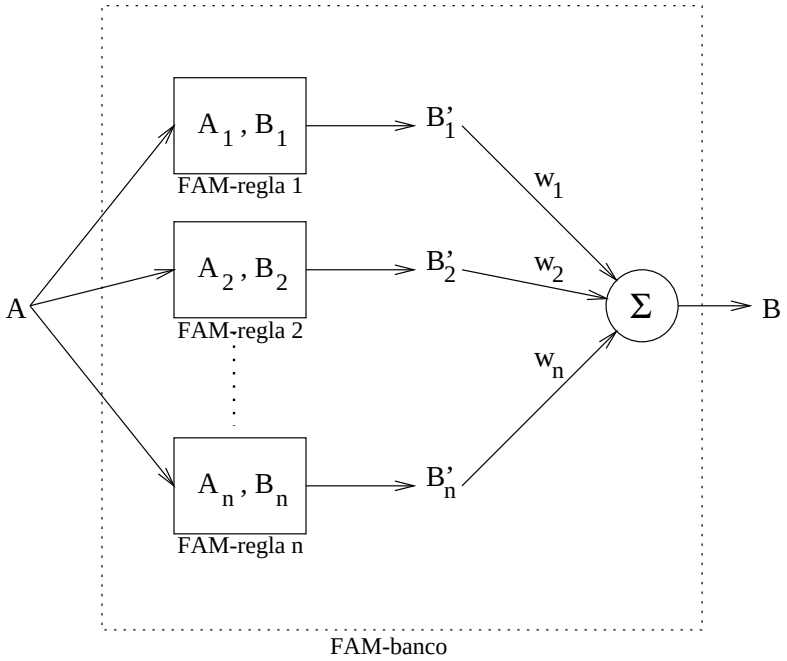
Este modo de recuperación garantiza que todos los patrones de entrada recuperan patrones de salida que están contenidos en las asociaciones almacenadas. Es decir, si se ha almacenado la asociación (A, B), a partir de un patrón de entrada en la capa F_A se recupera en la capa F_B el patrón $B' \subseteq B$; del mismo modo, a

partir de un patrón de entrada en la capa F_B se recupera en la capa F_A el patrón $A' \subseteq A$.

1.2.2.3 Análisis

Por el modo en que realiza el aprendizaje, la FAM es capaz de almacenar sólo una asociación de patrones. A dichos patrones asociados Kosko los denomina FAM-regla, donde el par de patrones (A, B) representa la regla $A \rightarrow B$. La agrupación de un conjunto de FAM-reglas se denomina FAM-banco, en el cual se accede a las reglas en paralelo sin que exista interferencia entre ellas para posteriormente combinar los resultados individuales obtenidos.

Figura 1.11: FAM-banco



1.2.2.4 Ventajas e inconvenientes

La principal desventaja de la memoria asociativa difusa es su incapacidad para almacenar más de una asociación. La solución propuesta por Kosko al introducir el concepto de FAM-banco dota al sistema de mayor flexibilidad ya que puede adaptarse al número de asociaciones que sean necesarias. No obstante, esto implica un consumo de memoria elevado, sobre todo cuando se intenta representar un sistema complejo.

A pesar de los inconvenientes indicados, la FAM es un modelo importante por su capacidad para trabajar con información difusa que nos permite almacenar conjuntos de reglas difusas.

1.2.3 FAM con entrada y salida binaria

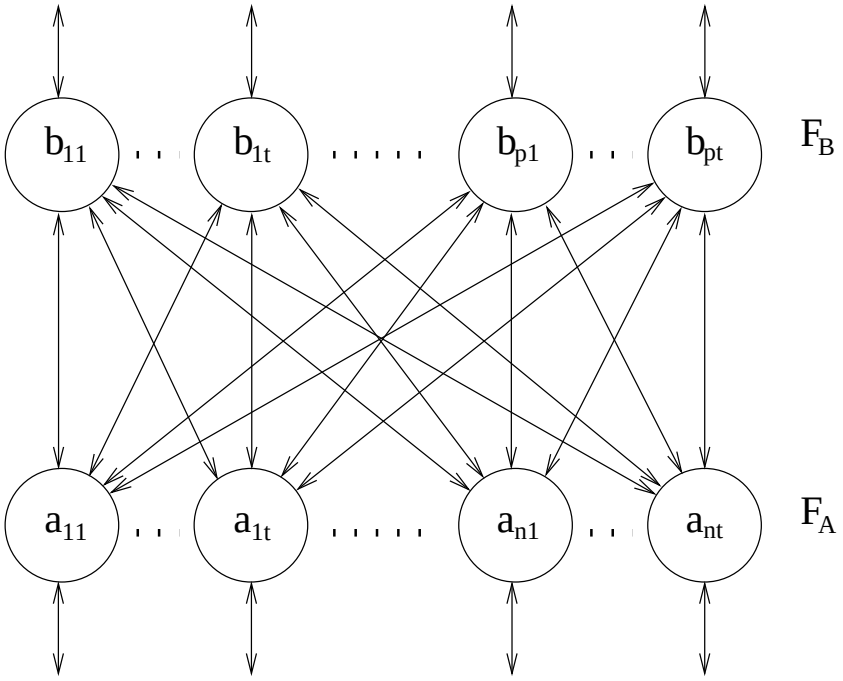
La memoria asociativa difusa con entrada y salida binaria (BIOFAM) introducida por B.Kosko en 1992 [33] es una adaptación de la FAM para el proceso de información expresada en forma discreta.. El modelo difiere de la FAM principalmente en el proceso de codificación.

La BIOFAM es un modelo heteroasociativo formado por dos capas de EP que almacena parejas de patrones espaciales arbitrarios representándolos mediante una codificación binaria. Los patrones espaciales son de la forma (A, B) donde $A = (a_1, \ldots, a_n)$ y $B = (b_1, \ldots, b_p)$. La codificación binaria de cada una de las componentes de los patrones espaciales da lugar a una representación binaria de los mismos como $A = (a_{11}, \ldots, a_{1t}, \ldots, a_{n1}, \ldots, a_{nt})$ y $B = (b_{11}, \ldots, b_{1t}, \ldots, b_{p1}, \ldots, b_{pt})$, donde n y p son el número de componentes de los patrones A y B respectivamente y t es la precisión con la que se trabajará. La BIOFAM opera en tiempo discreto sin retroalimentación y el entrenamiento se realiza *offline*.

La topología de la BIOFAM se muestra en la figura 1.12.

Los EP de la capa F_A corresponden a las componentes del patrón A y los de la capa F_B corresponden a las componentes del patrón B.

Figura 1.12: FAM con Entrada y Salida Binaria (BIOFAM)



1.2.3.1 Codificación

Mientras que la FAM codifica de forma continua en el intervalo $[0, 1]$ cada una de las componentes del patrón a memorizar, la BIOFAM las codifica en binario. Para ello sigue un método de codificación muy simple en el que expresa la información continua mediante un vector binario cuyo tamaño depende de la precisión deseada para la representación.

Para realizar la codificación, cada componente se traslada al vector binario en el que activará el bit correspondiente al grado de pertenencia a representar teniendo en cuenta que sólo puede haber un bit activo.

1.2.3.2 Aprendizaje

El aprendizaje hebbiano usado para el almacenamiento en esta memoria, obtiene los pesos de las conexiones entre los elementos de proceso de las dos capas mediante $m_{ij} = a_i b_j$, donde m_{ij} es el peso de la conexión existente entre los EP a_i y b_j.

En notación matricial, la matriz de pesos M se obtiene mediante el producto exterior $M = A^T B$.

1.2.3.3 Recuperación

En el proceso de recuperación, aunque las ecuaciones son bidireccionales, no hay retroalimentación, es decir, la información circula en un único sentido.

En la capa F_B los EP recuperan la información mediante:

$$b_j = \max_{i=1}^{n} \left[a_i m_{ij} \right]. \tag{1.51}$$

De un modo similar, la recuperación en la capa F_A se realiza mediante:

$$a_i = \max_{j=1}^{p} \left[b_j m_{ji} \right]. \tag{1.52}$$

Este modo de recuperación garantiza que todos los patrones de entrada recuperan patrones de salida que están contenidos en las asociaciones almacenadas. Es decir, si se ha almacenado la asociación (A, B), a partir de un patrón de entrada en la capa F_A se recupera en la capa F_B el patrón $B' \subseteq B$; del mismo modo, a partir de un patrón de entrada en la capa F_B se recupera en la capa F_A el patrón $A' \subseteq A$.

1.2.3.4 Ventajas e inconvenientes

La principal desventaja de la BIOFAM, al igual que ocurre con la FAM, es su incapacidad para almacenar más de una asociación de tal modo que es necesario recurrir a los BIOFAM-bancos para obtener una solución similar a la obtenida con la FAM. Esto lleva a que se dispare el consumo de memoria para el almacenamiento de la

información, situación que se agrava conforme vamos aumentando la precisión con la que se codificarán las componentes espaciales.

Como principal ventaja tiene su mayor facilidad de trabajo al tratarse de un modelo discreto. Al trabajar con información discreta no se sufren errores de cálculo derivados de los redondeos de números reales. Sin embargo es muy alta la cantidad de memoria necesaria para obtener con la codificación binaria una precisión similar a aquella en la que se producen los redondeos en la representación continua.

1.3 Notas finales

En este capítulo hemos repasado algunos modelos de memorias asociativas discretas y difusas. La baja capacidad de almacenamiento mostrada por todas ellas es su principal inconveniente y nos ha motivado para buscar una nueva memoria asociativa con una capacidad de almacenamiento que la haga realmente útil.

Con el objetivo de construir una memoria asociativa que pueda almacenar información expresada de forma imprecisa, en el siguiente capítulo vamos a repasar un elemento muy importante en el razonamiento aproximado para la representación de información imprecisa: las variables lingüísticas.

Además se muestra un método de codificación de variables lingüísticas que permite representar información imprecisa de forma discreta. Este método, llamado *Codificación por Discretización Incremental,* nos permite adaptar memorias asociativas discretas para el almacenamiento de información imprecisa.

Capítulo 2

Variables Lingüísticas y su Codificación

2.1 Variables lingüísticas

2.1.1 Introducción

La ciencia moderna ha tratado de alcanzar un conocimiento preciso de los fenómenos que nos rodean. Esa precisión implica la creación de modelos matemáticos que describan de forma detallada la aceleración de una manzana que se desprende de un árbol, el movimiento de un planeta que gira alrededor del Sol o la energía desprendida por un material radiactivo. Tras la aparición de la computadora como máquina que aporta gran potencia de cálculo, las expectativas depositadas en los sistemas de resolución de problemas se disparan hasta el punto de intentar emular el comportamiento del cerebro humano.

Una computadora es una máquina diseñada para trabajar con valores numéricos precisos mientras que el hombre usa en sus razonamientos conceptos vagos e imprecisos. Sin embargo, la imprecisión de nuestros sentidos no nos impide realizar tareas complejas como el caminar. Cuando caminamos no pensamos en la localización de nuestro centro de gravedad ni en la velocidad y posición de nuestro cuerpo ni en el impulso con que debemos dar el siguiente paso para no caer. Simplemente caminamos. Pero si lo que quere-

mos es hacer caminar de forma fluida a un robot, nos topamos con la complejidad asociada a algo que a nosotros nos resulta sencillo.

A pesar de la imprecisión de la información, el hombre es capaz de realizar tareas muy complejas obteniendo resultados suficientemente buenos mientras que el empleo de una mayor precisión mediante computadoras da peores resultados frente a las mismas tareas. Es decir, las personas realizamos con éxito tareas muy complejas manejando información imprecisa. Sin embargo fracasamos cuando tratamos de obtener un método preciso para realizarlas. Esto pone de manifiesto el principio de Incompatibilidad de Zadeh que afirma [59]:

> La precisión con que podemos modelar un sistema guarda una relación inversa con la complejidad del mismo, es decir, no se pueden abordar tareas de complejidad elevada mediante el uso de conceptos muy precisos.

Cuando tratamos de analizar sistemas complejos como los relacionados con el pensamiento humano, la complejidad de los mismos hace inútil el empleo de métodos numéricos precisos. Por ello, se plantea la reducción de la precisión para adoptar enfoques de carácter aproximado que aporten buenas soluciones, aunque no sean las óptimas, a problemas complejos.

Si queremos emular el comportamiento del cerebro humano mediante una computadora necesitamos utilizar una representación de la información similar a la que emplea el ser humano, es decir, debemos expresarla mediante el uso de términos vagos e imprecisos. De este modo se simplifica además la comunicación con el usuario ya que ésta se realiza usando conceptos con los que está familiarizado. El estudio de sistemas que emulen la capacidad del hombre para procesar información vaga e imprecisa es el objetivo de una rama de la Inteligencia Artificial denominada Razonamiento Aproximado.

El objetivo del análisis de un sistema es el de obtener un modelo del mismo que nos permita su implantación en un sistema automático de procesamiento de información. Como las herramientas de que disponemos trabajan con información numérica y operaciones matemáticas, necesitamos un medio para expresar conceptos

aproximados que puedan ser procesados con dichas herramientas.

2.1.2 Conjuntos difusos

Para establecer un enfoque aproximado en el análisis de sistemas complejos necesitamos un medio para expresar información imprecisa. La base que nos servirá para expresar la imprecisión serán los conjuntos difusos [54].

Con frecuencia los objetos que nos rodean no pueden dividirse en categorías perfectamente diferenciadas. Está claro que a la clase de los animales pertenecen el perro, el elefante y el ratón. Sin embargo, cuando tratamos de clasificar ciertos microorganismos parásitos o los virus, su pertenencia a la clase de los animales es algo menos clara. Del mismo modo, la altura necesaria para pertenecer a la categoría de las personas altas, está definida de forma muy ambigua.

Este tipo de conjuntos definidos de forma poco precisa juegan un papel muy importante en el pensamiento humano debido a que la información percibida por nuestros sentidos es imprecisa. Un jugador de golf no sabe la velocidad y dirección exactas del viento ni puede aplicar a la pelota una fuerza muy precisa. Sin embargo es capaz de golpear con una fuerza tal que la pelota se quede cerca de la bandera. En su razonamiento ha usado conceptos como *distancia corta, viento moderado en dirección sur-sureste* y *fuerza alta* en vez de *10 metros hasta el hoyo, 7km/h de velocidad del viento con dirección 37 grados sur* y *fuerza a aplicar de 150N*.

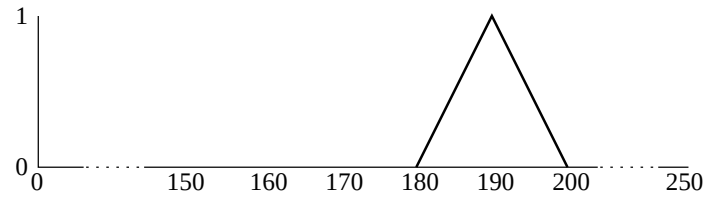
Los conjuntos difusos constituyen una herramienta que nos ayudará a trabajar con clases definidas de forma imprecisa. Cuando tratamos con conjuntos clásicos nos encontramos con dos tipos de elementos, los que pertenecen y los que no pertenecen al conjunto. En un conjunto difuso hay un número infinito de grados de pertenencia entre las situaciones de pertenencia y no pertenencia.

Definición 3. Sea X un espacio de objetos. Un *conjunto difuso* A en X se caracteriza por una función de pertenencia $\mu_A(x)$ que asocia a cada elemento x de X un valor real en el intervalo $[0,1]$ que representa el grado de pertenencia del elemento x al conjunto A.

En la función de pertenencia los grados 0 y 1 indican no pertenencia y pertenencia al conjunto respectivamente. Los valores intermedios representan aquellas situaciones en que la pertenencia o no de un elemento al conjunto no está definida de forma precisa. Con esta definición de conjunto difuso, los conjuntos clásicos son un caso particular de los difusos en que la función de pertenencia toma valores en el conjunto $\{0, 1\}$.

Ejemplo. Supongamos que el espacio sobre el que definiremos un conjunto difuso es el intervalo de medidas expresadas en centímetros $X = [0, 250]$. Podemos definir el conjunto difuso A de las alturas de las personas altas mediante la función de pertenencia μ_A representada en la figura 2.1.

Figura 2.1: Conjunto difuso asociado a la etiqueta ALTO



2.1.3 Variables lingüísticas

El empleo del lenguaje como soporte del razonamiento humano hace que la información que maneja esté expresada en términos lingüísticos que pueden ser vagos e imprecisos. Por ejemplo, la altura de una persona puede expresarse mediante términos como *muy baja, baj*a, *normal*, *alta* o *muy alta*. Pero, ¿cuáles son las alturas de las personas *altas*? ¿Con qué altura deja una persona de ser *normal* para pasar a ser *alta*?. Los conjuntos de las alturas pertenecientes a cada uno de los términos no tienen límites perfectamente definidos y por ello utilizaremos conjuntos difusos para dotarlos de significado. Cada término tendrá asociado un conjunto difuso con una función de pertenencia que indique en qué medida

pertenece una altura a dicho término. Esto conduce a la definición de lo que llamaremos variable lingüística.

Una variable lingüística [51, 52, 53] es aquella que no toma valores numéricos sino valores lingüísticos como pueden ser *alta*, *baja*, *normal*, *muy baja* o *muy alta*. Además, asociado a cada uno de estos términos hay un conjunto difuso que los dota de significado. Podemos definir más formalmente un variable lingüística del siguiente modo:

Definición 4. Una variable lingüística se caracteriza por la quíntupla (X, $T(X)$, U, G, M) donde:

- X es el nombre de la variable
- $T\,(X)$ es el conjunto de términos lingüísticos sobre el que puede tomar valor la variable X
- U es el universo de discurso sobre el que la variable X toma su valor cuantitativo
- G es una regla sintáctica que genera los términos de $T\,(X)$
- M es una regla semántica que asocia a cada término de $T\,(X)$ su significado mediante un conjunto difuso definido en U

Por ejemplo, una variable lingüística asociada a la altura de una persona, estaría definida por:

- $X =$ ALTURA
- $T\,(\text{ALTURA}) = \{\text{muy bajo}, \text{bajo}, \text{normal}, \text{alto}, \text{muy alto}\}$
- $U = [0, 250]$ cm
- $G =$ normal | [muy] (bajo | alto)
- M (ver figuras 2.2, 2.3, 2.4, 2.5 y 2.6)

Los conjuntos difusos utilizados como modelos para las variables lingüísticas satisfacen las necesidades de flexibilidad que esperamos para la representación del conocimiento, es decir, ayudan

Figura 2.2: M(muy bajo)

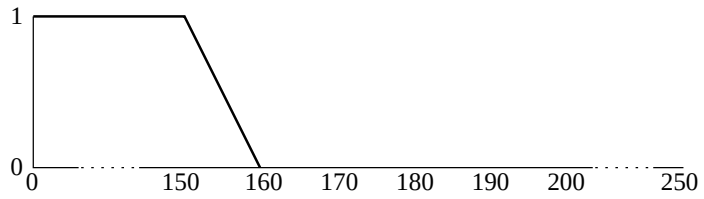
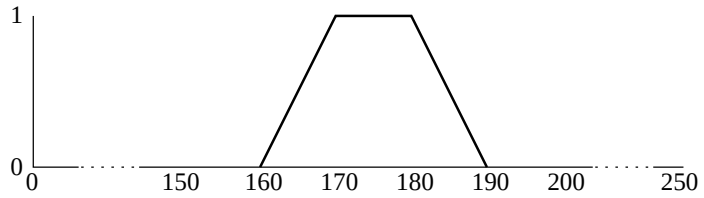


Figura 2.3: M(bajo)

Figura 2.4: M(normal)



a dar un significado al valor impreciso de una variable lingüística. De este modo el proceso de razonamiento a partir de términos lingüísticos se desarrollará mediante el empleo de los conjuntos difusos.

Figura 2.5: M(alto)

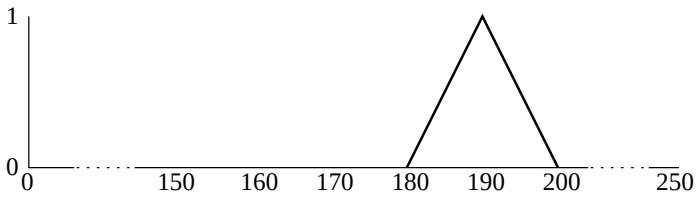
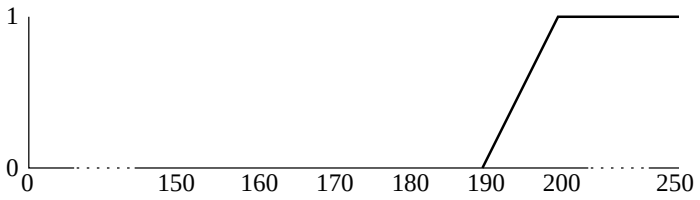


Figura 2.6: M(muy alto)



2.2 Codificación por discretización incremental

2.2.1 Introducción

Los sistemas automáticos de tratamiento de datos suelen requerir un alto grado de precisión. Puesto que los sentidos humanos perciben el entorno de forma imprecisa y ya que de forma natural razonamos en términos imprecisos, es muy poco natural comunicarse de forma altamente precisa con un sistema automático de tratamiento de datos.

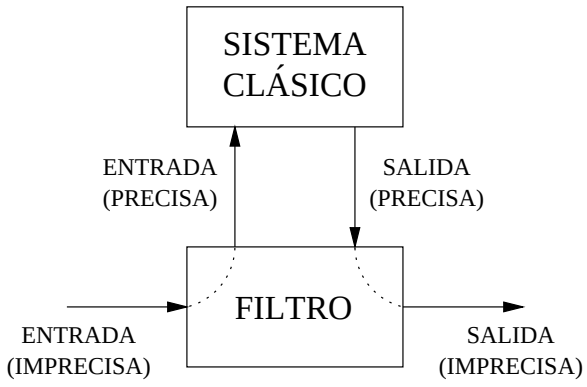
Es mejor diseñar sistemas que sean capaces de trabajar con información expresada tal y como el usuario la proporciona, mediante el uso de conceptos vagos e imprecisos. Para afrontar este problema tenemos dos opciones:

- Diseñar nuevos sistemas de tratamiento de datos que sean capaces de trabajar con información expresada de forma imprecisa abandonando los sistemas tradicionales.

- Diseñar un método que, a modo de filtro, admita información expresada de forma imprecisa y la convierta a una representación precisa que se ajuste a las necesidades de un sistema tradicional. De este modo se pueden reutilizar todos los sistemas existentes sin la necesidad de diseñar y estudiar otros nuevos.

El método de Codificación por Discretización Incremental (CDI) introducido por W.Fajardo en 1995 [15] permite tanto codificar la información imprecisa suministrada por el usuario de modo que pueda ser usada por un sistema clásico, como descodificar la salida precisa del sistema para presentar los resultados en términos más comprensibles para el usuario. Este filtro actúa de modo independiente al sistema de tratamiento de información de manera que puede "acoplarse" a cualquier sistema clásico.

Figura 2.7: Filtro para trabajar con información imprecisa en un modelo clásico



2.2.2 Codificación de variables lingüísticas mediante discretización incremental

Sea H una variable lingüística descrita por la quíntupla (H, $T(H)$, V, G, M).

Basándonos en una forma de representación de la información difusa altamente aceptada y muy extendida en el razonamiento en presencia de datos difusos, cualquier elemento u de V se representa en términos de $T(H)$ como:

$$u \equiv \alpha_1 t_1, \ldots, \alpha_n t_n; \qquad \alpha_i \in [0,1],\ t_i \in T(H). \tag{2.1}$$

siendo α_i la compatibilidad de u con t_i determinada según alguna medida de incertidumbre. Es obvio que $(\alpha_1, \ldots, \alpha_n)$ es la única representación válida para u bajo el convenio de emplear como único elemento de base el conjunto $T\,(H)$ de términos de la variable.

Supongamos que $T(H)$ se compone de n elementos. Dotamos a $T(H)$ de un orden arbitrario y a cada término de $T(H)$ se asocia un vector de dimensión m (supuesto que se desee una precisión global de orden $1/m$). A la totalidad de $T(H)$ se asocia un vector de dimensión $m \times n$.

Entonces u se codifica como un vector binario (o bipolar) de dimensión $m \times n$:

$$C(u) = (C_{11}, \ldots, C_{1m}, C_{21}, \ldots, C_{2m}, \ldots, C_{n1}, \ldots, C_{nm}). \tag{2.2}$$

donde:

$$\begin{cases} \alpha_i = 0 \ \rightarrow \ C_{ij} = 0;\ i = 1, \ldots, n;\ j = 1, \ldots, m \\ \alpha_i \neq 0 \ \rightarrow \begin{cases} \exists j \text{ t.q. } \frac{j}{m} \leq \alpha_i < \frac{j+1}{m} \\ C_{il} = \begin{cases} 1 \text{ si } l \leq j \\ 0 \text{ si } l > j. \end{cases} \end{cases} \end{cases} \tag{2.3}$$

o lo que es lo mismo, $C(u)$ es un vector cuyas componentes se calculan siguiendo el algoritmo expresado en el siguiente esquema:

1. – Desde i $= 1$ hasta n
   1.1. – Desde j $= 1$ hasta m
      1.1.1. – Si j $\leq$ Parte entera de $(\alpha_i \circ m)$
         1.1.1.1. – Entonces $C_{ij} = 1$
      1.1.2. – Si j $>$ Parte entera de $(\alpha_i \circ m)$
         1.1.2.1. – Entonces $C_{ij} = 0$
   1.2. – Fin
2. – Fin

De este modo tenemos un sistema de codificación capaz de expresar cualquier elemento del universo de discurso V en términos del conjunto $T(H)$.

A partir de estas ideas podemos extender nuestra codificación al caso en que el valor de la variable venga dado por el valor de un subconjunto difuso A sobre V. Basta seguir considerando α_i como el grado de compatibilidad de A con t_i medido adecuadamente.

En particular si $A = t_k$ entonces se codifica con:

$$C_{ij} = \begin{cases} 1 \text{ si } i = k;\ \forall j \\ 0 \text{ en otro caso .} \end{cases} \tag{2.4}$$

En este caso es redundante utilizar un número elevado de bits por etiqueta, por lo que es mejor utilizar un único bit $(m = 1)$ por término $t_i \in T(H)$.

Al fijar el número de bits utilizados en la codificación estamos determinando la medida de la granularidad con la que vamos a trabajar. Por ello hemos de prever desde el principio la precisión máxima con la que pretendemos trabajar.

2.2.2.1 Codificación de información expresada en términos de compatibilidad

Los valores del universo de discurso V son expresados por expertos como grados de compatibilidad α_i con los elementos del conjunto de términos $T(H)$. Entonces la codificación de la variable lingüísti-

ca, expresada con una precisión $1/m$ será:

$$C(u) = (C_{11}, \ldots, C_{1m}, C_{21}, \ldots, C_{2m}, \ldots, C_{n1}, \ldots, C_{nm}). \quad (2.5)$$

donde $C_{i1}, \ldots, C_{im}$ es la codificación de la compatibilidad α_i con el término t_i.

2.2.2.2 Codificación de información expresada en términos crisp

Los valores del universo de discurso V se obtienen mediante mecanismos de medición que proporcionan información crisp. Para calcular los grados de compatibilidad de la información crisp con el conjunto de términos se usa la representación semántica de la variable lingüística expresada mediante funciones de pertenencia de los conjuntos difusos asociados.

La codificación del valor crisp u de la variable lingüística, con precisión $1/m$ será:

$$C(u) = (C_{11}, \ldots, C_{1m}, C_{21}, \ldots, C_{2m}, \ldots, C_{n1}, \ldots, C_{nm}). \quad (2.6)$$

donde $C_{i1}, \ldots, C_{im}$ es la codificación de la compatibilidad $\alpha_i = \mu_i(u)$ con el término t_i.

2.2.2.3 Codificación de información expresada en términos lingüísticos

Los valores del universo de discurso V vienen expresados como subconjuntos difusos de V. En este caso hay que considerar α_i como el grado de compatibilidad del subconjunto difuso u de V con el término t_i.

La codificación del subconjunto difuso u con precisión $1/m$ será:

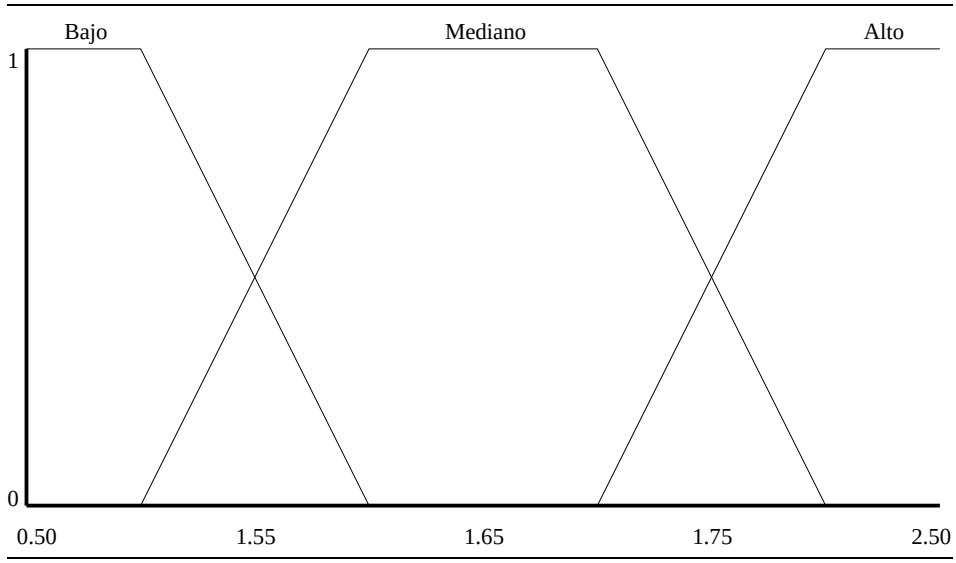
$$C(u) = (C_{11}, \ldots, C_{1m}, C_{21}, \ldots, C_{2m}, \ldots, C_{n1}, \ldots, C_{nm}). \quad (2.7)$$

donde $C_{i1}, \ldots, C_{im}$ es la codificación de la compatibilidad α_i de u con el término t_i.

2.2.3 Ejemplo

Supongamos que queremos codificar la variable lingüística altura y que el conjunto de términos sobre el que puede tomar sus valores la variable es {bajo, mediano, alto}, que se representan semánticamente en el universo de discurso $[0.50, 2.50]$ mediante los conjuntos difusos mostrados en la Fig. 2.8.

Figura 2.8: Conjuntos difusos asociados a la variable ALTURA



A continuación se muestran los tres casos anteriormente comentados que pueden presentarse para la codificación de información según el modo en que venga ésta expresada.

2.2.3.1 Codificación de información expresada en términos de compatibilidad

Supongamos que la información referente a la altura de un individuo es expresada por expertos en función del conjunto de términos

lingüísticos que puede tomar la variable altura. Es decir, se indica un grado de compatibilidad de la altura con cada una de las etiquetas lingüísticas bajo, mediano y alto.

Si la información suministrada h es 0.2 bajo, 0.8 mediano y queremos trabajar con una precisión global de orden $1/10$, la codificación de dicha información será:

$$\begin{aligned} C(h) &= (C_{bajo\,1}, \ldots, C_{bajo\,10}, C_{med\,1}, \ldots, C_{med\,10}, C_{alto\,1}, \ldots, C_{alto\,10}) \\ &= (1100000000\ 1111111100\ 0000000000)\,. \end{aligned} \tag{2.8}$$

2.2.3.2 Codificación de información crisp

Supongamos ahora que la altura del individuo ha sido medida usando mecanismos que proporcionan información crisp. En este caso tenemos que calcular los grados de compatibilidad de la medida obtenida con cada uno de los conjuntos difusos que representan la semántica de los términos lingüísticos de la variable altura.

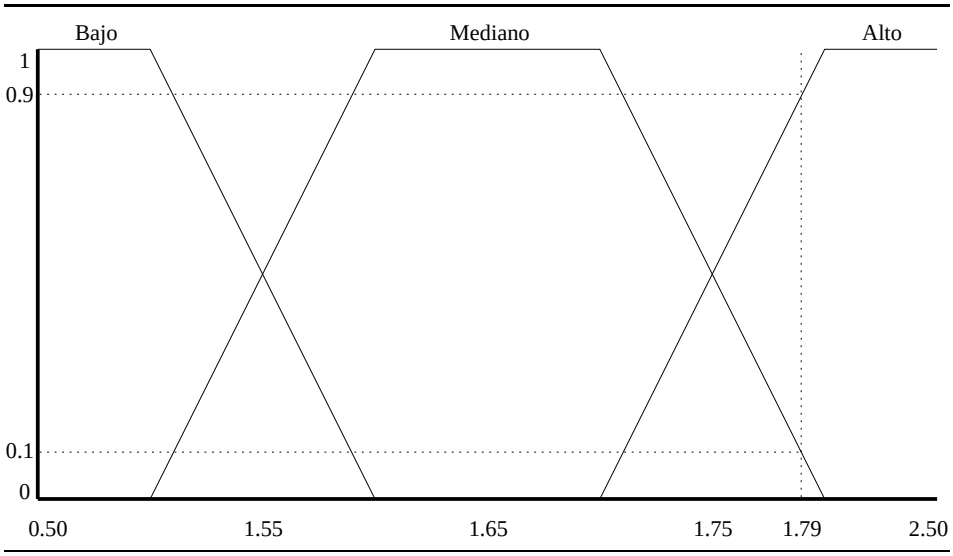
Si la medición indica un valor para la altura de 1.79, las compatibilidades obtenidas con los conjuntos difusos asociados son (ver Fig. 2.9):

$$\begin{aligned} \alpha_{bajo} &= \mu_{bajo}(1.79) = 0.0 \\ \alpha_{mediano} &= \mu_{med}(1.79) = 0.9 \\ \alpha_{alto} &= \mu_{alto}(1.79) = 0.1 \end{aligned} \tag{2.9}$$

Ahora tenemos la información expresada en grados de compatibilidad *0.9 mediano, 0.1 alto*. Por tanto, para codificar la información de la variable lingüística altura con una precisión global de $1/10$ procedemos como en el caso anterior.

$$\begin{aligned} C(h) &= (C_{bajo\,1}, \ldots, C_{bajo\,10}, C_{med\,1}, \ldots, C_{med\,10}, C_{alto\,1}, \ldots, C_{alto\,10}) \\ &= (0000000000\ 1111111110\ 1000000000)\,. \end{aligned} \tag{2.10}$$

Figura 2.9: Información crisp 1.79



2.2.3.3 Codificación de información lingüística

Es posible que la información suministrada acerca de la altura venga expresada como un subconjunto difuso sobre V. En este caso basta con seguir considerando α_i como el grado de compatibilidad con los términos lingüísticos medido adecuadamente.

Si la información aportada es h="alrededor de 165" y queremos realizar una codificación con una precisión de 10 bits por término el vector resultante será:

$$\begin{aligned} C(h) &= (C_{bajo\,1}, \ldots, C_{bajo\,10}, C_{med\,1}, \ldots, C_{med\,10}, C_{alto\,1}, \ldots, C_{alto\,10}) \\ &= (0000000000\ 1111111111\ 0000000000)\,. \end{aligned} \tag{2.11}$$

En este caso hubiese sido más natural emplear sólo un bit por término de modo que la codificación resultante sería:

$$\begin{aligned} C(h) &= (C_{bajo}, C_{med}, C_{alto}) \\ &= (0\ 1\ 0)\,. \end{aligned} \tag{2.12}$$

2.2.4 Decodificación de resultados codificados mediante discretización incremental

El proceso de codificación por discretización incremental de un valor A respecto al conjunto de términos $t_1, \ldots, t_n$ consta de dos pasos:

- Paso 1: $A \rightarrow \alpha_1 t_1, \ldots, \alpha_n t_n$
- Paso 2: $\alpha_1 t_1, \ldots, \alpha_n t_n \rightarrow C(A) \equiv (C_{11}, \ldots, C_{m1}, \ldots, C_{1n}, \ldots, C_{mn})$

Para realizar la decodificación debemos realizar el proceso inverso a los dos pasos de la codificación.

A partir de $C(A) \equiv (C_{11}, \ldots, C_{m1}, \ldots, C_{1n}, \ldots, C_{mn})$ es inmediata la obtención de las compatibilidades respecto a los términos $t_1, \ldots, t_n$ sin más que seguir el proceso inverso al de la codificación:

1. – Desde i = 1 hasta n
   1.1. – $\alpha_i' = 0$
   1.2. – Desde j = 1 hasta m
      1.2.1. – Si $C_{ji} = 1$ entonces
         1.2.1.1. – $\alpha_i' = \alpha_i' + \frac{1}{m}$
      1.2.2. – Fin
   1.3. – Fin
2. – Fin

Ahora bien, la recuperación de un cierto A a partir de las compatibilidades con los términos $(\alpha_1, \ldots, \alpha_n)$ no es inmediata porque el conjunto de términos de una variable lingüística no es cerrado bajo las operaciones conjuntistas o aritméticas propias de los conjuntos difusos. Por ejemplo, si R_1 y R_2 son valores lingüísticos de una variable cuantitativa X cuya semántica venga dada por los números reales difusos r_1 y r_2, parece lógico pensar que el producto de R_1 y R_2 se defina por medio del producto extendido $r_1 \otimes r_2$. El problema es que no tiene por qué existir una etiqueta en $T(X)$ cuya semántica sea $r_1 \otimes r_2$. Es lógico esperar que la salida de un modelo tenga las mismas características que la entrada por lo que si

estamos empleando términos lingüísticos como entrada podemos exigir que la salida sea una etiqueta que represente un cierto valor lingüístico.

Este problema es el de la aproximación lingüística y se puede optar por utilizar cualquier método de los habitualmente aceptados como, por ejemplo, el del centroide, para obtener la etiqueta lingüística que mejor representa al subconjunto difuso obtenido como resultado en el modelo.

2.2.5 Codificación de reglas lingüísticas IF-THEN mediante discretización incremental

Algunos sistemas de tratamiento de información difusa pueden ser modelados como sistemas basados en reglas difusas y una regla es una asociación de patrones causa-efecto susceptible de ser codificada de forma discreta mediante discretización incremental. La complejidad de la regla, que varía según el número de consecuentes asociados a un antecedente y los conectores que los unen, se refleja en asociaciones de patrones más complejas.

Conocido el método de codificación de los valores de una variable lingüística para las distintas formas en que pueden expresarse, bien sea en términos de compatibilidad, en términos crisp o en términos lingüísticos, la codificación de reglas se reduce a la codificación de sus antecedentes y consecuente.

De este modo, una regla con antecedente y consecuente difuso de la forma (A, B) , donde A y B se describen mediante las n y p etiquetas sobre las que pueden tomar valor las respectivas variables lingüísticas, se expresa mediante vectores de características que tienen por componentes las n y p etiquetas lingüísticas sobre las que se evalúan A y B respectivamente.

La codificación de la regla resultará en un vector de bits que representan las compatibilidades del antecedente A y del consecuente B con las n y p etiquetas lingüísticas sobre las que se evalúan A y B respectivamente.

Si la precisión en todas las etiquetas lingüísticas del antecedente A es j y la de las etiquetas del consecuente B es k, la regla difusa (A, B) se puede expresar como un par de vectores de $n \times j$ y $p \times k$ componentes respectivamente.

Por ejemplo, tomemos el siguiente conjunto de reglas que definen un proceso de control del frenado de un móvil y cuya codificación usando 4 bits de precisión es la mostrada en la figura 2.10:

1. Si posición positiva alta y velocidad cero entonces fuerza negativa alta.
2. Si posición positiva baja y velocidad positiva entonces fuerza negativa baja.
3. Si posición positiva baja y velocidad negativa entonces fuerza cero.
4. Si posición negativa alta y velocidad cero entonces fuerza positiva alta.
5. Si posición negativa baja y velocidad negativa entonces fuerza positiva baja.
6. Si posición negativa baja y velocidad positiva entonces fuerza cero.
7. Si posición cero y velocidad cero entonces fuerza cero.

2.2.6 Extensión de la codificación de etiquetas lingüísticas por discretización incremental para representación de varias medidas de incertidumbre

Puede ser interesante enriquecer la información presentada al sistema mediante el empleo simultaneo de la posibilidad y la necesidad de dicha información. Del mismo modo puede interesarnos el empleo de otras medidas de incertidumbre cualquiera.

Extenderemos el método de codificación por discretización incremental para su uso en casos de empleo de dos medidas de incertidumbre.

Figura 2.10: Codificación con 4 bits de precisión

	POSICIÓN					VELOCIDAD			FUERZA				
	NM	NB	CE	PB	PM	NB	CE	PB	NM	NB	CE	PB	PM
1	0000	0000	0000	0000	1111	0000	1111	0000	1111	0000	0000	0000	0000
2	0000	0000	0000	1111	0000	0000	0000	1111	0000	1111	0000	0000	0000
3	0000	0000	0000	1111	0000	1111	0000	0000	0000	0000	1111	0000	0000
4	1111	0000	0000	0000	0000	0000	1111	0000	0000	0000	0000	0000	1111
5	0000	1111	0000	0000	0000	1111	0000	0000	0000	0000	0000	1111	0000
6	0000	1111	0000	0000	0000	0000	0000	1111	0000	0000	1111	0000	0000
7	0000	0000	1111	0000	0000	0000	1111	0000	0000	0000	1111	0000	0000

Si nos centramos en el empleo de posibilidad y necesidad, el valor $u \in V$ se representa en términos t_i de $T(H)$ como $u \equiv (\alpha_1, \beta_1)\, t_1$, $, \ldots, (\alpha_n, \beta_n)\, t_n$, siendo α_i la posibilidad y β_i la necesidad de u con respecto a t_i.

Admitiendo que $T(H)$ está formado por las etiquetas $t_1, \ldots, t_n$, dispuestas en un orden arbitrario, asociamos a cada término t_i dos vectores de dimensión m (supuesta una precisión deseada $1/m$), uno para la posibilidad y otro para la necesidad. Por tanto, a la variable lingüística H se asocia un vector de dimensión $2m \cdot n$.

La codificación de u dará el vector de dimensión $2m \cdot n$:

$$C(u) = (C^P_{11}, \ldots, C^P_{1m}, C^N_{11}, \ldots, C^N_{1m}, \ldots, C^P_{n1}, \ldots, C^P_{nm}, C^N_{n1}, \ldots, C^N_{nm}). \tag{2.13}$$

C^P_{ij} se calcula mediante el método de discretización incremental

a partir de los valores de posibilidad α_i de los términos t_i:

$$\begin{cases} \alpha_i = 0 \ \rightarrow \ C_{ij}^P = 0;\ i = 1, \ldots, n;\ j = 1, \ldots, m \\ \alpha_i \neq 0 \ \rightarrow \ \begin{cases} \exists j \text{ t.q. } \frac{j}{m} \leq \alpha_i < \frac{j+1}{m} \\ C_{il}^P = \begin{cases} 1 \text{ si } l \leq j \\ 0 \text{ si } l > j. \end{cases} \end{cases} \end{cases} \tag{2.14}$$

Expresado de forma algorítmica, el valor de C_{ij}^P se calcula conforme al algoritmo:

1. – Desde i = 1 hasta n
   1.1. – Desde j = 1 hasta m
      1.1.1. – Si j $\leq$ Parte entera de $(\alpha_i \circ m)$
         1.1.1.1. – Entonces $C_{ij}^P = 1$
      1.1.2. – Si j $>$ Parte entera de $(\alpha_i \circ m)$
         1.1.2.1. – Entonces $C_{ij}^P = 0$
   1.2. – Fin
2. – Fin

De modo similar, C_{ij}^N se calcula a partir de los valores de posibilidad β_i de los términos t_i:

$$\begin{cases} \beta_i = 0 \ \rightarrow \ C_{ij}^N = 0;\ i = 1, \ldots, n;\ j = 1, \ldots, m \\ \beta_i \neq 0 \ \rightarrow \ \begin{cases} \exists j \text{ t.q. } \frac{j}{m} \leq \beta_i < \frac{j+1}{m} \\ C_{il}^N = \begin{cases} 1 \text{ si } l \leq j \\ 0 \text{ si } l > j. \end{cases} \end{cases} \end{cases} \tag{2.15}$$

El cálculo de C_{ij}^N se calcula mediante el algoritmo:

1. – Desde i $= 1$ hasta n
   1.1. – Desde j $= 1$ hasta m
      1.1.1. – Si j $\leq$ Parte entera de $(\beta_i \circ m)$
         1.1.1.1. – Entonces $C_{ij}^N = 1$
      1.1.2. – Si j $>$ Parte entera de $(\beta_i \circ m)$
         1.1.2.1. – Entonces $C_{ij}^N = 0$
   1.2. – Fin
2. – Fin

De este modo se ha extendido el sistema de codificación a un sistema capaz de codificar cualquier elemento del universo de discurso V en función de dos medidas cualquiera de tratamiento de incertidumbre. Por supuesto, el sistema es fácilmente extensible al empleo simultaneo de tantas funciones de representación como deseemos.

Además de información vaga o imprecisa podemos codificar información que permita un posterior análisis del grado de conflicto o ignorancia presente en el conocimiento presentado al sistema. Para ello es necesario obtener los datos conforme a dos medidas de incertidumbre duales para mediante un análisis posterior determinar el grado de incertidumbre presente en el marco de representación del conocimiento.

Para ello se pueden usar como medidas de tratamiento de incertidumbre la posibilidad y la necesidad. De este modo, si B representa un elemento del marco de conocimiento y X un dato de entrada, ambos definidos en el mismo universo de discurso, la posibilidad viene dada por:

$$\Pi(X \mid B) = \sup_{z \in X} \left[\min\left(X(z), B(z)\right)\right]. \tag{2.16}$$

y la necesidad viene dada por:

$$\text{Nec}(X \mid B) = \inf_{z \in X} \left[\max\left(\left(1 - X(z)\right), B(z)\right)\right]. \tag{2.17}$$

Cuando X es una información numérica precisa, la posibilidad y la necesidad coinciden y cuando X es un intervalo, la diferencia

entre posibilidad y necesidad suele ser distinta de cero. Pedrycz [40] sugiere utilizar ambas medidas conjuntamente para determinar la incertidumbre presente en los datos suministrados al sistema ya que se cumple:

$$\Pi(X_1 \mid B) - \text{Nec}(X_1 \mid B) \leq \Pi(X_2 \mid B) - \text{Nec}(X_2 \mid B); \quad X_1 \subset X_2. \tag{2.18}$$

Para ello se define:

$$\begin{aligned} \lambda &= \Pi \\ \mu &= 1 - \text{Nec}\,(X \mid B)\,. \end{aligned} \tag{2.19}$$

Entonces podemos distinguir tres casos posibles:

- $\lambda + \mu = 1$: no existe incertidumbre en la información.
- $\lambda + \mu > 1$: la información X presentada al sistema es conflictiva ya que hace referencia a B y a su complementario.
- $\lambda + \mu < 1$: la información X proporcionada al sistema presenta ignorancia ya que no aporta la suficiente información como para tomar una decisión a favor o en contra de B.

2.3 Notas finales

Las variables lingüísticas son las herramientas idóneas para manejar información imprecisa y el método de Codificación por Discretización Incremental nos va a permitir representarlas de forma discreta.

Usando las variables lingüísticas para manejar información imprecisa, nuestro objetivo es la construcción de una memoria asociativa de alta capacidad que las almacene. El método CDI permitirá usar una memoria asociativa discreta para dicho almacenamiento.

En el capítulo siguiente construiremos un modelo discreto de alta capacidad de almacenamiento a partir del cual construiremos dos modelos difusos.

Capítulo 3

Memoria Asociativa Clasificadora

3.1 Introducción

Como comentamos anteriormente, la utilidad de una memoria radica principalmente en su capacidad de almacenamiento. Para que nos sea útil necesitamos que pueda almacenar los patrones que necesitemos memorizar.

La mayoría de las memorias asociativas "clásicas" tienen una capacidad demasiado baja, lo que limita su utilidad real. Diversos autores han propuesto modificaciones para aumentar dicha capacidad pero, aunque los resultados son buenos, nos proponemos diseñar una memoria con una capacidad mayor.

El problema de la capacidad es provocado por dos factores importantes: los estados espurios y la dependencia de los datos.

- Los estados espurios aparecen cuando en el proceso de recuperación el sistema alcanza un estado que no corresponde a ningún patrón almacenado, es decir, se recupera un patrón inexistente. Esto es debido a que los patrones memorizados se almacenan en todos los pesos de las "conexiones" de los EP, de tal modo que al introducir uno nuevo se "mezcla" con los ya almacenados. De este modo los patrones almacenados pueden interferir entre sí haciendo que el patrón recupera-

do no sea ninguno de ellos. Los estados espurios pueden ser eliminados imponiendo determinadas condiciones sobre los patrones que se van a almacenar pero esto nos conduce al segundo factor que limita la capacidad: la dependencia de los datos.

- La dependencia de los datos limita el tipo de información que se puede memorizar sin interferencias. Este problema aparece en aquellos modelos de memorias asociativas que imponen condiciones a los patrones que se almacenarán. Este es el caso de la LAM, que sólo garantiza la perfecta recuperación de conjuntos de patrones siempre que sean ortogonales. Otros modelos no imponen condiciones pero su capacidad se ve muy afectada si no se memorizan determinado tipo de patrones. Este es el problema de la BAM cuya capacidad es baja cuando los patrones tienen un número muy distinto de componentes con cada uno de los dos posibles valores (0 y 1 en el caso binario, -1 y 1 en el caso bipolar).

En general podemos decir que se consigue aumentar la capacidad de la memoria a costa de que sólo se memoricen determinado tipo de patrones. En la mayoría de las ocasiones esto lleva a la imposibilidad de trabajar con patrones reales o, al menos, obliga a un preprocesamiento de los mismos para conseguir que cumplan las condiciones impuestas.

Para conseguir una memoria asociativa de alta capacidad es necesario que se eviten los problemas de estados espurios y dependencia de los datos. La *Memoria Asociativa Clasificadora* CLAM que proponemos en este capítulo surge como alternativa frente al resto de memorias asociativas al tener una topología y funcionamiento que evitan los factores que provocan la baja capacidad de almacenamiento.

En los próximos apartados se describirá la CLAM abordando los siguientes aspectos:

- Descripción de la topología de la CLAM
- Justificación del modo en que se consigue almacenar los patrones asociándolos a un conjunto de patrones internos orto-

normales

- Justificación de la arquitectura que presenta la CLAM
- Funcionamiento de las fases de aprendizaje y recuperación
- Demostración de la recuperación de todos los patrones almacenados y de la clasificación de los patrones de entrada por el vecino más cercano
- Estudio de la eficiencia de la CLAM comparándola con la de la BAM
- Se verá cómo los patrones almacenados establecen una partición del espacio en clases y cómo se puede incorporar un radio de error para distinguir los patrones que no pertenecen a ninguna de ellas
- Propondremos modificaciones que permitan asignar distinta importancia a las componentes de los patrones
- Explicaremos cómo realizar aprendizaje no supervisado con la CLAM

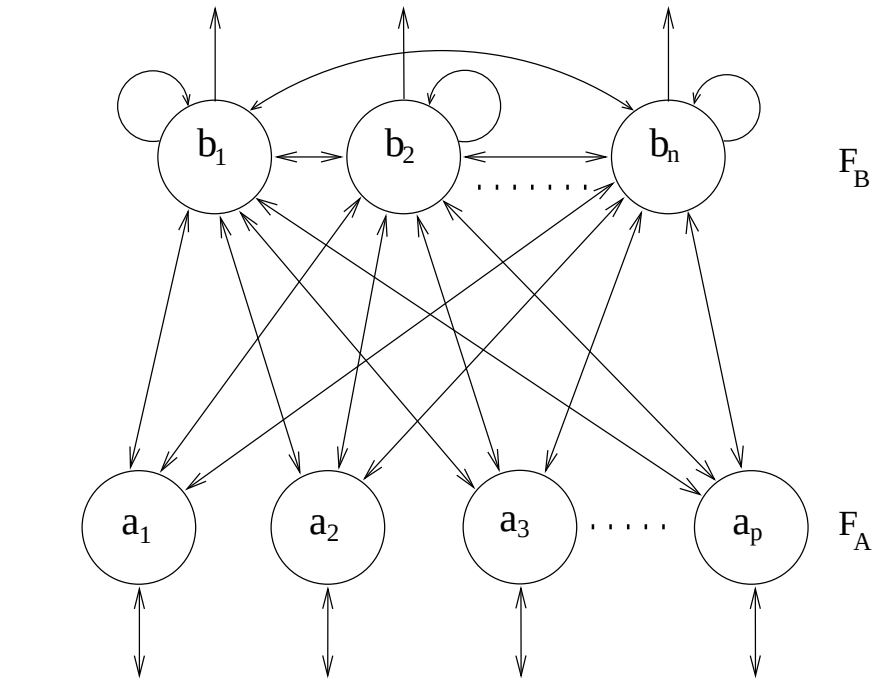
3.2 Topología

La CLAM [6, 7, 8] es un modelo de memoria asociativa formado por dos capas de EP que clasifica patrones bipolares por el criterio del vecino más cercano según la distancia de Hamming. Su topología se muestra en la figura 3.1 (más adelante se justificará el empleo de los elementos propuestos).

La topología de la memoria es dinámica porque el número de EP de la capa F_B y el número de conexiones cambian en el tiempo según varía el número de patrones almacenados.

La capa F_A está formada por p EP bipolares donde p es el número de componentes de los patrones que queremos memorizar. Los EP de esta capa no están umbralizados y su función de activación es la función signo.

Figura 3.1: Memoria Asociativa Clasificadora (CLAM)



En esta capa se presenta el patrón de entrada y, tras el proceso de recuperación, se muestra el patrón almacenado que es su vecino más cercano.

La capa F_B es de tipo competitivo (*winner-takes-all*) formada por EP de salida binaria no umbralizados y con función de activación identidad. Estos elementos representan los patrones almacenados y, por tanto, el número de EP de la capa F_B depende del número de patrones almacenados.

Las dos capas se encuentran completamente conectadas entre sí de modo que los pesos de dichas conexiones se pueden representar mediante una matriz de pesos $M_{n\times p}$, siendo n el número de patrones almacenados y p el número de componentes de cada patrón.

Cuando almacenamos un nuevo patrón se crea en la capa F_B un

nuevo EP que lo representa y se ajustan los pesos de las conexiones entre los elementos de la capa F_A y el recién creado de tal modo que, cuando se presenta dicho patrón en la capa F_A, el proceso de recuperación activa el EP de la capa F_B que lo representa. Por tanto podemos asociar a cada patrón almacenado A_i un patrón binario B_i que representa los estados de activación de los elementos de la capa F_B obtenidos tras el proceso de recuperación.

Obsérvese que los patrones binarios B_i que se asignan internamente a los patrones almacenados son los que soportan las condiciones necesarias para garantizar la recuperación de todos los patrones y la ausencia de estados espurios.

La CLAM recupera el patrón almacenado que es el vecino más cercano al patrón de entrada. Una clase está formada por el conjunto de todos los patrones que, al suministrarlos como entrada a la CLAM, dan lugar a la recuperación del mismo patrón. Dicho patrón recuperado será el representante de la clase y el nuevo EP que se creó en la capa F_B no sólo representa a dicho patrón sino que representa a la clase a la que pertenece. Por esta razón hemos introducido el calificativo de *clasificadora* en el nombre de esta memoria asociativa al tratarse de una importante característica diferenciadora de nuestro modelo.

Supongamos que memorizamos los patrones A y B en una CLAM. En la capa F_B se crean los elementos b_A y b_B asociados a cada uno de los patrones almacenados.

- Una clase, representada por el patrón A, estará formada por todos los patrones que se encuentran más cerca de A que de B.
- Otra clase, representada por B, la formarán los patrones que se encuentran más cerca de B que de A.
- La activación del elemento b_A en la capa F_B identificará al patrón de entrada como perteneciente a la clase representada por A.
- De igual modo, la activación del elemento b_B identificará al patrón de entrada como perteneciente a la clase representada por B.

Ilustraremos esta idea con un ejemplo:

En una CLAM que almacena patrones de cinco componentes memorizamos:

$$A = (+1 + 1 - 1 - 1 - 1)$$
$$B = (-1 + 1 + 1 - 1 + 1).$$

Estos dos patrones almacenados son los representantes de las dos clases por las que se clasificarán los patrones de entrada. En la siguiente tabla se muestran todos los posibles patrones de entrada junto con las distancias que los separan de los patrones A y B y la clase por la que se clasificarán.

Patrón	Dis.A	Dis.B	Clase	Patrón	Dis.A	Dis.B	Clase
-1-1-1-1-1	2	3	A	+1-1-1-1-1	1	4	A
-1-1-1-1+1	3	2	B	+1-1-1-1+1	2	3	A
-1-1-1+1-1	3	4	A	+1-1-1+1-1	2	5	A
-1-1-1+1+1	4	3	B	+1-1-1+1+1	3	4	A
-1-1+1-1-1	3	2	B	+1-1+1-1-1	2	3	A
-1-1+1-1+1	4	1	B	+1-1+1-1+1	3	2	B
-1-1+1+1-1	4	3	B	+1-1+1+1-1	3	4	A
-1-1+1+1+1	5	2	B	+1-1+1+1+1	4	3	B
-1+1-1-1-1	1	2	A	**+1+1-1-1-1**	0	3	A
-1+1-1-1+1	2	1	B	+1+1-1-1+1	1	2	A
-1+1-1+1-1	2	3	A	+1+1-1+1-1	1	4	A
-1+1-1+1+1	3	2	B	+1+1-1+1+1	2	3	A
-1+1+1-1-1	2	1	B	+1+1+1-1-1	1	2	A
-1+1+1-1+1	3	0	B	+1+1+1-1+1	2	1	A
-1+1+1+1-1	3	2	B	+1+1+1+1-1	2	3	A
-1+1+1+1+1	4	1	B	+1+1+1+1+1	3	2	B

3.3 Ortonormalidad

En realidad nuestra propuesta para la memoria asociativa clasificadora almacena pares de patrones. Un miembro del par es el patrón externo que el usuario quiere almacenar mientras que el otro patrón es uno asignado internamente por la memoria. Esta asignación se realiza de modo que:

- Se logra una perfecta recuperación.

- No se requieren condiciones sobre el patrón externo.

Para ello imponemos que los patrones internos sean ortonormales. Esta fuerte condición permite el almacenamiento de cualquier patrón externo sin establecer ningún tipo de restricción sobre él.

Para probar la ortonormalidad de los patrones internos debemos tener en cuenta el modo en que éstos son construidos.

Cuando hablamos de los patrones internos nos referimos a aquellos que representan los estados de activación de los EP de la capa F_B asociados a cada patrón. Supongamos que se han memorizado n patrones $A_1, \ldots, A_n$. En ese caso tendremos n patrones internos $B_1, \ldots, B_n$ que representan los respectivos estados de activación de la capa F_B.

Sea el conjunto de patrones internos $B_1, \ldots, B_n$, siendo $B_i = (b_{i1}, \ldots, b_{in})$.

Cada EP de la capa F_B ha sido asignado a un patrón almacenado. Por tanto todo patrón interno B_i tendrá activa la componente correspondiente al EP de la capa F_B asociado al patrón almacenado A_i:

$$\forall i \in [1, n] \, \exists s \qquad b_{is} = 1. \tag{3.1}$$

A un patrón almacenado se le asigna sólo un EP de la capa F_B. Por tanto, en cada patrón interno encontraremos una única componente activa mientras que las demás permanecerán inactivas:

$$\forall i \in [1, n] \, \forall s \neq t \qquad (b_{is} = 1) \Rightarrow (b_{it} \neq 1) \, . \tag{3.2}$$

Una vez que se ha asignado un EP de la capa F_B a un patrón almacenado no se vuelve a asignar a ningún otro. Por ello no podremos encontrar dos patrones internos que tengan activa la misma componente puesto que esto implicaría que el mismo EP de F_B fue asignado a dos patrones distintos:

$$\forall i \neq j \forall s \qquad (b_{is} = 1) \Rightarrow (b_{js} \neq 1) \, . \tag{3.3}$$

Por otra parte y, puesto que no se crean EP en la capa F_B sin que éstos sean asociados a algún patrón almacenado, no existirá ninguna componente en los patrones internos que sea siempre cero

(para asegurar esto debemos impedir que en el proceso de aprendizaje se memorice más de una vez el mismo patrón). Cualquier EP i de la capa F_B se creó para representar a algún patrón A_j; la componente i del patrón interno asociado a A_j valdrá 1. Es decir, para toda componente existe un patrón interno en el que vale 1:

$$\forall s \in [1,n]\ \exists i \qquad b_{is} = 1. \tag{3.4}$$

Para probar la ortonormalidad de los patrones internos definiremos una permutación que simplificará la demostración.

Proposición 1. *Dado el conjunto de patrones internos $B_1, \ldots, B_n$, siendo $B_i = (b_{i1}, \ldots, b_{in})$ existe una permutación σ definida como:*

$$\begin{array}{rccc} \sigma: & \mathbb{N} & \to & \mathbb{N} \\ & [1,n] & \mapsto & [1,n] \end{array} \qquad \sigma(s) = i \ \Leftrightarrow\ b_{is} = 1. \tag{3.5}$$

Demostración. Para probar que podemos encontrar la permutación σ cualesquiera que sean los patrones internos, necesitamos probar que σ es una función biyectiva que aplica los elementos de un conjunto sobre sí mismo. Por tanto dividiremos la demostración en las tres demostraciones más simples correspondientes a la existencia de la función y a la inyectividad y sobreyectividad de la misma.

- σ es una función, es decir, cada elemento del dominio tiene su imagen única en el rango. Primero probaremos que cada elemento del dominio tiene su imagen, es decir, $\forall s \exists i\ \sigma(s) = i$ y después probaremos que dicha imagen es única, es decir, $\forall i \neq j \forall s\ \ (\sigma(s) = i) \Rightarrow (\sigma(s) \neq j)$.

  A partir de la ecuación (3.4) tenemos:

  $$\forall s \in [1,n]\ \exists i\ b_{is} = 1. \tag{3.6}$$

  Al aplicar la definición de σ (3.5) la expresión anterior se transforma en:

  $$\forall s \exists i\ \sigma(s) = i \qquad \text{c.q.d.} \tag{3.7}$$

Del mismo modo, a partir de la ecuación (3.3) tenemos:

$$\forall i \neq j \forall s \ \ (b_{is} = 1) \Rightarrow (b_{js} \neq 1)\,. \tag{3.8}$$

Al aplicar de nuevo la definición de σ obtenemos:

$$\forall i \neq j \forall s \ \ (\sigma(s) = i) \Rightarrow (\sigma(s) \neq j) \qquad \text{c.q.d.} \tag{3.9}$$

- σ es una función inyectiva. Para ello vamos a probar que elementos distintos del dominio tienen imágenes distintas en el rango, es decir, $\forall i \forall s \neq t \ \ (\sigma(s) = i) \Rightarrow (\sigma(t) \neq i)$.

  De la ecuación (3.2) tenemos:

  $$\forall i \in [1, n] \, \forall s \neq t \ \ (b_{is} = 1) \Rightarrow (b_{it} \neq 1)\,. \tag{3.10}$$

  Aplicando la definición de σ (3.5) obtenemos:

  $$\forall i \forall s \neq t \qquad (\sigma(s) = i) \Rightarrow (\sigma(t) \neq i)\,. \qquad \text{c.q.d.} \tag{3.11}$$

- σ es una función sobreyectiva. Debemos probar que cada elemento del rango es imagen de algún elemento del dominio, es decir, $\forall i \exists s \ \sigma(s) = i$.

  De la ecuación (3.1) tenemos:

  $$\forall i \in [1, n] \, \exists s \ b_{is} = 1. \tag{3.12}$$

  Al aplicar la definición de σ (3.5) obtenemos:

  $$\forall i \exists s \qquad \sigma(s) = i \qquad \text{c.q.d.} \tag{3.13}$$

Corolario 1.

$$\forall s \in [1, n] \qquad b_{\sigma(s)s} = 1. \tag{3.14}$$

Corolario 2.

$$\forall s \neq t \qquad b_{\sigma(s)t} \neq 1. \tag{3.15}$$

La permutación σ asociada al conjunto de patrones internos $B_1, \ldots, B_n$ nos permitirá comprobar fácilmente la ortonormalidad de los mismos.

Proposición 2. *Sean $B_1, \dots B_n$ los patrones internos de una CLAM que ha almacenado n patrones. Los patrones $B_1, \dots, B_n$ son ortonormales:*

- *Ortogonalidad:*

$$B_i B_j^T = 0 \qquad \forall i \neq j. \tag{3.16}$$

- *Normalidad*

$$\|B_i\| = B_i B_i^T = 1. \tag{3.17}$$

Demostración. La demostración de la ortonormalidad de los patrones internos la dividiremos en las demostraciones de ortogonalidad y normalidad de los mismos:

- Ortogonalidad.

  Sea σ (3.5) una permutación asociada al conjunto de patrones internos. Podemos encontrar s y t tales que (3.13):

$$\sigma(s) = i \qquad \sigma(t) = j. \tag{3.18}$$

  Por tanto, $B_i = B_{\sigma(s)}$ y $B_j = B_{\sigma(t)}$. Entonces podemos escribir:

$$B_i B_j^T = B_{\sigma(s)} B_{\sigma(t)}^T. \tag{3.19}$$

  Al expandir el producto de los dos vectores obtenemos:

$$B_i B_j^T = B_{\sigma(s)} B_{\sigma(t)}^T = \sum_{k=1}^{n} b_{\sigma(s)k} b_{\sigma(t)k}. \tag{3.20}$$

  Sacando de la sumatoria los sumandos correspondientes a $k = s$ y $k = t$ simplificamos la expresión:

$$B_i B_j^T = \underbrace{b_{\sigma(s)s}}_{1} \underbrace{b_{\sigma(t)s}}_{0} + \underbrace{b_{\sigma(s)t}}_{0} \underbrace{b_{\sigma(t)t}}_{1} + \sum_{k \neq s,t}^{n} \underbrace{b_{\sigma(s)k}}_{0} \underbrace{b_{\sigma(t)k}}_{0} = 0 \qquad \text{c.q.d.} \tag{3.21}$$

- Normalidad.

  Sea σ (3.5) una permutación asociada al conjunto de patrones internos. Podemos encontrar s tal que (3.13) $\sigma(s) = i$.

  Puesto que $B_i = B_{\sigma(s)}$, el módulo de B_i se puede expresar como:

  $$\|B_i\| = B_i B_i^T = B_{\sigma(s)} B_{\sigma(s)}^T. \tag{3.22}$$

  Al expandir el producto de los vectores obtenemos:

  $$\|B_i\| = \sum_{k=1}^{n} b_{\sigma(s)k} b_{\sigma(s)k}. \tag{3.23}$$

  Sacando de la sumatoria el sumando correspondiente a $k = s$ nos queda:

  $$\|B_i\| = \underbrace{b_{\sigma(s)s}}_{1}\underbrace{b_{\sigma(s)s}}_{1} + \sum_{k \neq s}^{n} \underbrace{b_{\sigma(s)k}}_{0}\underbrace{b_{\sigma(s)k}}_{0} = 1 \qquad \text{c.q.d.} \tag{3.24}$$

3.4 Aprendizaje

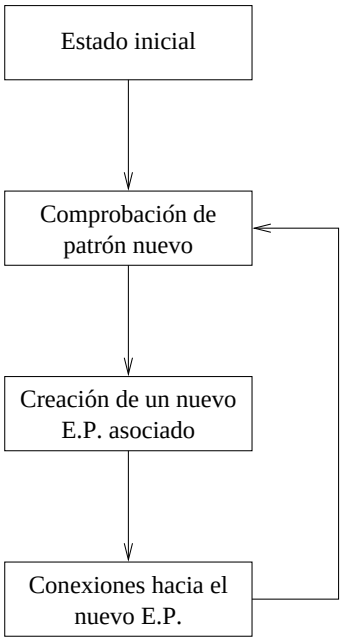
La Memoria Asociativa Clasificadora almacena los patrones por medio de un proceso que sigue los pasos esquematizados en la figura 3.2.

1. Estado inicial.

   La Memoria asociativa clasificadora modifica su topología dinámicamente de tal modo que en todo momento tiene tantos EP en la capa F_B como patrones se encuentren almacenados. Por tanto, en un estado inicial en que aún no se ha memorizado ningún patrón la capa F_B no tendrá EP. La capa F_A tendrá siempre un número de EP igual al de componentes de los patrones a memorizar.

   La matriz de pesos inicial $M_{p \times n}$ estará formada por $n = 0$ filas y p columnas.

Figura 3.2: Proceso de aprendizaje en CLAM



2. Comprobación de patrón nuevo.

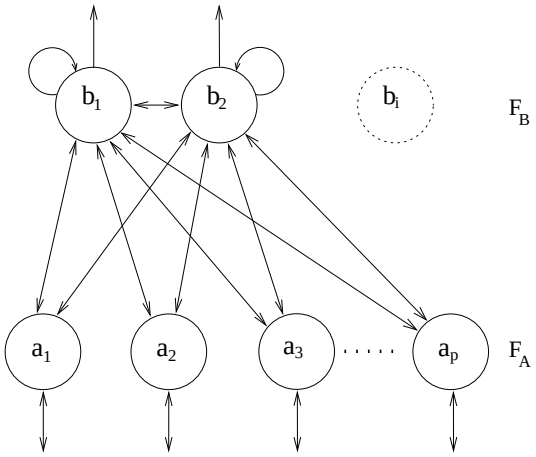
   Si memorizamos más de una vez el mismo patrón tendremos más de un EP asociado a dicho patrón. Hay que evitar esta situación no deseable porque impide lograr la ortonormalidad de los patrones internos. Para ello, antes de proseguir el proceso de aprendizaje de un patrón, verificaremos que dicho patrón no fue almacenado anteriormente.

   La memoria asociativa clasificadora siempre recupera uno de los patrones que se encuentren almacenados. De este modo, si presentamos como patrón de entrada el patrón que queremos memorizar y el proceso de recuperación obtiene un patrón distinto se puede afirmar que el patrón aún no ha sido memorizado y podemos proseguir con el proceso. Si, por el

contrario, la memoria recupera el mismo patrón que queremos almacenar, esto significa que el patrón ya está memorizado y no es necesario almacenarlo de nuevo.

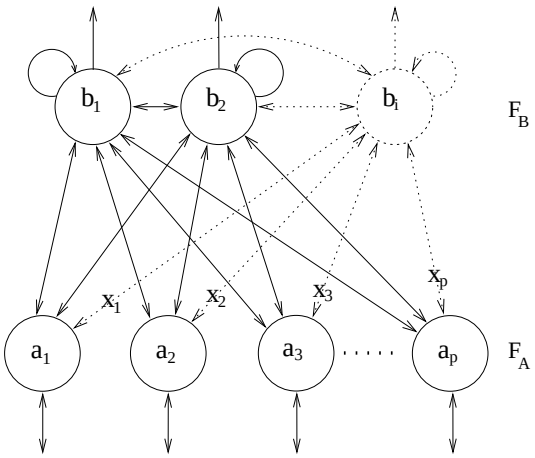
3. Creación de un nuevo EP asociado.

Para almacenar un nuevo patrón $X = (x_1, \ldots, x_p)$ se añade en la capa F_B un nuevo EP que lo representará. Gracias a la comprobación realizada en el paso 2 sabemos que no puede existir aún un EP asociado a dicho patrón.



4. Conexiones hacia el nuevo EP.

Los pesos de las conexiones que se establecen desde los EP de la capa F_A hacia el nuevo EP creado en la capa F_B se ajustan directamente como los valores de las componentes del patrón a memorizar. La matriz de pesos se verá incrementada con una nueva fila formada por los pesos de las conexiones establecidas hacia el nuevo EP.



Supongamos que una memoria asociativa clasificadora ha almacenado el siguiente conjunto de patrones distintos:

$$A_1, \ldots, A_n \qquad A_i = (a_{i1}, \ldots, a_{ip}) \qquad \forall i \neq j \; A_i \neq A_j \tag{3.25}$$

El número total de patrones almacenados es n y el número de componentes de los mismos es p. La matriz de pesos M obtenida tras el aprendizaje es:

$$M = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{pmatrix}. \tag{3.26}$$

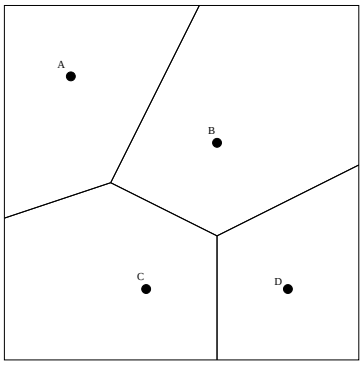
Al ser ortonormales los patrones internos B_i, la matriz de pesos se puede expresar del siguiente modo:

$$M = \sum_{i=1}^{n} B_i^T A_i. \tag{3.27}$$

3.5 Recuperación

En el proceso de recuperación, cuando se presenta un patrón de entrada, la memoria recupera el patrón almacenado más cercano. Una clase estará formada por el grupo de todos los patrones que hacen que la memoria recupere el mismo patrón que será el representante de la clase. En la figura 3.3 se muestra una representación gráfica de la partición provocada por la memorización de cuatro patrones.

Figura 3.3: Partición del espacio en clases al memorizar cuatro patrones.



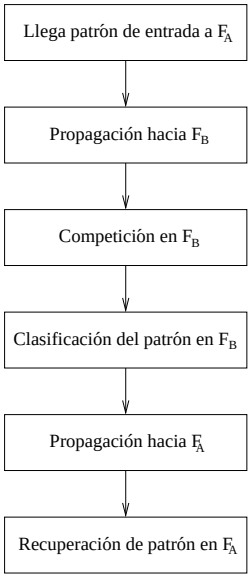
Por tanto, puede decirse que el proceso de recuperación muestra el patrón que representa a la clase a la que pertenece el patrón de entrada.

Supongamos una memoria asociativa clasificadora que ha almacenado n patrones de p componentes cada uno.

En el proceso de recuperación, en la capa de F_A se introduce un patrón de entrada, obteniéndose en la capa F_B la activación del EP que representa al patrón almacenado que es el vecino más cercano al patrón de entrada; en la capa F_A se recuperará dicho patrón almacenado.

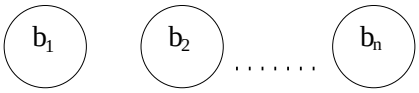
El proceso de recuperación se esquematiza pues en los pasos mostrados en la figura 3.4.

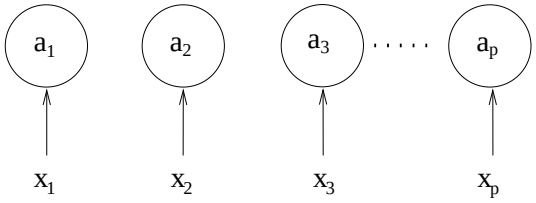
Figura 3.4: Proceso de recuperación en CLAM



1. Llega patrón de entrada a F_A.

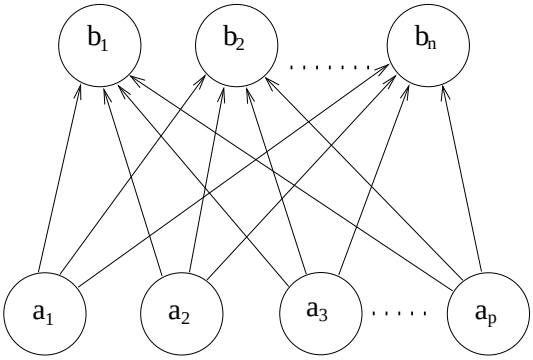
   En la capa F_A se presenta el patrón de entrada $X = (x_1, \ldots, x_p)$ a partir del cual realizaremos la recuperación.





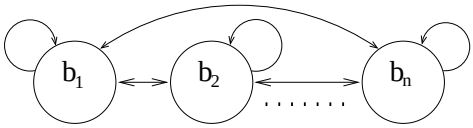
2. Propagación hacia F_B.

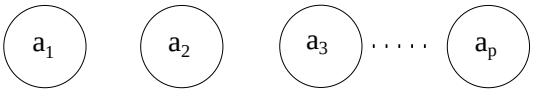
La señal de entrada se propaga hacia la capa F_B mediante las conexiones establecidas entre las capas. Los elementos de la capa F_B reciben $Y = XM^T$.



3. Competición en F_B.

Los EP de la capa F_B compiten entre sí hasta que aquél que recibió la mayor señal procedente de la capa F_A queda activo. Esta competición es del tipo *winner-takes-all* de modo que el EP ganador queda activo mientras que los demás quedan inactivos.





El estado de activación de la capa F_B tras el proceso de

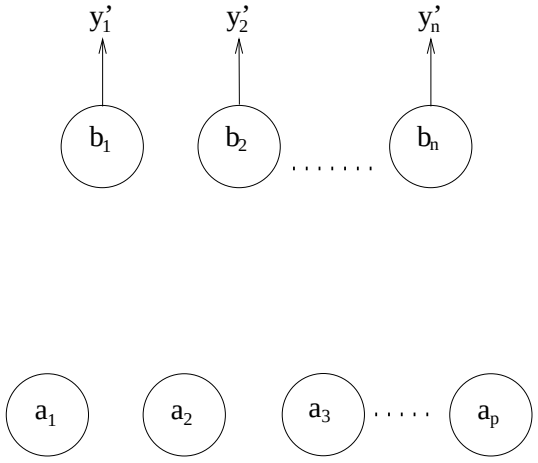
competición es $Y' = (y_1', \ldots, y_n')$ donde:

$$y_i' = \begin{cases} 1 & \text{si } y_i = \max\{y_k\} \quad \forall k = 1, \ldots, n \\ 0 & \text{en otro caso .} \end{cases} \tag{3.28}$$

Si queda más de un EP activo en la capa F_B esto significa que los patrones que representan se hallan a la misma distancia del patrón de entrada y, por tanto, todos recibieron la misma señal de activación procedente de la capa F_A. Como dichos candidatos son igualmente válidos se ha de establecer un criterio para seleccionar uno de ellos, es decir, debe quedar solamente uno de ellos activo.

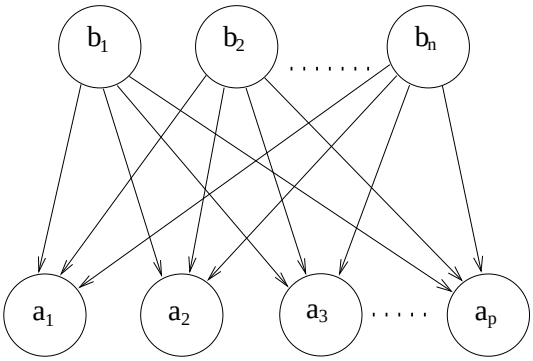
4. Clasificación del patrón en F_B.

El único EP de la capa F_B que permanece activo tras la competición es el que representa al patrón almacenado que se encuentra más cerca del patrón de entrada.



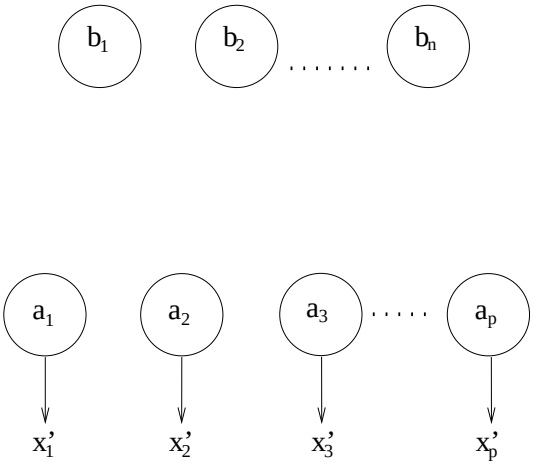
5. Propagación hacia F_A.

El estado de activación de los EP de la capa F_B se propaga hacia la capa F_A. La capa F_A recibe $X' = Y'M$



6. Recuperación de patrón en F_A.

En la capa F_A obtenemos el patrón almacenado que se encuentra a una menor distancia de hamming del patrón de entrada, es decir, obtenemos el vecino más cercano al patrón de entrada.



3.6 Ejemplo

Para mostrar con detalle los pasos de los procesos de aprendizaje y recuperación vamos a considerar un ejemplo concreto en el cual:

- Almacenaremos en una memoria asociativa clasificadora tres patrones de cinco componentes cada uno.
- A continuación presentaremos uno de dichos patrones como entrada en la CLAM para ilustrar el proceso de recuperación.
- Por último, presentaremos a la entrada el mismo patrón pero modificado por el ruido para comprobar la perfecta recuperación de uno de los patrones previamente almacenados.

Los patrones que se almacenarán son:

$$\begin{aligned} A_1 &= (+1, -1, -1, +1, +1) \\ A_2 &= (-1, -1, +1, +1, -1) \\ A_3 &= (+1, +1, -1, -1, +1)\,. \end{aligned} \tag{3.29}$$

3.6.1 Memorización

Antes de haber almacenado algún patrón no hay ningún EP en la capa F_B y por tanto la matriz de pesos tiene cinco columnas y cero filas, es decir, es una matriz vacía:

$$M = \emptyset. \tag{3.30}$$

Aprendizaje de A_1

Presentamos el patrón A_1 en la capa F_A para ver si fue anteriormente memorizado. El proceso de recuperación devuelve el resultado:

$$(0, 0, 0, 0, 0) \neq A_1. \tag{3.31}$$

Se crea un nuevo EP en la capa F_B asociado al patrón A_1 y se ajustan los pesos convenientemente. La nueva matriz de pesos es:

$$M = (+1, -1, -1, +1, +1)\,. \tag{3.32}$$

Aprendizaje de A_2

Presentamos A_2 en la capa de entrada para ver si fue memorizado anteriormente.El proceso de recuperación devuelve:

$$(+1, -1, -1, +1, +1) \neq A_2. \tag{3.33}$$

Se crea un nuevo EP en la capa F_B asociado al patrón A_2 y se ajustan los pesos. La nueva matriz de pesos es:

$$M = \begin{pmatrix} +1, -1, -1, +1, +1 \\ -1, -1, +1, +1, -1 \end{pmatrix}. \tag{3.34}$$

Aprendizaje de A_3

Presentamos el patrón A_3 en la capa F_A para comprobar si ya fue almacenado. El proceso de recuperación proporciona:

$$(+1, -1, -1, +1, +1) \neq A_3. \tag{3.35}$$

Se crea un nuevo EP asociado a A_3 y se ajustan los pesos obteniéndose la matriz:

$$M = \begin{pmatrix} +1, -1, -1, +1, +1 \\ -1, -1, +1, +1, -1 \\ +1, +1, -1, -1, +1 \end{pmatrix}. \tag{3.36}$$

Aprendizaje de A_2

Si intentamos memorizar de nuevo el patrón A_2 el proceso de recuperación obtiene en $F_A$el patrón:

$$(-1, -1, +1, +1, -1) = A_2. \tag{3.37}$$

Al coincidir el patrón recuperado y el que queremos almacenar el proceso de aprendizaje lo descarta al tratarse de un patrón que fue almacenado anteriormente.

3.6.2 Recuperación

Veamos a continuación el proceso de recuperación del patrón almacenado A_3.

Recuperación de $A_3 = (+1, +1, -1, -1, +1)$

La señal de entrada se propaga hacia la capa F_B que recibe:

$$Y = A_3 M^T = (1, -5, 5)\,. \tag{3.38}$$

Se produce una competición entre los EP de la capa F_B que da por resultado:

$$Y' = (0, 0, 1)\,. \tag{3.39}$$

En la capa F_B permanece activo el EP que representa al patrón almacenado más cercano al de entrada.

La señal se propaga hacia la capa F_A que recibe:

$$X' = Y'M = (+1, +1, -1, -1, +1) = A_3. \tag{3.40}$$

Por tanto, el patrón almacenado A_3 ha sido perfectamente recuperado.

3.6.3 Recuperación de un patrón alterado

A continuación vamos a realizar el proceso de recuperación cuando se presenta a la entrada un patrón afectado por la presencia de ruido.

Recuperación de $\widetilde{A_2} = (+1, -1, +1, +1, -1)$

La señal de entrada se propaga hacia F_B:

$$Y = \widetilde{A_2} M^T = (1, 3, -3)\,. \tag{3.41}$$

Los EP de la capa F_B compiten entre sí hasta que sólo uno queda activo:

$$Y' = (0, 1, 0)\,. \tag{3.42}$$

El EP activo de la capa F_B es el que representa al patrón almacenado que se encuentra a una menor distancia de Hamming del patrón de entrada.

La señal se propaga hacia la capa F_A:

$$X' = Y'M = (-1, -1, +1, +1, -1) = A_2. \tag{3.43}$$

El patrón recuperado A_2 es el vecino más cercano del patrón de entrada.

3.7 Justificación de la arquitectura

El proceso de recuperación, de manera similar al funcionamiento de la BAM, propaga el estado de activación de los EP de la capa F_A por medio de la matriz de pesos. Dado el patrón de entrada X y la matriz de pesos M, la señal propagada hacia la capa F_B será XM^T. El resultado de este producto es el equivalente al producto escalar de cada uno de los patrones almacenados por el patrón de entrada, dando por resultado un valor que refleja la semejanza existente entre ambos.

Si la codificación de los patrones se realizase en binario encontraríamos que no se daría la misma importancia al producto de componentes coincidentes en los casos en que éstas tomasen el valor cero y uno. Por ello, el producto escalar de los patrones codificados en binario puede dar resultados que no son satisfactorios porque queremos que ese producto refleje la semejanza existente entre ellos. Para ilustrar esta idea consideremos el ejemplo mostrado en la tabla siguiente:

Patrón almacenado A		
1 0 0 1 0 1 0 1		
Patrón de entrada X	Producto AX	Semejanza entre A y X
0 1 0 1 0 1 1 0	2	50%
0 1 1 1 1 1 1 0	2	25%
1 0 0 1 0 1 0 1	4	100%
1 1 1 1 1 1 1 1	4	50%

En el ejemplo se puede comprobar que hay patrones con el mismo porcentaje de semejanza y que dan productos distintos. Por otro lado, hay patrones cuyo producto es idéntico pero que tienen un porcentaje de semejanza distinto.

Esta diferencia entre los productos de los patrones y su semejanza se debe a que en el producto sólo influyen las componentes que coinciden con valor uno (en el resto de casos el producto de las componentes es cero). Para solucionar este problema optamos por la representación bipolar de los patrones de tal modo que se da la misma importancia al producto de componentes coincidentes ($1 \times 1 = -1 \times -1 = 1$). El mismo ejemplo mostrado anteriormente da distintos resultados cuando la codificación es bipolar.

Patrón almacenado A		
+1-1-1+1-1+1-1+1		
Patrón de entrada X	Producto AX	Semejanza entre A y X
-1+1-1+1-1+1+1-1	0	50%
-1+1+1+1+1+1+1-1	-4	25%
+1-1-1+1-1+1-1+1	8	100%
+1+1+1+1+1+1+1+1	0	50%

Se puede comprobar que, con codificación bipolar, el producto de dos patrones es mayor cuanto mayor es la semejanza existente entre ellos. Además, cuando el grado de semejanza es idéntico entre dos pares de patrones, sus productos coinciden.

La capa F_B recibe procedentes de la capa F_A unas señales de activación que miden la semejanza existente entre los patrones almacenados y el patrón de entrada. Si dicha información, tras ser transformada en binario o bipolar, fuese directamente propagada hacia la capa F_A se produciría una interferencia entre las señales propagadas por el EP que representa al patrón más cercano al de entrada y todos los demás. Para evitar esto se establece una competición entre todos los EP de la capa F_B para que sólo quede activo aquel que recibió de la capa F_A una señal de activación mayor, es decir, aquel que representa al patrón almacenado más cercano al de entrada.

Tras la competición establecida en la capa F_B permanece activo

sólo uno de los EP. Es entonces cuando el estado de activación de los EP de la capa F_B se propaga hacia la capa F_A mediante la matriz de pesos. Si el vector B representa el estado de activación de los elementos de F_B y la matriz de pesos es M, la señal propagada hacia la capa F_A es BM. Los patrones almacenados se encuentran codificados en los pesos de las conexiones que se realizan desde los EP de la capa F_A hasta el EP de la capa F_B que los representa. Por ello, la señal propagada desde el elemento activo de la capa F_B representa al patrón recuperado mientras que la señal que procede de los elementos inactivos se puede considerar como ruido.

Si la codificación de los EP de la capa F_B fuese bipolar, el ruido propagado por los elementos inactivos, es decir, aquellos con estado de activación -1, interferiría con la señal procedente del elemento activo. De este modo se recuperaría en la capa de entrada un patrón con ruido.

Esta idea se ilustra con los ejemplos que se presentan en la siguiente tabla:

Patrones almacenados	
-1+1-1+1-1+1+1-1	
-1+1+1+1+1+1+1-1	
+1-1-1+1-1+1-1+1	
+1+1+1+1+1+1+1+1	
Patrón de entrada	Patrón recuperado
-1+1-1+1-1+1+1-1	-1+0-1-1-1-1+0-1
-1+1+1+1+1+1+1-1	-1+0+1-1+1-1+0-1
+1-1-1+1-1+1-1+1	+1-1-1-1-1-1-1+1
+1+1+1+1+1+1+1+1	+1+0+1-1+1-1+0+1

Al ser los EP de la capa F_B binarios el ruido propagado por los elementos inactivos es nulo ya que el estado de activación de los mismos es cero. De este modo, el patrón recuperado no tiene ruido y se anula la interferencia producida por los EP perdedores de la competición.

3.8 Recuperación de todos los patrones almacenados

Cuando hablamos de la capacidad de una memoria asociativa nos referimos a la cantidad de patrones que se pueden almacenar y que posteriormente se pueden recuperar. Si se producen interferencias entre los patrones almacenados es muy posible que sólo se puedan recuperar algunos y no todos.

Las condiciones de ortonormalidad impuestas sobre los patrones internos permiten que todos los patrones almacenados en la memoria asociativa clasificadora puedan ser perfectamente recuperados. Para probarlo, comprobaremos cómo al suministrar a la entrada uno de los patrones almacenados el patrón recuperado es él mismo.

Supongamos que en la fase de aprendizaje se ha almacenado el siguiente conjunto de patrones:

$$A_i = (a_{i1}, \ldots, a_{ip}) \quad i \in [1, n]\,. \tag{3.44}$$

Cada patrón almacenado A_i tendrá asociado el patrón interno:

$$B_i = (b_{i1}, \ldots, b_{in})\,. \tag{3.45}$$

Tras la memorización de los patrones la matriz que representa los pesos de las conexiones establecidas entre las dos capas de EP es:

$$M = \sum_{i=1}^{n} B_i^T A_i. \tag{3.46}$$

Cuando presentamos A_e, $e \in [1, n]$ como patrón de entrada esperamos que el patrón recuperado sea A_e y que en la capa F_B quede activo el EP asociado a él, es decir, el patrón interno mostrado en la capa F_B debe ser B_e. En primer lugar presentamos el patrón A_e en la capa F_A de modo que la señal propagada hacia F_B es:

$$Y = A_e M^T = (y_1, \ldots, y_n)\,. \tag{3.47}$$

La señal propagada hacia F_B puede expresarse del siguiente modo:

$$Y = A_e \left(\sum_{i=1}^{n} B_i^T A_i \right)^T = \sum_{i=1}^{n} A_e A_i^T B_i. \tag{3.48}$$

El estado de activación del EP k-ésimo será:

$$y_k = \sum_{i=1}^{n} \left(b_{ik} \sum_{j=1}^{p} a_{ej} a_{ij} \right). \tag{3.49}$$

Como se mostró anteriormente (3.5) podemos encontrar una permutación σ asociada al conjunto de patrones internos $B_1, \ldots, B_n$. Por tanto, podemos encontrar un valor $i = \sigma(k)$. Sacando de la sumatoria el sumando $\sigma(k)$ obtenemos:

$$y_k = \underbrace{b_{\sigma(k)k}}_{1} \sum_{j=1}^{p} a_{ej} a_{\sigma(k)j} + \underbrace{\sum_{\substack{i=1 \\ i \neq \sigma(k)}}^{n} \left(\underbrace{b_{ik}}_{0} \sum_{j=1}^{p} a_{ej} a_{ij} \right)}_{0} = \sum_{j=1}^{p} a_{ej} a_{\sigma(k)j}. \tag{3.50}$$

Al estar los patrones codificados en forma bipolar, el producto de cada pareja de componentes será 1 si ambas son iguales (ambas valen 1 ó -1). En cambio, el producto de componentes distintas valdrá -1.

Supongamos que hay d_k pares a_{ej} , $a_{\sigma(k)j}$ tales que $a_{ej} \neq a_{\sigma(k)j}$. En tal caso existirán $p - d_k$ pares en que se cumpla $a_{ej} = a_{\sigma(k)j}$. Por la codificación bipolar de los patrones, la expresión (3.50) se puede escribir como:

$$y_k = 1\,(p - d_k) + (-1)\,d_k = p - 2d_k. \tag{3.51}$$

La distancia de Hamming existente entre dos vectores V y W, siendo $V = (v_1, \ldots, v_t)$ y $W = (w_1, \ldots, w_t)$, se define como el número

de componentes distintas que existe entre ambos. Por tanto, la distancia de Hamming vendrá dada por la siguiente expresión:

$$\mathrm{H}(V,W)=\sum_{i=1}^{t}\mathrm{g}(v_i,w_i). \tag{3.52}$$

Donde la función g se define como:

$$\mathrm{g}(a,b)=\begin{cases}1 & \text{si } a\neq b\\ 0 & \text{si } a=b.\end{cases} \tag{3.53}$$

Puesto que el número de componentes distintas entre los patrones A_e y $A_{\sigma(k)}$ es d_k podemos afirmar:

$$\mathrm{H}\left(A_e,A_{\sigma(k)}\right)=d_k. \tag{3.54}$$

Entonces la expresión (3.51) se puede reescribir del siguiente modo:

$$y_k=p-2\mathrm{H}\left(A_e,A_{\sigma(k)}\right). \tag{3.55}$$

En el proceso de aprendizaje el segundo paso previene de la memorización múltiple del mismo patrón. De ello deducimos:

$$\mathrm{H}(A_s,A_t)=\begin{cases}0 & s=t\\ >0 & s\neq t.\end{cases} \tag{3.56}$$

A partir de las expresiones (3.55) y (3.56) calculamos los valores de y_k sabiendo que, al ser biyectiva la función σ existe su inversa:

$$\begin{array}{c}y_{\sigma^{-1}(e)}=p-2\mathrm{H}\left(A_e,A_{\sigma(\sigma^{-1}(e))}\right)\\ y_{q\neq\sigma^{-1}(e)}=p-2\mathrm{H}\left(A_e,A_{\sigma(q)}\right).\end{array} \tag{3.57}$$

En la expresión anterior, la composición de una función con su inversa nos da la identidad, por lo que $\sigma\left(\sigma^{-1}(e)\right)=e$.

Puesto que σ es inyectiva no puede haber dos elementos del dominio con la misma imagen de modo que $\sigma\left(q\neq\sigma^{-1}(e)\right)\neq e$.

Volviendo al cálculo de los valores de y_k tenemos:

$$\begin{aligned} y_{\sigma^{-1}(e)} &= p - 2\underbrace{\mathrm{H}\left(A_e, A_e\right)}_{0} = p \\ y_{q\neq\sigma^{-1}(e)} &= p - 2\underbrace{\mathrm{H}\left(A_e, A_{\sigma(q)}\right)}_{>0} < p. \end{aligned} \tag{3.58}$$

Una vez propagada la señal desde la capa F_A hasta la capa F_B se produce la competición en esta última de modo que sólo el EP que recibió la señal de activación mayor permanece activo mientras que el resto quedan inactivos. Por tanto, a partir de la expresión anterior calculamos los resultados obtenidos por la competición:

$$\begin{aligned} y'_{\sigma^{-1}(e)} &= 1 \\ y'_{q\neq\sigma^{-1}(e)} &= 0. \end{aligned} \tag{3.59}$$

Por tanto el valor del patrón interno obtenido tras la competición, B_z, tendrá por componentes:

$$\begin{aligned} b_{z\sigma^{-1}(e)} &= 1 \\ b_{zq} &= 0 \; q \neq \sigma^{-1}\left(e\right). \end{aligned} \tag{3.60}$$

A partir de la definición de σ en (3.5) obtenemos:

$$b_{z\sigma^{-1}(e)} = 1 \quad \Leftrightarrow \quad \sigma\left(\sigma^{-1}\left(e\right)\right) = z \quad \Leftrightarrow \quad e = z. \tag{3.61}$$

Por tanto en la capa F_B obtenemos el patrón interno esperado B_e. El patrón ha sido clasificado correctamente. Veamos la propagación hacia la capa F_A del estado de activación de la capa F_B. La señal X' recibida en la capa F_A es:

$$X' = B_e M = B_e \sum_{i=1}^{n} B_i^T A_i. \tag{3.62}$$

Sacando de la sumatoria el sumando e-ésimo y teniendo en cuenta las propiedades de ortogonalidad (3.16) y normalidad (3.17) de los patrones B_i obtenemos:

$$X' = \underbrace{B_e B_e^T}_{1} A_e + \sum_{i\neq e}^{n} \underbrace{B_e B_i^T}_{0} A_i = A_e \qquad \text{c.q.d.} \tag{3.63}$$

Por tanto queda probado que todos los patrones almacenados pueden ser recuperados perfectamente.

3.9 Clasificación por el vecino más cercano

En el proceso de recuperación, la CLAM clasifica el patrón de entrada por la clase representada por el patrón almacenado que es su vecino más próximo. Es decir, al presentar como entrada un patrón se recuperará en la capa F_A su vecino más cercano de los almacenados y además se activará en la capa F_B el EP asociado al mismo.

Supongamos que la memoria ha almacenado el conjunto de patrones $A_1, \ldots, A_n$ asociando a cada uno de ellos el patrón interno B_i correspondiente.

Tras el aprendizaje de los patrones la matriz de pesos M será:

$$M = \sum_{i=1}^{n} B_i^T A_i. \tag{3.64}$$

Sea X el patrón de entrada presentado en la capa F_A. La señal propagada hacia la capa F_B es:

$$Y = X M^T. \tag{3.65}$$

Expandiendo la expresión anterior obtenemos:

$$Y = X \left(\sum_{i=1}^{n} B_i^T A_i \right)^T = \sum_{i=1}^{n} X A_i^T B_i. \tag{3.66}$$

La señal recibida por el EP k-ésimo de la capa F_B será:

$$y_k = \sum_{i=1}^{n} X A_i^T b_{ik}. \tag{3.67}$$

A partir de la definición de σ vamos a extraer el sumando $i = \sigma(k)$ de la sumatoria anterior:

$$y_k = X A_{\sigma(k)}^T b_{\sigma(k)k} + \sum_{i \neq \sigma(k)}^{n} X A_i^T b_{ik}. \tag{3.68}$$

Teniendo en cuenta (3.14) y (3.15) la expresión anterior se simplifica en:

$$y_k = X A^T_{\sigma(k)}. \tag{3.69}$$

La codificación bipolar de los patrones almacenados hace que el producto de componentes idénticas de 1 mientras que el producto de componentes distintas da -1. Supongamos que entre los patrones X y $A^T_{\sigma(k)}$ hay d_k componentes distintas. Entonces el valor recibido por el EP es:

$$y_k = 1\,(p - d_k) + (-1)\,d_k = p - 2d_k. \tag{3.70}$$

Teniendo en cuenta la definición de la distancia de Hamming (3.52) la expresión anterior se transforma en:

$$y_k = p - 2\mathrm{H}\left(X, A_{\sigma(k)}\right). \tag{3.71}$$

En la fase de competición establecida en la capa F_B se selecciona el EP que recibió la máxima señal procedente de la capa F_A. Para poder expresar esta selección definimos las siguientes funciones:

$$\begin{array}{lll} \max_k \{m_k\} = r & \Leftrightarrow & \forall j\ m_r \geq m_j \\ \min_k \{m_k\} = r & \Leftrightarrow & \forall j\ m_r \leq m_j. \end{array} \tag{3.72}$$

Sea c una constante positiva. Las funciones anteriores tienen las siguientes propiedades:

$$\max_k \{m_k + c\} = \max_k \{m_k\} \tag{3.73}$$

$$\max_k \{m_k - c\} = \max_k \{m_k\} \tag{3.74}$$

$$\max_k \{cm_k\} = \max_k \{m_k\} \tag{3.75}$$

$$\max_k \{-m_k\} = \min_k \{m_k\}. \tag{3.76}$$

La clase que representa al patrón de entrada es aquella dada por el único EP de la capa F_B que queda activo tras la competición.

El EP ganador b_s es el que recibe la mayor señal procedente de F_A, es decir:

$$s = \max_k \{y_k\} . \tag{3.77}$$

A partir del resultado obtenido anteriormente podemos reescribir la expresión anterior como:

$$s = \max_k \left\{p - 2\mathrm{H}\left(X, A_{\sigma(k)}\right)\right\} . \tag{3.78}$$

Aplicando las propiedades de la función $\max_k$ obtenemos:

$$s = \min_k \left\{\mathrm{H}\left(X, A_{\sigma(k)}\right)\right\} . \tag{3.79}$$

La expresión anterior calcula cuál es el patrón $A_{\sigma(k)}$ que se encuentra a una menor distancia de Hamming del patrón de entrada X y asigna a s el valor de k hallado. Por tanto, el patrón más cercano al de entrada es $A_{\sigma(s)}$ y el elemento seleccionado en la capa F_B es el EP b_s.

3.10 Eficiencia

En una memoria asociativa la eficiencia espacial es la relación que existe entre el número de patrones que es capaz de almacenar y el número de EP y conexiones que la constituyen. Por supuesto se considera como patrones almacenados a aquellos que posteriormente pueden ser perfectamente recuperados.

La eficiencia espacial de la CLAM será comparada con la de la BAM para poder comprobar el incremento de capacidad aportado por el nuevo modelo.

3.10.1 Eficiencia de CLAM

Cuando almacenamos n patrones de p componentes cada uno en una CLAM, la capa F_A debe tener tantos EP como componentes tengan los patrones y la capa F_B tendrá tantos EP como patrones almacenados. Por tanto el número total de EP necesarios para almacenar los patrones es $E = n + p$.

Las capas F_A y F_B se encuentran totalmente conectadas de modo que el número de conexiones que se establece entre ellas es $C = np$.

En la capa F_B se establece una competición entre todos los EP de modo que todos se encuentran conectados entre sí. El número de conexiones de los EP hacia sí mismos es n. El número de conexiones entre EP distintos es $\frac{n(n-1)}{2}$. Por tanto el número total de conexiones establecidas para la competición es $\frac{n^2+n}{2}$.

Sin embargo las conexiones de la competición no tienen peso de modo que no es necesario almacenarlas. Por ello, para almacenar n patrones de p componentes el espacio total necesario S es:

$$S = E + C = n + p + np. \tag{3.80}$$

3.10.2 Eficiencia de BAM

Se han realizado varios estudios encaminados a hallar la capacidad de la BAM, siendo uno de los más aceptados [18] el que estima que el número de patrones que puede almacenar es $n = \frac{q}{4 \log q}$, donde $q = \min\{r, s\}$, siendo r y s el número de EP de cada una de las capas de la BAM.

Para un mismo número de componentes, el número de patrones n alcanza su valor máximo cuando $r = s$. Por ello vamos a suponer que la BAM almacena pares de patrones en que ambos patrones son de igual tamaño.

Supongamos que tenemos una BAM con p EP divididos en dos capas de $\frac{p}{2}$ elementos cada una. Esta memoria almacenará pares de patrones de $\frac{p}{2}$ componentes cada patrón. El número máximo de pares de patrones que puede almacenar es:

$$n = \frac{\frac{p}{2}}{4 \log \frac{p}{2}} = \frac{p}{8 \log \frac{p}{2}}. \tag{3.81}$$

El número de EP de la BAM coincide con el número de componentes p. Al estar las dos capas completamente conectadas entre sí, el número de conexiones establecidas es $\left(\frac{p}{2}\right)^2$. Por tanto el espacio necesario por una BAM que almacena pares de patrones de

$\frac{p}{2}$ componentes cada patrón es:

$$S = p + \frac{p^2}{4}. \tag{3.82}$$

A partir de (3.82) vamos a hallar el tamaño del patrón en función del espacio en una BAM:

$$p = 2\left(\sqrt{S+1} - 1\right). \tag{3.83}$$

Sustituyendo en (3.81) obtenemos la expresión que refleja el número de patrones almacenables en función del espacio disponible en una BAM:

$$n = \frac{\sqrt{S+1} - 1}{4\log\left(\sqrt{S+1} - 1\right)}. \tag{3.84}$$

3.10.3 CLAM vs BAM

Para comparar la capacidad de la CLAM y la BAM vamos a usar el mismo espacio en ambas para poder comparar el número de patrones que son capaces de almacenar.

A partir de (3.80) obtenemos la capacidad de una CLAM en función del espacio y el tamaño de los patrones:

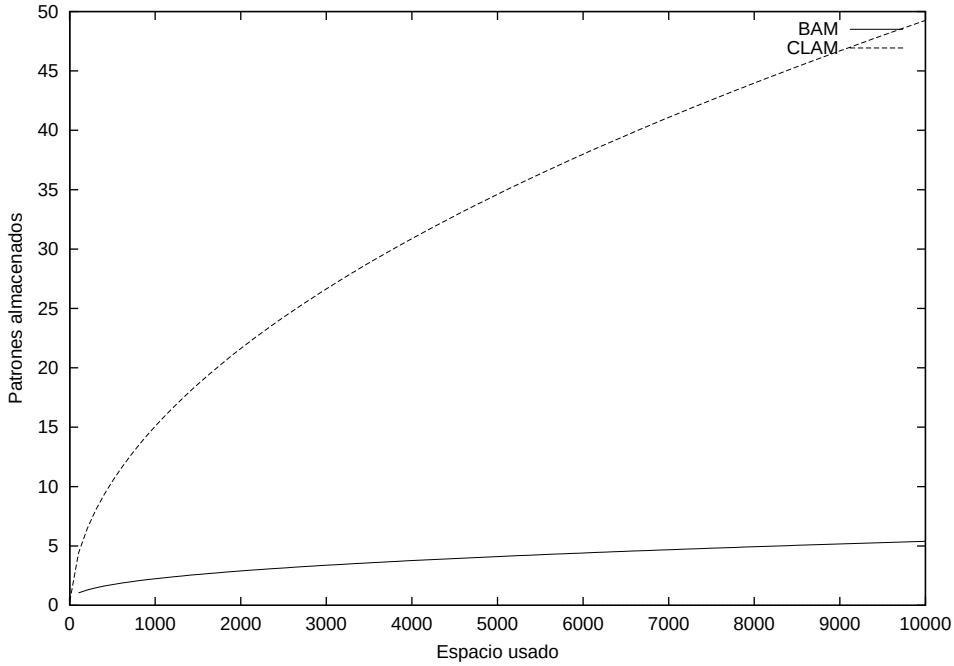
$$n = \frac{S - p}{1 + p}. \tag{3.85}$$

Como en la BAM el tamaño del patrón depende del espacio utilizado (3.83), haremos que la CLAM memorice patrones del mismo tamaño que la BAM. Por tanto, la cantidad de patrones que puede almacenar la CLAM es:

$$n = \frac{S - 2\left(\sqrt{S+1} - 1\right)}{1 + 2\left(\sqrt{S+1} - 1\right)}. \tag{3.86}$$

La relación existente entre (3.84) y (3.86) se puede observar en la figura 3.5.

Figura 3.5: Capacidad en función del espacio usado



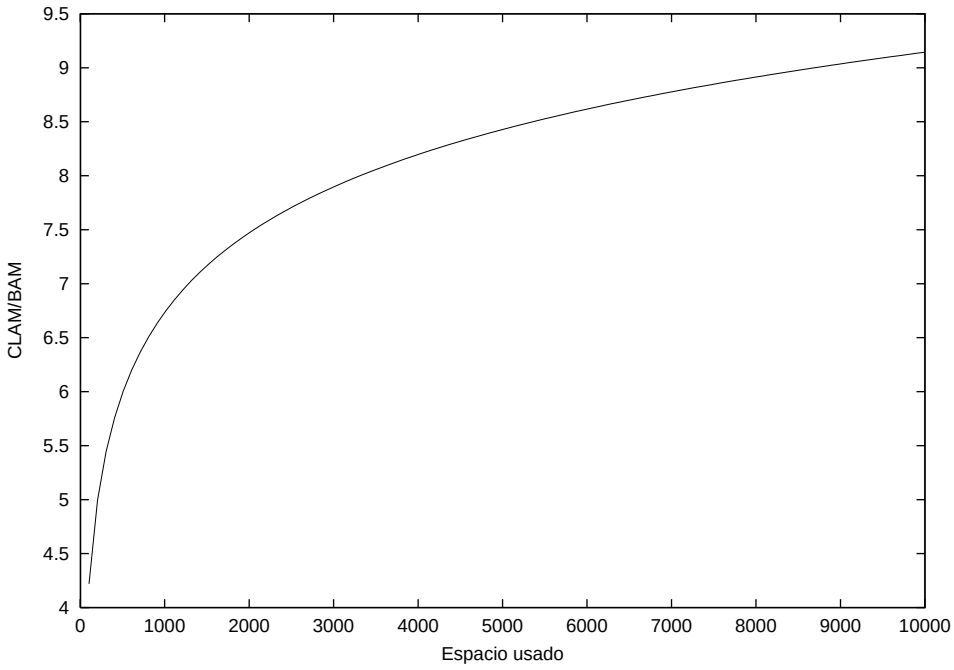
La cantidad de patrones que almacena más la CLAM que la BAM viene dada por la expresión:

$$4\frac{2S+1-\sqrt{S+1}}{4S+3}\log\left(\sqrt{S+1}-1\right). \tag{3.87}$$

Esta expresión es estrictamente creciente como se comprueba en la figura 3.6.

Mientras que en la memoria asociativa clasificadora el tamaño de los patrones no limita la capacidad de almacenamiento, en la memoria asociativa bidireccional el tamaño de los patrones p y el número de ellos que se pueden almacenar n están ligados por la expresión (3.81). La cantidad total t de información almacenada en una memoria es el producto del número de patrones n por el número de componentes de cada uno p. Es decir, $t = np$.

Figura 3.6: Relación entre las capacidades de CLAM y BAM en función del espacio usado



La cantidad total de información que almacena una BAM, teniendo en cuenta (3.81), es:

$$t = \frac{p^2}{8 \log \frac{p}{2}}. \tag{3.88}$$

Teniendo en cuenta la relación existente entre el tamaño del patrón y el espacio utilizado (3.83) podemos escribir:

$$t = \frac{\left(\sqrt{S+1}-1\right)^2}{2 \log \left(\sqrt{S+1}-1\right)}. \tag{3.89}$$

En la CLAM el tamaño del patrón no limita el número de patrones a almacenar por lo que se puede almacenar la cantidad de

información t de diversas maneras (pocos patrones muy grandes, muchos patrones pequeños,...).

Si fijamos el tamaño del patrón, a partir de (3.80) hallamos:

$$n = \frac{S-p}{p+1}. \tag{3.90}$$

De ahí hallamos la expresión que nos indica la cantidad de información almacenada:

$$t = np = p\frac{S-p}{p+1}. \tag{3.91}$$

Si almacenamos patrones de igual tamaño a los memorizados por BAM, a partir de (3.83) tenemos:

$$t = 2\frac{(S-4)\sqrt{S+1} - 3s - 4}{2\sqrt{S+1} - 1}. \tag{3.92}$$

Si, por el contrario, fijamos el número de patrones a almacenar, a partir de (3.80) hallamos:

$$p = \frac{S-n}{n+1}. \tag{3.93}$$

De nuevo calculamos la cantidad total de información almacenada:

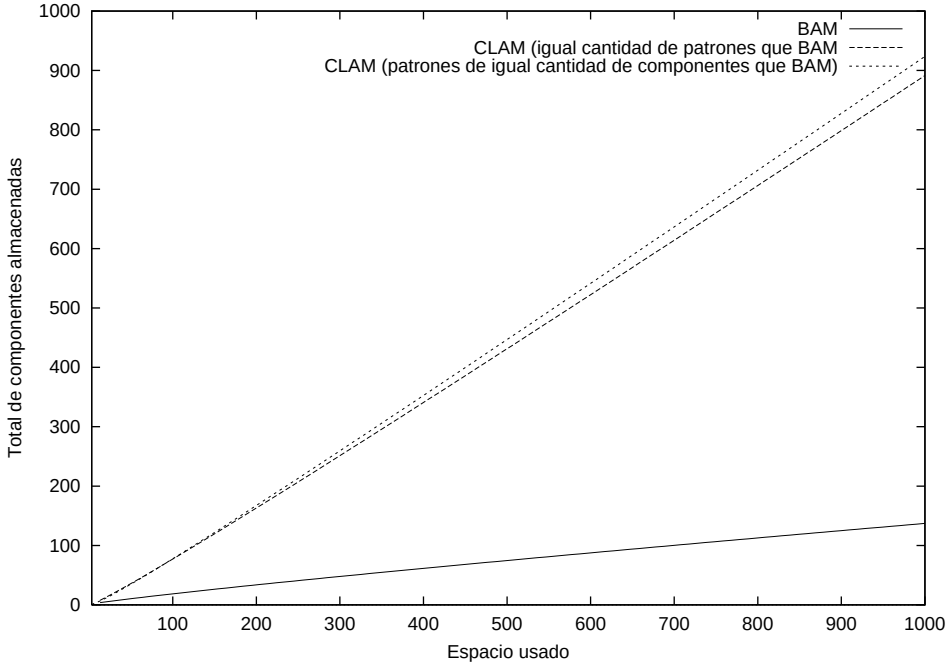
$$t = np = n\frac{S-n}{n+1}. \tag{3.94}$$

Almacenando el mismo número de patrones que la BAM, a partir de (3.84) obtenemos:

$$t = \frac{\sqrt{S+1}-1}{\log\left(\sqrt{S+1}-1\right)} \frac{s - \frac{\sqrt{S+1}-1}{\log\left(\sqrt{S+1}-1\right)}}{\frac{\sqrt{S+1}-1}{\log\left(\sqrt{S+1}-1\right)} + 1}. \tag{3.95}$$

En la figura 3.7 vemos la relación existente entre la cantidad de información que se puede almacenar en función del espacio en los casos de BAM (3.89), cuando CLAM almacena patrones del mismo tamaño que BAM (3.92) y cuando CLAM almacena el mismo número de patrones que BAM (3.95).

Figura 3.7: Cantidad de información almacenada en función del espacio usado



3.11 Partición del espacio en clases

La CLAM clasifica patrones por el criterio del vecino más cercano según la distancia de Hamming. De este modo, cuando se presenta un patrón en la capa F_A se recupera el patrón almacenado más cercano, es decir, aquel con el que coincide en un mayor número de componentes. En la capa F_B se activa el EP asociado al patrón recuperado.

Si consideramos el conjunto de todos los patrones que, al ser presentados como entrada a la memoria, hacen que ésta recupere el mismo patrón, podemos considerar a dicho patrón como representante de la clase definida por el conjunto de patrones.

Sea el conjunto de patrones bipolares distintos $\mathcal{A} = \{A_1, \ldots, A_n\}$

tales que $A_i = (a_{i1}, \ldots, a_{ip})$.

Sea la función $C_{\mathcal{A}}$ que define el comportamiento de una CLAM que ha almacenado el conjunto de patrones $\mathcal{A}$. Es decir, si a dicha CLAM se presenta el patrón de entrada X y el patrón recuperado es Y entonces $C_{\mathcal{A}}(X) = Y$:

$$C_{\mathcal{A}}: \quad \{-1,1\}^p \quad \rightarrow \quad \{-1,1\}^p \ . \tag{3.96}$$

El patrón A_i será el representante de la clase formada por el conjunto:

$$\{X \ \mid \ C_{\mathcal{A}}\,(X) = A_i\}\,. \tag{3.97}$$

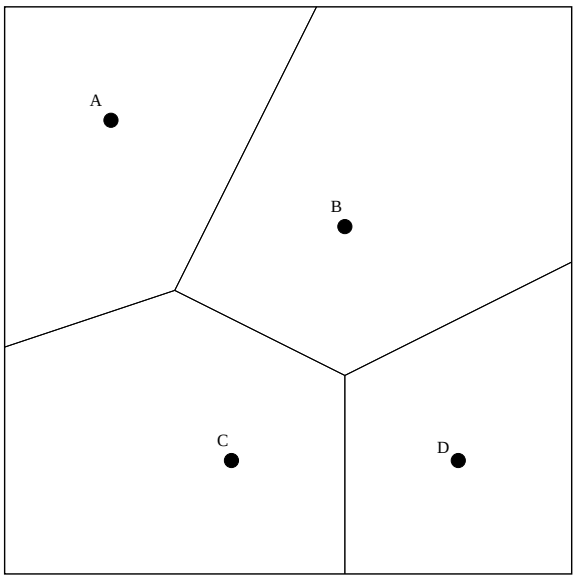
Por tanto, una memoria asociativa clasificadora establece una partición del espacio en clases disjuntas de modo que se obtiene una teselación de Voronoi de $\{-1,1\}^p$ usando la distancia de hamming. La figura 3.8 muestra una división de $[-1,1]\times[-1,1]$ en clases representadas por los elementos marcados con puntos y usando distancia Euclídea (debe tenerse en cuenta que el ejemplo es sólo ilustrativo ya que la partición que se obtiene con la memoria es de $\{-1,1\}^p$ según la distancia de hamming).

3.12 Establecimiento del radio de error

La CLAM siempre recupera un patrón de los que fueron almacenados aunque el patrón de entrada sea muy distinto de todos ellos. Siguiendo con el ejemplo ilustrativo de la figura 3.8 en el que se representa la partición en clases debida a la memorización de los patrones A, B, C y D, en la figura 3.9 podemos ver cómo el patrón de entrada E es clasificado como perteneciente a la clase representada por el patrón B aunque se encuentra muy alejado de él.

Hay ocasiones en que nos conviene que un patrón tan distinto de los almacenados no provoque la recuperación de uno de ellos sino que, de algún modo, se nos indique esta circunstancia y no se recupere ningún patrón. Es posible que el patrón de entrada E sea un patrón nuevo no memorizado previamente en vez de ser una ocurrencia de B o de algún otro patrón muy perturbado por ruido. También es posible que, aunque no sea un patrón nuevo,

Figura 3.8: Partición en clases

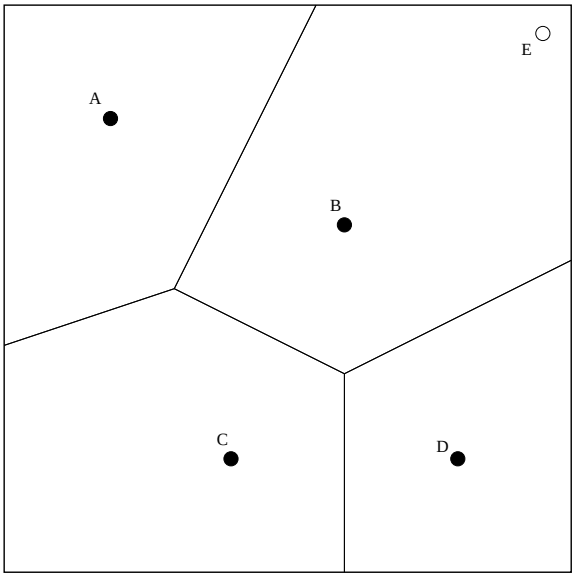


no deseemos realizar la recuperación de un patrón almacenado cuando el ruido existente es tan alto.

Vamos a establecer un radio de error ρ tal que sólo permitimos que se lleve a cabo la recuperación de un patrón cuando su distancia al patrón de entrada sea inferior a ρ. De este modo realizamos una división del espacio tal que hay patrones que no se clasifican por ninguna clase como puede verse en la parte sombreada de la figura 3.10.

En el proceso de recuperación de la memoria asociativa clasificadora, la señal recibida por cada EP de la capa F_B es proporcional a la semejanza existente entre el patrón de entrada y el patrón almacenado representado por dicho elemento. Si los patrones almacenados tienen n componentes, la señal recibida en la capa F_B varía entre p que es la señal recibida por un EP representante de un patrón idéntico al de entrada y $-p$ que es la recibida por uno representante de un patrón totalmente distinto al de entrada.

Figura 3.9: Clasificación de un patrón lejano



El radio de error se establecerá de modo que mida el error relativo cometido para independizarlo del número de componentes de los patrones. El radio de error ρ será una constante positiva que tomará valores entre cero (máxima semejanza exigida) y uno (se permite cualquier error).

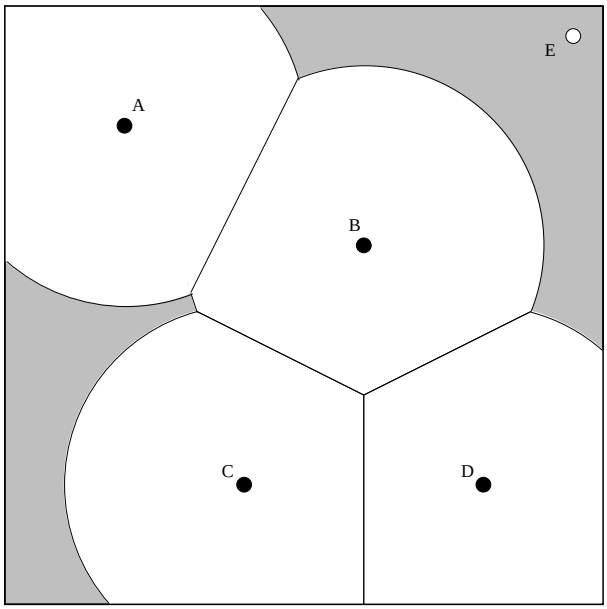
El proceso de recuperación se ve modificado de modo que el paso 3 del proceso se incluye la limitación impuesta por el radio de error:

1. Llega a F_A el patrón de entrada $X = (x_1, \ldots, x_p)$.

2. La señal de activación se propaga hacia F_B, que recibe $Y = XM^T$.

3. Se establece en F_B la competición teniendo en cuenta el radio de error ρ. El estado de los EP después de la competición es

Figura 3.10: Partición en clases con radio de error



$Y' = (y'_1, \ldots, y'_n)$

$$y'_i = \begin{cases} 1 & \text{si } y_i = \max\{y_k\} \quad \forall k = 1, \ldots, n \quad \text{y} \quad E(y_i) \leq \rho \\ 0 & \text{en otro caso .} \end{cases} \tag{3.98}$$

E es la función que calcula el error cometido por el patrón y se define como:

$$E(x) = \frac{p - x}{2p}. \tag{3.99}$$

4. En F_B se encuentra la clasificación obtenida para el patrón de entrada. Si el patrón de entrada no se encuentra suficientemente cerca de ninguno de los patrones almacenados, es decir, si el error cometido supera el umbral, ninguno de los EP de la capa F_B estará activo.

5. La señal se propaga hacia la capa F_A, que recibe $X' = Y'M$

6. Si el error cometido es inferior al umbral, en F_A se recupera el patrón almacenado más cercano al de entrada. En otro caso encontraremos en la entrada el patrón $(0, \ldots, 0)$.

3.13 Ponderación de la importancia de las componentes

Hasta ahora, cuando se calculaba la semejanza existente entre dos patrones se daba la misma importancia a todas las componentes. Sin embargo en muchas ocasiones tendremos patrones en que sus componentes representan características de distinta importancia. Por ejemplo, puede ser más importante equivocar el sexo de una persona que su color de pelo. Por ello sería conveniente poder establecer unos pesos que representen la importancia relativa de las componentes de los patrones.

Supongamos que tenemos una CLAM que almacena patrones de p componentes. El vector $R = (r_1, \ldots, r_p)$, siendo $r_i > 0$ y $\sum_{k=1}^{p} r_k = 1$, representa la importancia relativa dada a cada una de las componentes de los patrones.

En el proceso de recuperación, cada componente de entrada se operará con la correspondiente componente del vector R de modo que el estado de activación resultante de los EP de F_A será:

$$\widetilde{X} = XR^T \quad \widetilde{x_i} = x_i r_i. \tag{3.100}$$

Ahora la recuperación se realiza de modo que, una vez recibido el patrón de entrada, se pondera por el vector R antes de propagar la señal hacia F_B. La señal propagada hacia F_B será

$$Y = \widetilde{X}M^T, \tag{3.101}$$

siendo $\widetilde{X}$=$(\widetilde{x_1}, \ldots, \widetilde{x_p})$.

Cuando se ponderan las componentes se recupera el patrón más cercano al de entrada según una distancia a la que deno-

minaremos de "Hamming ponderada", definida como:

$$H_R\left(A,B\right)=\sum_{i=1}^{p}r_i\mathrm{g}\left(a_i,b_i\right), \tag{3.102}$$

siendo $R=(r_1,\ldots,r_p)$ el vector de pesos que pondera las componentes y g() la función definida en (3.53).

La incorporación de un conjunto de pesos que ponderen la importancia de las componentes no impide que todos los patrones que se almacenan en la CLAM puedan ser perfectamente recuperados.

3.13.1 Recuperación de todos los patrones memorizados

Supongamos que en la fase de aprendizaje se ha almacenado el conjunto de patrones $A_i=(a_{i1},\ldots,a_{ip})$, $i\in[1,n]$. Cada patrón almacenado A_i tendrá asociado el patrón interno $B_i=(b_{i1},\ldots,b_{in})$. Tras la memorización de los patrones la matriz que representa los pesos de las conexiones establecidas entre las dos capas de EP es $M=\sum_{i=1}^{n}B_i^TA_i$. El vector $R=(r_1,\ldots,r_p)$ marca la ponderación por importancia de las componentes.

Cuando presentamos A_e, $e\in[1,n]$ como patrón de entrada esperamos que el patrón recuperado sea A_e y que en la capa F_B quede activo el EP asociado a él, es decir, el patrón interno mostrado en la capa F_B debe ser B_e. En primer lugar presentamos el patrón A_e en la capa F_A de modo que la señal propagada hacia F_B es:

$$Y=\widetilde{A_e}M^T=A_eR^TM^T=\left(y_1,\ldots,y_n\right). \tag{3.103}$$

La señal propagada hacia F_B puede expresarse del siguiente modo:

$$Y=A_eR^T\left(\sum_{i=1}^{n}B_i^TA_i\right)^T=\sum_{i=1}^{n}A_eR^TA_i^TB_i. \tag{3.104}$$

El estado de activación del EP k-ésimo será:

$$y_k=\sum_{i=1}^{n}\left(b_{ik}\sum_{j=1}^{p}a_{ej}a_{ij}r_j\right). \tag{3.105}$$

Como se expresó en (3.5), podemos encontrar una permutación σ asociada al conjunto de patrones internos $B_1, \ldots, B_n$ de modo que $\forall i \exists s_i \; \sigma(s_i) = i$.

La expresión anterior podemos escribirla como:

$$y_k = \sum_{i=1}^{n} \left(b_{\sigma(s_i)k} \sum_{j=1}^{p} a_{ej} a_{\sigma(s_i)j} r_j \right). \tag{3.106}$$

Esta expresión se puede simplificar sacando de la sumatoria el sumando correspondiente al valor de i que hace $s_i = k$:

$$\begin{aligned} y_k &= \underbrace{b_{\sigma(k)k}}_{1} \textstyle\sum_{j=1}^{p} a_{ej} a_{\sigma(k)j} r_j + \underbrace{\sum_{\substack{i=1 \\ s_i \neq k}}^{n} \left(\underbrace{b_{\sigma(s_i)k}}_{0} \sum_{j=1}^{p} a_{ej} a_{\sigma(s_i)j} r_j \right)}_{0} = \\ &= \textstyle\sum_{j=1}^{p} a_{ej} a_{\sigma(k)j} r_j. \end{aligned} \tag{3.107}$$

Al patrón A_e se asoció el EP $b_{\sigma^{-1}(e)}$ de la capa F_B. Comprobemos ahora la señal recibida por dicho elemento:

$$y_{\sigma^{-1}(e)} = \sum_{j=1}^{p} a_{ej} a_{\sigma(\sigma^{-1}(e))j} r_j = \sum_{j=1}^{p} \underbrace{a_{ej} a_{ej}}_{1} r_j = \sum_{j=1}^{p} r_j = p. \tag{3.108}$$

Entre los patrones A_e y $A_{\sigma(k) \neq e}$ hay al menos una componente distinta. Sea la componente a_{ez} una de las que difieren. El valor de $y_{k \neq \sigma^{-1}(e)}$ es:

$$y_{k \neq \sigma^{-1}(e)} = \underbrace{a_{ez} a_{\sigma(k)z} r_z}_{-r_z} + \underbrace{\sum_{\substack{j=1 \\ j \neq z}}^{p} a_{ej} a_{\sigma(k)j} r_j}_{\leq p-1} < p \tag{3.109}$$

Al ser $y_{\sigma^{-1}(e)} > y_{k \neq \sigma^{-1}(e)}$, tras la competición el único EP activo de la capa F_B es $y_{\sigma^{-1}(e)}$. Por la definición de σ en (3.5) el patrón

interno que representa el estado de activación es B_e. El patrón recuperado es:

$$X' = B_e M = \sum_{i=1}^{n} B_e B_i^T A_i = \underbrace{B_e B_e^T}_{1} A_e + \sum_{i \neq e}^{n} \underbrace{B_e B_i^T}_{0} A_i = A_e \qquad \text{c.q.d.} \tag{3.110}$$

3.14 Aprendizaje no supervisado

Hasta ahora hemos supuesto que la memoria asociativa clasificadora realiza aprendizaje supervisado, es decir, al memorizar un patrón, junto a dicho patrón va implícita nuestra certeza de que no es erróneo, que no está afectado por ruido.

Sin embargo habrá ocasiones en que los patrones a memorizar provengan de un conjunto de ejemplos afectados por ruido. En este caso no sabemos cuál de las observaciones es la correcta (si la hay). Al desconocer cuáles son los patrones correctos no podemos realizar un aprendizaje supervisado sino que debe ser el sistema el que determine cuál es el patrón que, con mayor probabilidad, será el correcto.

Para saber qué patrones pertenecen a la misma clase, es decir, provienen de distintas observaciones afectadas por ruido realizadas sobre el mismo patrón, necesitamos establecer cuál será el error máximo permitido por encima del cual un patrón se considera perteneciente a otra clase distinta. En la fase de aprendizaje este radio de error permitirá agrupar en la misma clase las distintas observaciones del mismo patrón.

El patrón representante de una clase será el resultante de hallar la media de todos los patrones de entrenamiento que se consideraron pertenecientes a la misma y convirtiendo sus componentes en bipolar. Esta elección se debe a que la CLAM realiza entrenamiento *online* y no se conocerán a priori ni los patrones ni la cantidad total que hay.

Sea $\mathcal{A}$ el conjunto de patrones con los que se realizará el apren-

dizaje:

$$\mathcal{A} = \bigcup_{i=1}^{n} \mathcal{A}_i. \tag{3.111}$$

siendo $\mathcal{A}_i$ el conjunto de todos los patrones $\{A_{i1}, A_{i2}, \ldots\}$ que en el aprendizaje se clasificarán como pertenecientes a la misma clase:

$$\mathcal{A}_i = \{A_{i1}, A_{i2}, \ldots\}. \tag{3.112}$$

Los patrones estarán definidos como:

$$A_{ij} = (a_{ij1}, a_{ij2}, \ldots, a_{ijn}). \tag{3.113}$$

El patrón representante de la clase formada por el conjunto $\mathcal{A}_i$ será:

$$A_i = \phi\left((\overline{a_{i1}}, \ldots, \overline{a_{in}})\right) = \left(\phi\left(\overline{a_{i1}}\right), \ldots, \phi\left(\overline{a_{in}}\right)\right). \tag{3.114}$$

La función ϕ está definida como:

$$\phi(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ -1 & \text{si } x < 0. \end{cases} \tag{3.115}$$

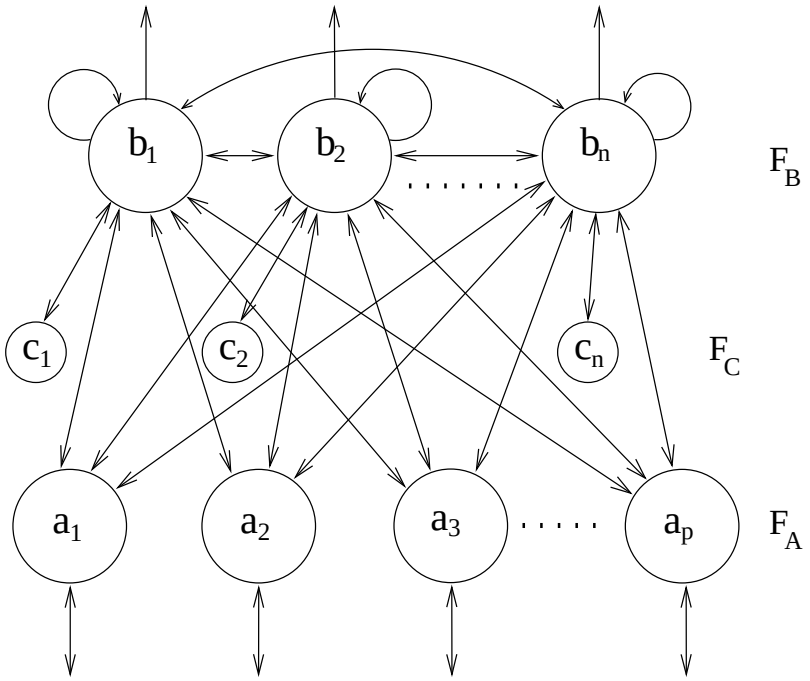
Los valores $\overline{a_{ik}}$ se calculan mediante:

$$\overline{a_{ik}} = \frac{\sum_j a_{ijk}}{|\mathcal{A}_i|}. \tag{3.116}$$

Como el proceso de aprendizaje se realiza *online*, para que se pueda calcular el valor correcto de $\overline{a_{ik}}$ hace falta conocer en cada instante el número de patrones que hasta ese momento se han memorizado como pertenecientes a la misma clase, es decir, se deberá saber el valor de $|\mathcal{A}_i|$. Para esto se llevará la cuenta del número de patrones que se han aprendido por cada una de las clases mediante unos nuevos EP que se conectarán a cada uno de los existentes en la capa F_B.

El nuevo elemento, c_i , llevará la cuenta del número de patrones que, en la fase de aprendizaje, se clasificaron por la clase representada por b_i.

Figura 3.11: CLAM con aprendizaje no supervisado



El valor de $\sum_j a_{ijk}$ se calculará acumulando los valores de las componentes de los patrones en los pesos de las conexiones que van desde la capa F_A hasta la capa F_B.

3.14.0.1 Aprendizaje

1. Llegada de un nuevo patrón X a la capa F_A.
2. Comprobar si existe una clase almacenada a la que pertenezca el patrón X. Para ello se procederá a la recuperación usando como patrón de entrada X y con el radio de error establecido ρ. Si en la capa F_B no queda ningún EP activo significa que no hay ninguna clase a la que pertenezca X. En caso contrario, el único EP activo en F_B representa a la clase a la que pertenece X.

3. Si no hay ninguna clase almacenada a la que pertenezca X se creará un nuevo EP en la capa F_B que representará a la nueva clase a la que pertenece X. A todos los pesos de las conexiones que se establecen desde la capa F_A hasta el nuevo EP se les asigna el valor cero. En la capa F_C se crea un nuevo EP asociado al recientemente creado en F_B y la conexión que los une tendrá inicialmente peso cero.

4. Sea Z el EP de F_B que representa a la clase por la que se clasificará el patrón X y C_Z el EP de F_C asociado a Z.

   (a) Los pesos de las conexiones que van desde F_A hacia Z se ven incrementados con los valores de las componentes de X, es decir, el nuevo peso de la conexión m_{iz}^{t+1} que va desde a_i hasta Z será $m_{iz}^{t+1} = m_{iz}^{t} + x_i$, siendo m_{iz}^{t} el anterior peso de la conexión.

   (b) El peso de la conexión existente entre Z y C_z se incrementa en uno. Esta conexión será por tanto la que lleve la cuenta del número de patrones aprendidos por esa clase.

3.14.0.2 Recuperación

1. Llega un patrón X a la capa F_A.

2. La señal de activación se propaga hacia la capa F_B mediante la matriz de pesos. Los elementos de F_B reciben $Y = XM^T$

3. Se realiza la ponderación por el número de patrones aprendidos por cada clase dividiendo la señal recibida y_k por el valor de c_k

4. Los EP de F_B compiten teniendo en cuenta el radio de error.

$$y_i' = \begin{cases} 1 & \textbf{si } y_i = \max\{y_k\} \quad \forall k = 1, \ldots, n \quad \textbf{y} \quad E(y_i) \leq \rho \\ 0 & \text{en otro caso.} \end{cases}$$

(a) Si ningún elemento de F_B permanece activo podemos considerar que el patrón de entrada es demasiado ruidoso para realizar una recuperación acertada o bien que se trata de un patrón perteneciente a una clase nueva. En este último caso se puede proceder al aprendizaje del mismo.

(b) Si hay un elemento activo en F_B será el representante de la clase a la que pertenece el patrón de entrada.

5. La señal se propaga hacia F_A mediante la matriz de pesos

6. En F_A se convierte la señal en bipolar obteniéndose el patrón recuperado.

3.15 Notas finales

La CLAM es un modelo de memoria asociativa de alta capacidad que nos permite almacenar patrones discretos arbitrarios. El objetivo es ahora tomar como base la CLAM para construir una memoria asociativa difusa que tenga también una elevada capacidad de almacenamiento y nos permita almacenar información imprecisa.

En el siguiente capítulo abordaremos dicha construcción de dos modos distintos. Por un lado usaremos el método CDI para emplear una CLAM discreta para el almacenamiento de información difusa. Por otro lado realizaremos unas modificaciones a la CLAM para que pueda almacenar información difusa sin perder su elevada capacidad de almacenamiento.

Capítulo 4

Memorias Asociativas Clasificadoras Difusas

4.1 Introducción

El ser humano percibe su entorno de forma imprecisa y, por tanto, los sistemas de tratamiento de información que reciben datos de un usuario humano han de estar capacitados para procesar la información en los términos en que éste es capaz de suministrarla. Cuando describimos a una persona no expresamos su altura en centímetros ni describimos su color de pelo en base a la longitud de onda electromagnética que refleja. Hablamos de personas altas, bajas, rubias, morenas, delgadas, obesas. No hay nada tan impreciso como decir que algo es normal.

Son muchas las ocasiones en que almacenamos y recuperamos información expresada de forma imprecisa. Con los sistemas discretos de almacenamiento nos vemos obligados a expresar de forma precisa algo que percibimos de forma imprecisa. Es más razonable diseñar sistemas que sean capaces de trabajar con información expresada en los términos en que el hombre la suministra.

Con el objetivo de conseguir modelos de Memoria Asociativa Clasificadora que sean capaces de trabajar con información difusa vamos a emplear dos métodos distintos:

- El método de Codificación por Discretización Incremental pa-

ra la construcción de un filtro que nos permita utilizar una CLAM discreta para el almacenamiento de información difusa expresada en términos lingüísticos.

- Construcción de una CLAM continua que conserve la alta capacidad de almacenamiento de la CLAM discreta. En ella almacenaremos información imprecisa expresada en términos lingüísticos mediante la memorización de patrones formados por los grados de compatibilidad con dichos términos.

4.2 Empleo del método de codificación por discretización incremental para la obtención de una CLAM difusa

El método de codificación por discretización incremental permite usar un sistema clásico que trabaja con información discreta para el tratamiento de información imprecisa.

Todo el intercambio de información realizado con el modelo discreto se hace a través del método de codificación por discretización incremental. El empleo del método a modo de filtro permite que el conjunto formado por el modelo discreto y el filtro se comporte como un modelo difuso capaz de trabajar con información imprecisa.

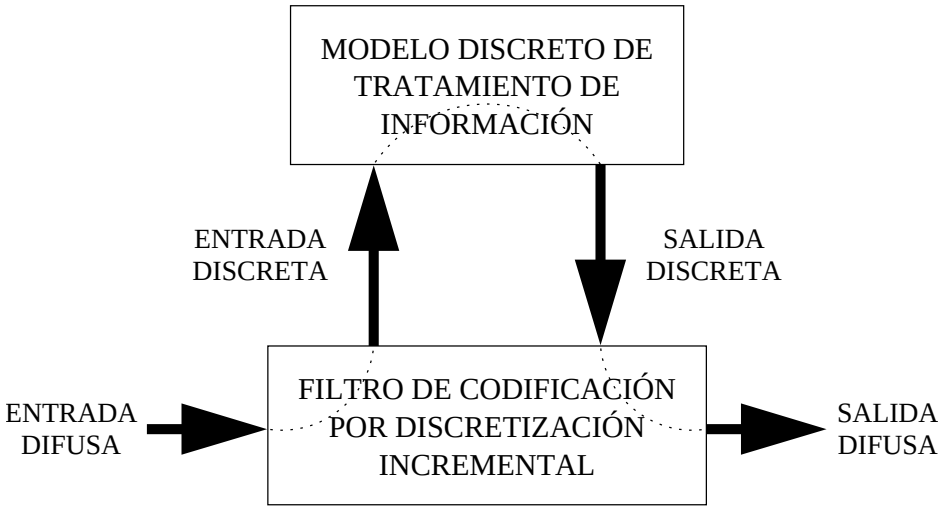
El método de codificación por discretización incremental se usa como un filtro que realiza la traducción de la información que se intercambia con la memoria:

- La información difusa que se suministra a la memoria se codifica en una representación discreta de modo que pueda ser almacenada en un modelo discreto de memoria asociativa.

- La información recuperada por la memoria, que estará representada de forma discreta, será decodificada por el filtro para obtener la información difusa que representa.

La mayoría de los modelos de memorias asociativas están diseñados para trabajar con información discreta por lo que el método

Figura 4.1: Tratamiento de información difusa en un modelo discreto mediante un filtro



de codificación de información lingüística por discretización incremental es especialmente útil ya que va a permitir almacenar información difusa en cualquier modelo de memoria asociativa discreta.

Consideramos equivalentes los elementos con idéntica compatibilidad con los términos lingüísticos de $T(H)$:

$$\left.\begin{array}{l} u \equiv \alpha_1 t_1, \ldots, \alpha_n t_n \\ v \equiv \alpha_1 t_1, \ldots, \alpha_n t_n \end{array}\right\} \Rightarrow u \equiv v; \qquad \alpha_i \in [0,1],\ t_i \in T(H). \tag{4.1}$$

Teniendo en cuenta esta consideración, la función realizada por el método de discretización es biyectiva (la relación existente entre un patrón difuso y su correspondiente codificación es biunívoca). Por tanto, al ser empleado el método CDI como filtro de un modelo discreto, todas la propiedades del modelo discreto son heredadas por el modelo difuso resultante. De este modo se pueden seguir utilizando los procedimientos de recuperación y memorización del modelo discreto de forma transparente en el nuevo modelo difuso.

4.2.1 Memoria asociativa clasificadora lingüística

Como se ha comentado, el sistema de codificación de etiquetas lingüísticas por discretización incremental usado junto con una CLAM da lugar a un modelo capaz de almacenar información difusa al que denominamos Memoria Asociativa Clasificadora Lingüística LCLAM.

Supongamos que se quieren almacenar los valores de v variables lingüísticas. La variable lingüística k se representará por medio de un conjunto de t_k etiquetas lingüísticas que se codificarán usando s_k bits de precisión. El tamaño de los patrones discretos que codificarán las variables lingüísticas y que, por tanto, determinará el número de EP de la capa F_A es:

$$p = \sum_{k=1}^{v} t_k s_k. \tag{4.2}$$

La LCLAM trabajará con patrones formados por $\sum_{k=1}^{v} t_k$ componentes continuas correspondientes a los grados de pertenencia de cada una de las etiquetas lingüísticas. Internamente se almacenarán patrones discretos de $p = \sum_{k=1}^{v} t_k s_k$ componentes.

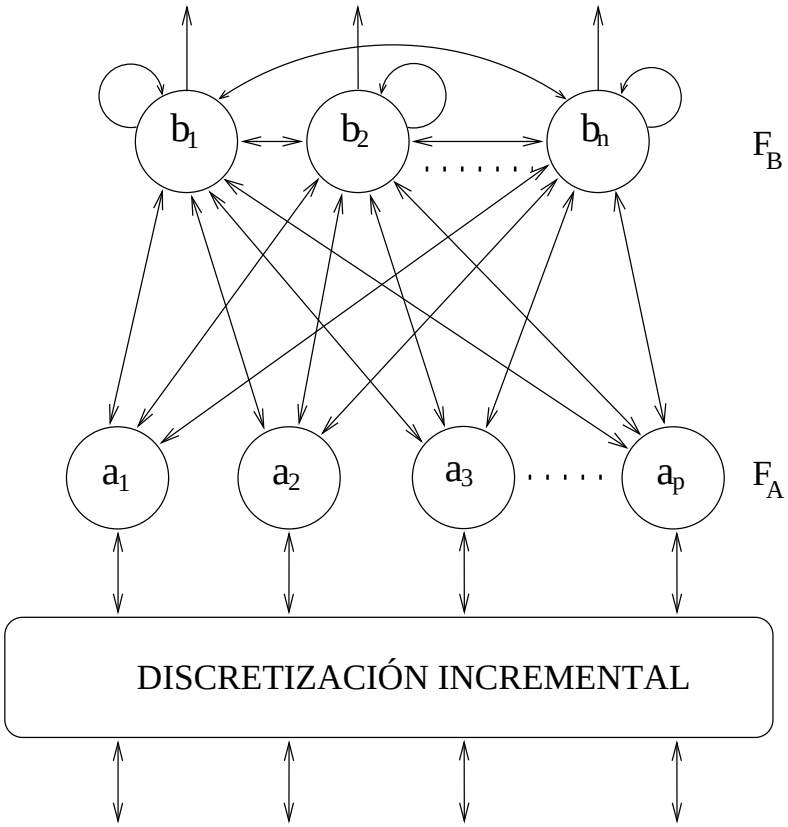
Los procesos de aprendizaje y recuperación de información diferirán de los de la CLAM en la incorporación del filtro de codificación por discretización incremental.

4.2.2 Aprendizaje

Una LCLAM es una memoria asociativa clasificadora a la que se accede a través de un filtro de codificación por discretización incremental. Este filtro es el encargado de discretizar la información difusa suministrada a la memoria y de decodificar la información discreta obtenida por la memoria para obtener la información difusa que representa.

En el proceso de aprendizaje podemos distinguir dos fases en el tratamiento de la información. Por un lado está la codificación de la información difusa y por otro lado está la memorización de los patrones discretos resultantes de la codificación. La memorización de los patrones discretos es la correspondiente al proceso de aprendizaje de la CLAM.

Figura 4.2: CLAM con filtro de codificación por discretización incremental



Para la memorización de un conjunto de patrones difusos se han de seguir los siguientes pasos:

1. Codificación de los patrones difusos mediante el método de discretización incremental

2. Aprendizaje en la CLAM de los patrones discretos obtenidos.

Antes de realizar la memorización de los patrones es necesario que se haya establecido la precisión con la que trabajará el método CDI.

4.2.3 Recuperación

En el proceso de recuperación de información se ha de tener en cuenta que los patrones recuperados se han de presentar al usuario en los mismos términos en que éste los memorizó, es decir, se debe obtener información difusa.

La recuperación de la información almacenada en la LCLAM seguirá tres pasos:

1. Codificación mediante el método de discretización incremental del patrón difuso que se presenta como entrada
2. A partir del patrón discreto obtenido por el proceso de codificación se realiza el proceso de recuperación en la CLAM para obtener un patrón discreto
3. Decodificación del patrón discreto recuperado para obtener la información difusa que representa

La codificación y decodificación de información necesita que se haya establecido previamente la precisión con la que se trabajará.

4.2.4 Ejemplos

Para ilustrar la memorización de información en la LCLAM se mostrarán ejemplos de almacenamiento de patrones y de reglas. El almacenamiento de reglas requiere un tratamiento especial porque se espera un comportamiento especial en la fase de recuperación. Dicho comportamiento consiste en la recuperación de un consecuente apropiado para los antecedentes presentados a la entrada mientras que el comportamiento normal sería el recuperar el patrón almacenado más semejante al de entrada.

4.2.4.1 Almacenamiento de patrones

Supongamos que queremos memorizar el siguiente conjunto de patrones:

	ALTURA					PESO				
PATRÓN	MB	B	N	A	MA	ML	L	N	P	MP
Antonio	0	0.1	0.9	0	0	0	0	0.9	0.1	0
Conchi	0	0	0.8	0.2	0	0	0.1	0.9	0	0
Bautista	0	0	0	0.8	0.2	0	0	0	0.7	0.3
David	0	0.1	0.9	0	0	0	0	0.4	0.6	0
Ana	0	0	0.5	0.5	0	0	0	0.8	0.2	0

La información difusa mostrada en la tabla se codificará mediante el método de discretización incremental usando una precisión de $\frac{1}{10}$. El resultado de la codificación de los patrones usando dicha precisión será:

	PESO				
	ML	L	N	P	MP
Antonio	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1+1-1	+1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1
Conchi	-1-1-1-1-1-1-1-1-1-1	+1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1+1-1	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1
Bautista	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1-1-1-1	+1+1+1-1-1-1-1-1-1-1
David	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1-1-1-1-1-1-1	+1+1+1+1+1+1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1
Ana	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1-1-1	+1+1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1

	ALTURA				
	MB	B	N	A	MA
Antonio	-1-1-1-1-1-1-1-1-1-1	+1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1+1-1	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1
Conchi	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1-1-1	+1+1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1
Bautista	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1-1-1	+1+1-1-1-1-1-1-1-1-1
David	-1-1-1-1-1-1-1-1-1-1	+1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1+1+1+1+1-1	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1
Ana	-1-1-1-1-1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1	+1+1+1+1+1-1-1-1-1-1	+1+1+1+1+1-1-1-1-1-1	-1-1-1-1-1-1-1-1-1-1

Los patrones discretos serán almacenados en una CLAM que tendrá 100 componentes en la capa de entrada (2 variables lin-

güísticas con 5 etiquetas cada una codificadas con 10 bits de precisión) y 5 componentes en la capa de salida (una por cada patrón almacenado).

Comprobemos ahora el resultado obtenido por la recuperación cuando se presenta en la entrada el patrón correspondiente a una persona ALTA de peso NORMAL.

El método de codificación por discretización incremental codificará la altura ALTA en el siguiente patrón:

$$
\begin{array}{c}
-1-1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1 \\
+1+1+1+1+1+1+1+1+1+1 \\
-1-1-1-1-1-1-1-1-1-1
\end{array}
$$

Del mismo modo, el peso NORMAL se codificará en el siguiente patrón:

$$
\begin{array}{c}
-1-1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1 \\
+1+1+1+1+1+1+1+1+1+1 \\
-1-1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1
\end{array}
$$

A continuación los patrones discretos obtenidos se presentan a la CLAM para realizar el proceso de recuperación. El patrón recuperado es el correspondiente a ANA. Las componentes recuperadas para la altura son:

$$
\begin{array}{c}
-1-1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1 \\
+1+1+1+1+1-1-1-1-1-1 \\
+1+1+1+1+1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1
\end{array}
$$

Las componentes recuperadas para el peso son:

$$
\begin{array}{c}
-1-1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1 \\
+1+1+1+1+1+1+1+1-1-1 \\
+1+1-1-1-1-1-1-1-1-1 \\
-1-1-1-1-1-1-1-1-1-1
\end{array}
$$

En este momento el filtro decodifica la información recuperada obteniendo como resultado una altura 0.5 normal, 0.5 alta y un peso 0.8 normal, 0.2 pesado.

4.2.4.2 Almacenamiento de reglas

Una regla es una asociación causa-efecto en que la causa la componen un conjunto de antecedentes y el efecto lo compone un consecuente.

El almacenamiento de reglas no sólo consiste en la memorización de patrones que codifiquen los antecedentes y consecuente de dichas reglas. Cuando almacenamos reglas, al suministrar un patrón como antecedente esperamos obtener un patrón consecuente que no tiene que ser necesariamente uno de los almacenados sino el resultado de la aplicación de todas las reglas que se disparen.

Para ilustrar el almacenamiento de un conjunto de reglas se usarán las del sistema de frenado de un móvil. Dicho conjunto de reglas establece la fuerza que es necesario aplicar a un móvil para frenarlo en función de la posición y velocidad del mismo.

1. Si posición positiva alta y velocidad cero entonces fuerza negativa alta.
2. Si posición positiva baja y velocidad positiva entonces fuerza negativa baja.
3. Si posición positiva baja y velocidad negativa entonces fuerza cero.
4. Si posición negativa alta y velocidad cero entonces fuerza positiva alta.
5. Si posición negativa baja y velocidad negativa entonces fuerza positiva baja.
6. Si posición negativa baja y velocidad positiva entonces fuerza cero.
7. Si posición cero y velocidad cero entonces fuerza cero.

La posición del móvil se expresa mediante una variable lingüística POSICIÓN cuyo valor se representa por los términos lingüísti-

cos negativa alta (NA), negativa baja (NB), cero (CE), positiva baja (PB) y positiva alta (PA).

La velocidad del móvil se expresa con los términos lingüísticos negativa (NE), cero (CE) y positiva (PO) de la variable lingüística VELOCIDAD.

La posición y velocidad del móvil constituyen el antecedente de las reglas de frenado del móvil. El consecuente lo forma la fuerza aplicable para frenarlo. La variable lingüística FUERZA se expresa mediante los términos lingüísticos negativa alta (NA), negativa baja (NB), cero (CE), positiva baja (PB) y positiva alta (PA).

La codificación de las reglas que queremos memorizar dará lugar a un conjunto de patrones en que unas componentes corresponderán a la causa y otras al efecto. Por ejemplo, usando 3 bits de precisión podemos realizar la siguiente codificación:

Si posición positiva alta y velocidad cero entonces fuerza negativa alta

$$\underbrace{\overbrace{-1-1-1,-1-1-1,-1-1-1,-1-1-1,+1+1+1}^{\text{Posición}}\ \overbrace{-1-1-1,+1+1+1,-1-1-1}^{\text{Velocidad}}}_{\substack{\text{CAUSA}\\ \text{Fuerza}}}$$

$$\underbrace{\overbrace{+1+1+1,-1-1-1,-1-1-1,-1-1-1,-1-1-1}^{\text{Fuerza}}}_{\text{EFECTO}}$$

La codificación bipolar de los patrones permite asignar el valor cero a las componentes desconocidas. Para la recuperación de un patrón almacenado, en el patrón de entrada se pondrán a cero las componentes desconocidas mientras que el resto conservará su valor. Por ejemplo, si queremos saber la fuerza necesaria que ha de aplicarse cuando el móvil se encuentra en una posición negativa baja con una velocidad negativa, el patrón de entrada será:

$$\underbrace{\overbrace{-1-1-1,+1+1+1,-1-1-1,-1-1-1,-1-1-1}^{\text{Posición}}\ \overbrace{+1+1+1,-1-1-1,-1-1-1}^{\text{Velocidad}}}_{\substack{\text{CAUSA}\\ \text{Fuerza}}}$$

$$\underbrace{\overbrace{000,000,000,000,000}^{\text{Fuerza}}}_{\text{EFECTO}}$$

Con dicha entrada, el patrón recuperado debe ser:

$$\overbrace{\overbrace{-1-1-1,+1+1+1,-1-1-1,-1-1-1,-1-1-1}^{\text{Posición}}\ \overbrace{+1+1+1,-1-1-1,-1-1-1}^{\text{Velocidad}}}^{\text{CAUSA}}$$

$$\overbrace{\overbrace{-1-1-1,-1-1-1,-1-1-1,+1+1+1,-1-1-1}^{\text{Fuerza}}}^{\text{EFECTO}}$$

Para mostrar la diferencia existente entre el almacenamiento de patrones y el almacenamiento de reglas veremos cómo no basta la simple memorización de las reglas codificadas porque cuando almacenamos reglas esperamos un comportamiento especial. Usando un único bit por etiqueta, la codificación de las reglas por discretización incremental resultante es:

	POSICIÓN					VELOCIDAD			FUERZA				
	NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
1	-1	-1	-1	-1	+1	-1	+1	-1	+1	-1	-1	-1	-1
2	-1	-1	-1	+1	-1	-1	-1	+1	-1	+1	-1	-1	-1
3	-1	-1	-1	+1	-1	+1	-1	-1	-1	-1	+1	-1	-1
4	+1	-1	-1	-1	-1	-1	+1	-1	-1	-1	-1	-1	+1
5	-1	+1	-1	-1	-1	+1	-1	-1	-1	-1	-1	+1	-1
6	-1	+1	-1	-1	-1	-1	-1	+1	-1	-1	+1	-1	-1
7	-1	-1	+1	-1	-1	-1	+1	-1	-1	-1	+1	-1	-1

La LCLAM se comporta como un clasificador por el vecino más próximo. Por ello, ante una entrada que no corresponde a ninguno de los patrones almacenados, busca el patrón memorizado más cercano entendiendo que las diferencias se deben a algún tipo de error que debe ser corregido. Esta característica dota de robustez a la memoria y es muy deseable en el modelo discreto pero en el caso lingüístico plantea algunos problemas.

Por ejemplo, si la entrada al sistema corresponde a una situación para la que no hay ninguna regla aplicable, esperamos que la memoria no recupere ninguna regla. Supongamos que la posición es negativa baja y la velocidad cero. El patrón que se presenta en

la entrada de la memoria es:

$$\overbrace{\overbrace{-1,+1,-1,-1,-1}^{\text{Pos.}}\ \overbrace{-1,+1,-1}^{\text{Vel.}}}^{\text{CAUSA}}\ \overbrace{\overbrace{0,0,0,0,0}^{\text{Fuerza}}}^{\text{EFECTO}}$$

Como no se ha memorizado ningún patrón que codifique esta situación, la LCLAM encuentra varios patrones almacenados que se hallan a igual distancia de Hamming del patrón de entrada, en concreto los correspondientes a las reglas 1, 4, 5, 6 y 7 y recupera uno de ellos siguiendo algún criterio que hayamos establecido previamente. Por ejemplo, dicho criterio puede establecer que en caso de igualdad se recupera el patrón que fue memorizado primero. Este no es el comportamiento deseado.

Queremos que no se recuperen patrones cuando no haya reglas aplicables. Para ello necesitamos indicarle a la LCLAM que no recupere patrones en los que las componentes correspondientes a la causa sean distintas a las indicadas en la entrada. Es decir, sólo se recuperará una regla si ésta es aplicable. Para lograr el comportamiento adecuado es conveniente tener claro el modo en que la CLAM mide las distancias entre patrones.

Las diferencias entre patrones se miden según la distancia de Hamming. La distancia existente entre dos patrones se incrementa en una unidad por cada componente en que difieran ambos.

Pares de patrones						Distancia
-1	$+1$	$+1$	$+1$	$+1$	-1	2
-1	$+1$	$+1$	-1	$+1$	$+1$	
$+1$	-1	-1	-1	$+1$	-1	0
$+1$	-1	-1	-1	$+1$	-1	
$+1$	$+1$	$+1$	$+1$	-1	$+1$	6
-1	-1	-1	-1	$+1$	-1	

Cuando desconozcamos el valor de alguna de las componentes de un patrón la representaremos con el valor cero de modo que la codificación de los patrones será ternaria:

- -1 representa ausencia de una característica
- $+1$ representa presencia de una característica
- 0 representa desconocimiento

Utilizando esta codificación ternaria, la LCLAM calcula la distancia existente entre dos componentes según se muestra en la siguiente tabla:

Comp.	Comp.	Distancia
1	-1	1
1	0	$\frac{1}{2}$
-1	0	$\frac{1}{2}$

Por tanto, cada componente desconocida en un patrón incrementa la distancia en $1/2$.

Pares de patrones						Distancia
$+1$	$+1$	-1	$+1$	-1	$+1$	$\frac{3}{2}$
$+1$	$+1$	-1	0	0	0	
-1	$+1$	-1	-1	$+1$	$+1$	3
-1	-1	0	0	0	0	
-1	$+1$	$+1$	$+1$	-1	$+1$	$\frac{11}{2}$
$+1$	-1	-1	-1	$+1$	0	

La memoria recibe como entrada el valor de las variables lingüísticas que corresponden a los antecedentes de las reglas. A partir de estos valores se recuperará el valor (lingüístico) del consecuente.

Si el valor del consecuente es desconocido, las componentes que lo codifican tomarán valor cero. Nuestra intención es que la memoria complete dichos valores desconocidos. Para ello debemos permitir que se recuperen patrones que se encuentren a una distancia

máxima igual a la distancia introducida por ese desconocimiento del consecuente. Es decir, si nuestros patrones tienen 20 componentes correspondientes al consecuente, la distancia introducida por su desconocimiento es $20 \times 1/2 = 10$.

Si, teniendo 20 componentes para indicar el consecuente, en la recuperación encontramos un patrón a una distancia superior a 10, podemos concluir que algunas componentes correspondientes al antecedente son distintas. En ese caso no se recuperaría dicho patrón porque corresponde a una regla no aplicable, es decir, corresponde a una regla con un antecedente distinto. Si el patrón de entrada no se encuentra a una distancia menor o igual a 10 de ninguno de los patrones almacenados no se recupera ninguno. Este es el comportamiento esperado cuando ninguna regla es aplicable.

En el caso que nos ocupa, como tenemos 5 componentes correspondientes al efecto, podemos marcar una distancia umbral de valor $5/2$ de tal modo que cualquier patrón que se halle a una distancia superior a dicho umbral no será recuperado.

Patrón de entrada	Patrón recuperado	Dist.
-1-1-1+1-1,+1-1-1,0 0 0 0 0	-1-1-1+1-1,+1-1-1,-1-1+1-1-1	$\frac{5}{2}$
+1-1-1-1-1,-1-1+1,0 0 0 0 0	0 0 0 0 0,0 0 0,0 0 0 0 0 No hay regla aplicable	$> \frac{5}{2}$

Con esto hemos logrado que la memoria presente el comportamiento deseado cuando codificamos las etiquetas lingüísticas usando un bit de precisión. No obstante, usando un sólo bit, no es posible utilizar grados de apreciación en las etiquetas. Por ejemplo, no podemos especificar una posición $1/4$ positiva baja, $3/4$ cero.

Si queremos usar grados de apreciación sobre las etiquetas necesitamos usar más de un bit de precisión. Si usamos 4 bits para la codificación, las reglas quedan expresadas según se muestra a continuación.

	POSICIÓN					VELOCIDAD			FUERZA				
	NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1
2	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1
3	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1
4	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1
5	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1
6	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1
7	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1

Para que sólo se recuperen los patrones correspondientes a reglas aplicables, se fija el umbral a un valor adecuado. En este caso, como tenemos 20 componentes correspondientes al efecto, podemos fijar el umbral a $20 \times 1/2 = 10$.

El sistema aún no se comporta todo lo bien que esperamos. Para mostrar los problemas que presenta vamos a ver un ejemplo.

Supongamos que queremos saber la fuerza que hay que aplicar ante una posición $1/4$ negativa alta, $3/4$ negativa baja y una velocidad $1/2$ negativa, $1/2$ cero. El patrón que codifica esta situación es:

$$\begin{array}{rc} \text{Posición} & +1-1-1-1,+1+1+1-1,-1-1-1-1,-1-1-1-1,-1-1-1-1 \\ \text{Velocidad} & +1+1-1-1,+1+1-1-1,-1-1-1-1 \end{array}$$

Al suministrar al sistema la entrada indicada obtenemos a la salida un patrón que muestra en las componentes correspondientes a la fuerza los siguientes valores:

$$\text{Fuerza} \quad -1-1-1-1,-1-1-1-1,-1-1-1-1,+1+1+1+1,-1-1-1-1$$

Es decir, devuelve la codificación de una fuerza positiva baja. En su lugar esperábamos obtener como respuesta una fuerza $1/4$ positiva alta, $1/2$ positiva baja porque la regla 4 se cumple en un grado $1/4$ dando lugar a una fuerza $1/4$ positiva alta y la regla 5 se cumple en un grado $1/2$ dando lugar a una fuerza $1/2$ positiva baja.

$$\text{Fuerza} \quad -1-1-1-1,-1-1-1-1,-1-1-1-1,+1+1-1-1,+1-1-1-1$$

Esta diferencia se debe a que los grados de apreciación distintos de 0000 y 1111 se interpretan como patrones erróneos que hay que corregir. Debemos tener en cuenta que la CLAM sólo recupera patrones que fueron previamente memorizados y no realiza ningún tipo de interpolación.

Si queremos que el sistema responda de forma correcta debemos realizar un entrenamiento acorde al grado de apreciación esperado. Por ejemplo, si queremos que ante una posición $1/4$ negativa media, $3/4$ negativa baja y una velocidad $1/2$ negativa baja, $1/2$ cero, el sistema responda con una fuerza $1/4$ positiva media, $1/2$ positiva baja, se debe realizar el entrenamiento con el patrón:

Posición $+1-1-1-1, +1+1+1-1, -1-1-1-1, -1-1-1-1, -1-1-1-1$
Velocidad $+1+1-1-1, +1+1-1-1, -1-1-1-1$
Fuerza $-1-1-1-1, -1-1-1-1, -1-1-1-1, +1+1-1-1, +1-1-1-1$

Esto implica que cada regla dará lugar a un conjunto de patrones que represente todas las posibilidades que se pueden presentar como entrada al sistema en las que dicha regla es aplicable.

Por ejemplo tomemos la regla "si posición positiva baja y velocidad positiva entonces fuerza negativa baja". Para que esta regla sea aplicable es necesario que la posición de entrada sea positiva baja en un grado mayor que cero. De igual modo, la velocidad ha de ser positiva en un grado superior a cero. Las distintas situaciones en que se cumplen esos requisitos para la posición y la velocidad se muestran en las siguientes tablas:

POSICIÓN				
NA	NB	CE	PB	PA
–1–1–1–1	–1–1–1–1	+1+1+1–1	+1–1–1–1	–1–1–1–1
–1–1–1–1	–1–1–1–1	+1+1–1–1	+1+1–1–1	–1–1–1–1
–1–1–1–1	–1–1–1–1	+1–1–1–1	+1+1+1–1	–1–1–1–1
–1–1–1–1	–1–1–1–1	–1–1–1–1	+1+1+1+1	–1–1–1–1
–1–1–1–1	–1–1–1–1	–1–1–1–1	+1+1+1–1	+1–1–1–1
–1–1–1–1	–1–1–1–1	–1–1–1–1	+1+1–1–1	+1+1–1–1
–1–1–1–1	–1–1–1–1	–1–1–1–1	+1–1–1–1	+1+1+1–1

VELOCIDAD		
NE	CE	PO
–1–1–1–1	+1+1+1–1	+1–1–1–1
–1–1–1–1	+1+1–1–1	+1+1–1–1
–1–1–1–1	+1–1–1–1	+1+1+1–1
–1–1–1–1	–1–1–1–1	+1+1+1+1

Por tanto la regla será aplicable en cada una de las 28 situaciones resultantes al combinar las 7 posibilidades de la posición con cada una de las 4 posibilidades de la velocidad.

En los 28 patrones resultantes a partir de esa regla hay que codificar la fuerza resultante. Por ejemplo, para el patrón que codifica la posición $3/4$ positiva baja, $1/4$ positiva media y velocidad $1/2$ cero, $1/2$ positiva baja, hay que codificar una fuerza $1/4$ negativa media, $1/2$ negativa baja.

Posición $-1-1-1-1, -1-1-1-1, -1-1-1-1, +1+1+1-1, +1-1-1-1$
Velocidad $-1-1-1-1, +1+1-1-1, +1+1-1-1$
Fuerza $+1-1-1-1, +1+1+1-1, -1-1-1-1, -1-1-1-1, -1-1-1-1$

En las figuras 4.3 a 4.11 se muestran los patrones que será necesario memorizar para obtener una memoria que proporciona una respuesta correcta ante las entradas que se presentan al sistema.

Figura 4.3: Reglas a memorizar cuando la velocidad es negativa

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	-++++	-++++
+++++	+++++	-++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
+++++	+++++	-++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
+++++	+++++	-++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
-++++	+++++	-++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
-++++	+++++	+++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
-++++	+++++	+++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
-++++	+++++	+++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	+++++	-++++
-++++	-++++	+++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	-++++	-++++
-++++	-++++	+++++	-++++	-++++	+++++	-++++	-++++	-++++	-++++	-++++	-++++	-++++
-++++	-++++	+++++	+++++	-++++	+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++
-++++	-++++	+++++	+++++	-++++	+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++
-++++	-++++	-++++	+++++	-++++	+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++
-++++	-++++	-++++	+++++	-++++	+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++
-++++	-++++	-++++	+++++	+++++	+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++
-++++	-++++	-++++	+++++	+++++	+++++	-++++	-++++	-++++	-++++	+++++	-++++	-++++

Figura 4.4: Reglas a memorizar cuando la velocidad es 0.75 negativa, 0.25 cero

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1+1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1
+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1−1−1−1
+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1−1−1−1
+1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1
−1−1−1−1	+1+1+1+1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	−1−1−1−1
−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1+1−1	−1−1−1−1
−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1−1−1	−1−1−1−1
−1−1−1−1	+1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1+1+1+1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1+1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1+1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1+1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1+1	+1+1+1−1	+1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1

Figura 4.5: Reglas a memorizar cuando la velocidad es 0.5 negativa, 0.5 cero

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1+1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1
+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1−1−1
+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1
+1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1−1−1−1
−1−1−1−1	+1+1+1+1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1
−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1−1−1	−1−1−1−1
−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1
−1−1−1−1	+1−1−1−1	+1+1+1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1+1+1+1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1+1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1+1	−1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1−1	+1−1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1−1−1	+1+1−1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	+1−1−1−1	+1+1+1−1	+1+1−1−1	+1+1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	+1−1−1−1	−1−1−1−1	−1−1−1−1
−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	+1+1+1+1	+1+1−1−1	+1+1−1−1	−1−1−1−1	+1+1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1	−1−1−1−1

Figura 4.6: Reglas a memorizar cuando la velocidad es 0.25 negativa, 0.75 cero

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1
+1+1+1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1-1
+1+1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1-1-1
+1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1-1-1-1
-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1
-1-1-1-1	+1+1+1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1-1-1-1	-1-1-1-1
-1-1-1-1	+1+1+1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1	+1-1-1-1	-1-1-1-1
-1-1-1-1	+1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1	+1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1+1+1-1	+1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1+1-1-1	+1+1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1-1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1	+1-1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	+1-1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1	+1+1-1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1-1	+1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	+1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1

Figura 4.7: Reglas a memorizar cuando la velocidad es cero

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1
+1+1+1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1
+1+1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1
+1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1
-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	+1+1+1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	+1+1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	+1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1+1+1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1+1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1-1	+1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1-1-1	+1+1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	+1-1-1-1	+1+1+1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1
-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	+1+1+1+1	-1-1-1-1	-1-1-1-1	-1-1-1-1	-1-1-1-1

Figura 4.8: Reglas a memorizar cuando la velocidad es 0.75 cero, 0.25 positiva

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1
+1+1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	+1+1-1
+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	+1+1-1
+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	+1-1-1
-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	+1-1-1	-1-1-1	+1+1-1	+1-1-1	+1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1-1-1	+1+1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	+1+1+1	-1-1-1	+1+1-1	+1-1-1	+1+1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	+1-1-1	-1-1-1	+1+1-1	+1-1-1	+1+1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	+1+1-1	-1-1-1	+1+1-1	+1-1-1	+1+1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	+1+1+1	-1-1-1	+1+1-1	+1-1-1	+1+1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1

Figura 4.9: Reglas a memorizar cuando la velocidad es 0.5 cero, 0.5 positiva

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1
+1+1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	+1+1-1
+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	+1+1-1
+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	+1-1-1
-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	+1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1-1	+1+1+1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	+1-1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1-1	+1+1+1	+1+1+1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1

Figura 4.10: Reglas a memorizar cuando la velocidad es 0.25 cero, 0.75 positiva

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1
+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	+1-1-1
+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	+1-1-1
+1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	+1-1-1
-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1
-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1
-1-1-1	+1+1+1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	+1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	+1+1-1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	+1+1+1	-1-1-1	-1-1-1	+1-1-1	+1+1+1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	+1-1-1	+1+1+1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	+1+1+1	+1+1-1	-1-1-1	+1-1-1	+1+1+1	+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	+1+1+1	+1+1+1	-1-1-1	+1-1-1	+1+1+1	+1-1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	+1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1

Figura 4.11: Reglas a memorizar cuando la velocidad es positiva

POSICIÓN					VELOCIDAD			FUERZA				
NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1
+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	-1-1-1
+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
+1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1
-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1
-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	+1+1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	+1-1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	+1+1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	+1+1-1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	+1+1+1	+1+1+1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	+1+1+1	+1-1-1	+1+1+1	-1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	+1+1+1	+1+1-1	-1-1-1	-1-1-1	+1+1+1	+1-1-1	+1+1-1	-1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	+1+1+1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	+1-1-1	+1-1-1	-1-1-1	-1-1-1	-1-1-1
-1-1-1	-1-1-1	-1-1-1	-1-1-1	+1+1+1	-1-1-1	-1-1-1	+1+1+1	+1-1-1	-1-1-1	-1-1-1	-1-1-1	-1-1-1

4.3 Construcción de una CLAM continua para el almacenamiento de información imprecisa

En la sección anterior, mediante el empleo de CDI hemos construido una máquina soportada por una CLAM, capaz de procesar información difusa. Ahora bien, existe otra alternativa para obtener una tal memoria asociativa trabajando directamente con valores continuos.

Así, a partir de la CLAM surge la Memoria Asociativa Clasificadora Continua CCLAM para la memorización de patrones cuyas componentes toman valores en el intervalo $[0, 1]$. La CCLAM permite el almacenamiento y recuperación efectivos de información difusa (atributos o reglas) al almacenar información imprecisa expresada en términos lingüísticos mediante la memorización de patrones formados por los grados de compatibilidad con dichos términos. Las características de la CCLAM son, en cierta medida, similares a las de la CLAM, dado lo cual prestaremos especial atención a las características diferenciales específicas.

4.3.1 Memoria asociativa clasificadora continua

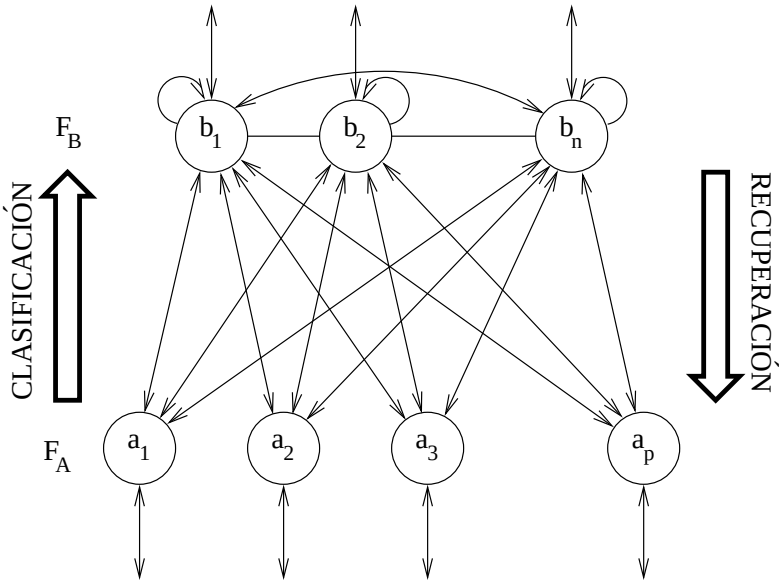
La CCLAM almacena patrones espaciales arbitrarios que tendrán componentes continuas en el intervalo $[0, 1]$. La topología de la CCLAM (fig. 4.12) se diferencia de la CLAM en que todos los EP son continuos en lugar de bipolares y binarios, no están umbralizados y su función de activación es:

$$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } 0 \leq x \leq 1 \\ 1 & \text{si } x > 1 \end{cases} \tag{4.3}$$

En la capa F_A hay tantos EP como componentes tengan los patrones a almacenar. En esta capa se presentarán los patrones de entrada y se recuperarán los patrones almacenados.

Los EP de la capa F_B representan a los patrones almacenados. Su número es variable y cambia dinámicamente con el aprendizaje, de tal modo que, al igual que en la CLAM discreta, siempre habrá

Figura 4.12: Memoria Asociativa Clasificadora Continua (CCLAM)



tantos EP en la capa F_B como patrones almacenados. El estado de activación de los mismos indica el grado en que se reconoce el patrón asociado a cada uno de ellos.

En el funcionamiento de la CCLAM podemos distinguir dos procesos: clasificación y recuperación. El proceso de clasificación recibe en la capa F_A un patrón de entrada y obtiene en cada EP de F_B un valor que indica el grado en que el patrón de entrada se asemeja al patrón almacenado representado por el EP de F_B. El proceso de recuperación recibe en F_B los grados en que hacemos referencia a cada una de los patrones almacenados y en F_A recupera una combinación de dichos patrones.

La capa F_B es competitiva. Los pesos de las señales de competición entre los EP de la capa F_B se representan en la matriz $\beta_{n\times n}$, siendo n el número de patrones almacenados y β_{ij} el peso de la conexión existente desde b_i hasta b_j. El peso de la conexión indica el grado de competición de modo que una conexión desde el EP i hasta el j con peso cero indica que el elemento i no compite con el

j; si el peso es uno, el elemento i compite con el j con un grado máximo indicado por su estado de activación. Si todas las conexiones de la competición tienen peso unitario obtenemos una competición en que se da la misma importancia a todos los patrones; si todos los pesos valen cero anulamos la competición.

Para la competición podemos establecer si el EP ganador conservará su estado de activación o tomará como nuevo estado el valor 1. Sea Y_i el estado de activación del EP B_i. Si tras la competición el elemento ganador conserva su estado, el valor de activación de B_i será:

$$\begin{cases} Y_i & \text{si } Y_i \geq Y_j \beta_{ji} \quad \forall j \\ 0 & \text{en otro caso .} \end{cases} \tag{4.4}$$

En caso de que el ganador tome el valor 1, el nuevo valor de B_i será:

$$\begin{cases} 1 & \text{si } Y_i \geq Y_j \beta_{ji} \quad \forall j \\ 0 & \text{en otro caso .} \end{cases} \tag{4.5}$$

En caso de que haya más de un elemento ganador se seleccionará alguno de ellos siguiendo algún criterio establecido y se desactivarán los demás.

La competición que se establece en F_B puede ser distinta en las fases de clasificación y recuperación. β_{ij}^C es el peso de la competición de i hacia j en el proceso de clasificación; β_{ij}^R es el peso de la competición de i hacia j en el proceso de recuperación.

En el proceso de clasificación, en F_A presentamos un patrón de entrada y en la capa F_B obtenemos el grado de compatibilidad existente entre cada uno de los patrones almacenados y el patrón de entrada o bien encontramos sólo activo el representante del patrón que tiene mayor compatibilidad con el patrón de entrada. Esto depende del tipo de competición que se establezca.

En el proceso de recuperación, en F_B presentamos los grados en que hacemos referencia a cada uno de los patrones almacenados y recuperamos en F_A una combinación de dichos patrones.

4.3.2 Aprendizaje

En el proceso de aprendizaje, la memorización de un patrón lleva asociada la creación de un nuevo EP en F_B y el ajuste conveniente de los pesos de las nuevas conexiones que se realizan hacia él.

El aprendizaje tiene los siguientes pasos:

1. Llegada a F_A de un patrón que se quiere memorizar.
2. Comprobación de que el patrón no fue previamente memorizado.
3. Creación de un nuevo EP en F_B asociado al patrón.
4. Ajuste de los pesos de competición de la capa F_B.
5. Ajuste de los pesos de F_A hacia el nuevo EP.

4.3.2.1 Comprobación de que el patrón no fue previamente memorizado

Para comprobar que el patrón no está ya almacenado en la memoria se realiza el proceso de clasificación con dicho patrón y se comprueba el resultado obtenido.

El proceso de clasificación permite especificar unas funciones Υ^C y Ψ^C apropiadas. Para la comprobación de la previa memorización de patrón se usarán las funciones:

$$\begin{array}{l} \Upsilon^C(x,y) = 1 - |x-y| \\ \Psi^C = \text{media aritmética.} \end{array} \tag{4.6}$$

Con esta configuración de funciones se compara componente a componente el patrón de entrada con cada uno de los patrones almacenados. La señal recibida por los EP de la capa F_B sólo puede valer uno si el patrón de entrada y el patrón almacenado coinciden.

En la capa F_B hacemos que en la competición todos los pesos valgan uno y que el elemento ganador conserve su estado de activación. De este modo, si tras el proceso de clasificación queda algún elemento con valor uno indica que el patrón de entrada coincide con uno de los almacenados y, por tanto, no se debe repetir el aprendizaje del mismo.

4.3.2.2 Ajuste de los pesos de competición de la capa F_B

En la capa F_B el peso de la conexión (i, j) indica el grado en que la señal de activación del EP i se compara con la del j. Si el peso de la conexión es uno se comparan los dos estados de activación de los EP conectados; si el peso es cero quiere decir que no hay competición entre los EP conectados; otros valores intermedios indican distintos grados de influencia.

Como la capa F_B es dinámica, por cada patrón almacenado hay que asignar los pesos de las conexiones que van hasta el nuevo EP que representa al patrón.

El peso de la conexión de competición entre dos EP puede verse como la importancia relativa que se da a los dos patrones representados por los EP conectados. Generalmente encontraremos con peso $\beta_{ij} = 1$ todas las conexiones de competición para reflejar que se da la misma importancia a todos los patrones.

Los pesos establecidos para la competición en el proceso de clasificación pueden ser distintos de los del proceso de recuperación. β_{ij}^C será el peso de la competición que existe entre los EP i y j durante la clasificación y β_{ij}^R el peso durante la recuperación.

4.3.2.3 Ajuste de los pesos de F_A hacia el nuevo EP

El número de EP's de la capa F_B es dinámico, de tal modo que cuando se almacena un nuevo patrón se crea un nuevo EP en la capa F_B asociado a dicho patrón.

Cuando se almacena el patrón $X = (x_1, \ldots, x_p)$, a los pesos de las conexiones de los EP de la capa F_A hacia el nuevo EP b_p de la capa F_B se asigna el valor $m_{pi} = x_i$.

Una memoria que ha almacenado p patrones de n componentes cada uno tendrá asociada una matriz de pesos $M_{p\times n}$ que representa los pesos de las conexiones entre los EP de las capa F_A y F_B.

$$M_{n\times p} = \begin{pmatrix} m_{11} & \ldots & m_{1p} \\ \vdots & \ddots & \vdots \\ m_{n1} & \ldots & m_{np} \end{pmatrix}. \tag{4.7}$$

Por tanto, el almacenamiento del conjunto de patrones $X_1, \ldots, X_n$, siendo $X_i = (x_{i1}, \ldots, x_{ip})$ llevará a la matriz de pesos:

$$M = \begin{pmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \ldots & x_{np} \end{pmatrix}. \tag{4.8}$$

Por cada nuevo patrón almacenado se añade una fila a la matriz de pesos con los pesos de las conexiones de los EP de la capa F_A hacia el nuevo EP que representa al nuevo patrón.

Los pesos de las conexiones entre las dos capas en el proceso de clasificación pueden ser distintos de los del proceso de recuperación. Para diferenciarlos llamaremos M^C a la matriz de pesos en el proceso de clasificación y M^R a la matriz de pesos en el proceso de recuperación.

4.3.3 Clasificación

En el proceso de clasificación los EP envían señales de activación desde la capa F_A hacia la capa F_B. Ante un patrón de entrada presentado en la capa F_A se obtienen en la capa F_B unos estados de activación que representan la semejanza existente entre el patrón de entrada y cada uno de los patrones almacenados.

El proceso de clasificación tiene los siguientes pasos:

1. A la capa F_A llega un patrón $X = (x_1, \ldots, x_p)$.

2. La señal se propaga hacia la capa F_B de modo que el EP b_i recibe la señal se activación:

   (a)
   $$\begin{aligned} y_i &= \Psi_j^C \left\{ \Upsilon^C \left(x_i, m_{ij}^C \right) \right\} = \\ &= \Psi^C \left(\Upsilon^C \left(x_1, m_{1j}^C \right), \Upsilon^C \left(x_2, m_{2j}^C \right), \ldots, \Upsilon^C \left(x_n, m_{nj}^C \right) \right). \end{aligned}$$

   (b) Υ^C y Ψ^C son funciones acotadas al intervalo $[0, 1]$.

3. Los EP de la capa F_B compiten entre sí con la intensidad reflejada en la matriz de pesos β^C.

4. En la capa F_B obtenemos el resultado del proceso de clasificación. El estado de activación obtenido en los EP de la capa F_B indica el grado en que el patrón de entrada se clasifica por cada una de las clases representadas por los patrones almacenados.

4.3.4 Recuperación

En el proceso de recuperación el flujo de información es opuesto al del proceso de clasificación. En la capa F_B se presenta un patrón de entrada que representa el grado en que la recuperación implica a cada uno de los patrones almacenados. La recuperación tiene los siguientes pasos:

1. A la capa F_B llega un patrón $Y = (y_1, \ldots, y_n)$.
2. Los EP de la capa F_B compiten entre sí con la matriz de pesos β^R. Los nuevos estados de activación forman el patrón $Y' = (y'_1, \ldots, y'_n)$.
3. La señal se propaga desde la capa F_B hacia la capa F_A de modo que el EP a_i recibe la señal de activación:

   (a)
   $$\begin{aligned} x_i &= \Psi_j^R \left\{ \Upsilon^R \left(y'_j, m_{ji}^R \right) \right\} = \\ &= \Psi^R \left(\Upsilon^R \left(y'_1, m_{1i}^R \right), \Upsilon^R \left(y'_2, m_{2i}^R \right), \ldots, \Upsilon^R \left(y'_p, m_{pi}^R \right) \right). \end{aligned}$$

   (b) Υ^R y Ψ^R son funciones acotadas al intervalo $[0, 1]$.

4. En la capa F_A obtenemos el patrón resultante del proceso de recuperación. Éste puede ser uno de los patrones almacenados o una combinación de ellos.

4.3.5 Almacenamiento de distintos tipos de información

La CCLAM posee la suficiente flexibilidad como para lograr un comportamiento aceptable ante distintas necesidades de almacenamiento y recuperación.

A continuación vamos a estudiar su empleo en distintas situaciones. Comenzaremos por el caso más simple, aquel en que se

memorizan los valores lingüísticos de una única variable. Para casos más complejos (existencia de varias variables o almacenamiento de reglas difusas) podemos emplear la CCLAM como elemento para construir estructuras de mayor nivel de complejidad capaces de gestionarlos.

4.3.5.1 Almacenamiento de etiquetas de una variable lingüística

Este caso se presenta cuando queremos almacenar el valor de un sólo atributo de un cierto tipo de objetos que expresaremos por medio de etiquetas lingüísticas. En esta situación nos encontramos cuando, por ejemplo, queremos almacenar la altura de un conjunto de individuos.

Para el almacenamiento de las etiquetas lingüísticas utilizaremos una CCLAM. El aprendizaje de los patrones siempre se realiza del mismo modo con la excepción del ajuste de los pesos de la competición que, por claridad, se indicará al mostrar el proceso de recuperación. En cuanto a la recuperación de la información es necesario especificar las funciones de propagación de información entre las capas F_A y F_B y los pesos de las conexiones de la competición en F_B tanto para el proceso de clasificación como para el de recuperación. El objetivo es recuperar el patrón almacenado más próximo al patrón presentado como entrada y, para ello, las fases de clasificación y recuperación tendrán el comportamiento que se muestra a continuación.

Clasificación

La comparación que se realiza entre los patrones almacenados y el patrón de entrada por medio de las funciones Υ^C y Ψ^C debe indicarnos el grado en que se parecen ambos patrones. Para ello usaremos como función Υ^C una expresión que sólo de el resultado uno cuando las componentes operadas sean idénticas y que, en el resto de los casos, devuelva un valor comprendido entre cero y uno que indique la semejanza existente. Para ello se seleccionará la $\Upsilon^C(x, y) = 1 - |x - y|$.

La función Ψ^C es la encargada de aunar la información particular obtenida de las componentes por lo que se seleccionará como función Ψ^C la media aritmética. La media aritmética sumará los grados de semejanza obtenidos por cada componente para posteriormente normalizar el resultado al intervalo $[0, 1]$ al dividir entre el número de componentes.

En la capa F_B recibimos procedente de F_A la media de los grados de semejanza existente entre las componentes del patrón. El EP con mayor estado de activación será aquel que representa al patrón almacenado más cercano al de entrada.

Como se trata de recuperar el patrón almacenado que sea más parecido al de entrada, en la competición de F_B el elemento ganador tomará el estado de activación uno mientras que los demás permanecerán inactivos. En este momento tenemos en la capa F_B un sólo EP activo que será el representante del patrón que queremos recuperar. Con esto se logra que en el proceso de recuperación no interfieran entre sí los patrones almacenados y se recupere el patrón deseado.

Recuperación

En el proceso de recuperación no se debe establecer competición en F_B porque ya la ha habido al final del proceso de clasificación. Por tanto, los pesos β_{ij}^R que se establecen son todos nulos.

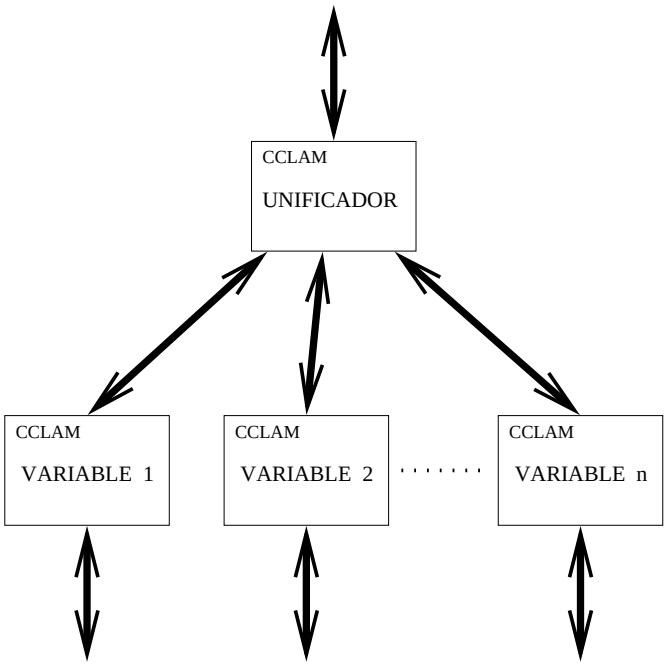
La función Υ^R que se usará para transmitir la información desde F_B hasta F_A será la t-norma del mínimo. Con esto se logra que todos los EP inactivos de F_B envíen señales de activación nulas. Estableciendo la t-conorma del máximo como función Ψ^R, el estado de activación final de los elementos de F_A será el recibido procedente del elemento representante del patrón que se recuperará.

4.3.5.2 Almacenamiento de etiquetas de más de una variable lingüística

Si queremos almacenar patrones que codifiquen diversas características de un conjunto de objetos necesitaremos emplear varias variables lingüísticas para describirlos. Para el almacenamiento de

información referente a más de una variable lingüística se usará la memoria asociativa clasificadora continua como elemento estructural de tal modo que tendremos una CCLAM por cada variable lingüística más otra CCLAM que unificará toda la información.

Figura 4.13: Almacenamiento de etiquetas de más de una variable lingüística



Cada CCLAM asociada a una variable lingüística tendrá tantos EP en F_A como etiquetas lingüísticas definidas sobre la variable y tantos EP en F_B como patrones memorizados. La CCLAM unificadora tendrá en F_B tantos EP como patrones memorizados. En F_A habrá por cada variable lingüística tantos EP como patrones almacenados.

El funcionamiento es una extensión del logrado en el caso de una sola variable lingüística. Para cada variable lingüística computaremos la semejanza existente entre el patrón de entrada y el

patrón almacenado y la CCLAM unificadora aunará toda la información para determinar cuál es el patrón que se recuperará. El comportamiento del sistema en las fases de clasificación y recuperación tanto en las CCLAM asociadas a variables lingüísticas como en la CCLAM unificadora será el mostrado a continuación.

Clasificación

En el proceso de clasificación se determina, a partir del patrón de entrada, cuál es el patrón almacenado que se debe recuperar por ser aquél con el que guarda más semejanza. Todo este proceso se divide en dos fases.

La primera fase tiene lugar en las CCLAM asociadas a las variables lingüísticas que representan los atributos de los patrones. En cada memoria se determinará la semejanza existente en el valor de la variable entre el patrón de entrada y cada uno de los almacenados.

En la segunda fase, desarrollada en el unificador, a partir de las semejanzas individuales obtenidas en la primera fase, se obtiene la semejanza global que existe entre el patrón de entrada y los patrones memorizados. El patrón con el que guarde más semejanza será el patrón seleccionado.

CCLAM asociada a variable lingüística. Cada CCLAM asociada a una variable lingüística debe computar la semejanza existente entre el patrón de entrada y los patrones almacenados del mismo modo que en el caso de una sola variable lingüística. La diferencia se encuentra en que en este caso no habrá competición ya que el objetivo de la competición es seleccionar el EP que representa al patrón más cercano y esta tarea la realiza el unificador tras recibir la información del resto de CCLAM.

Por tanto los pesos de la competición se anulan todos $\beta_{ij}^C = 0 \quad \forall i, j$ para que no se realice competición y cada EP conserve su estado.

El patrón de entrada se propagará desde F_A hasta F_B mediante las funciones Υ^C y Ψ^C. Al igual que ocurría en el caso del almacenamiento de una sola variable lingüística la función Ψ^C será la

media aritmética y $\Upsilon^C(x,y) = 1 - |x - y|$.

Unificador La CCLAM que unifica la información recibe por cada EP de su capa F_A el estado de uno de los EP de la capa F_B de las CCLAM asociadas a las variables lingüísticas.

Sea w_{ijk} el peso de la conexión que va desde el EP de F_A que recibe su estado de activación del elemento representante del patrón j-ésimo de la CCLAM asociada a la variable lingüística i-ésima y llega hasta el EP asociado al patrón k-ésimo de F_B. Los pesos se asignan del siguiente modo:

$$w_{ijk} = \begin{cases} 1 & \text{si } j = k \\ 0 & \text{en otro caso .} \end{cases} \tag{4.9}$$

Es decir, EP asociados al mismo patrón tienen peso uno mientras que el resto tiene peso cero. Con esto se consigue que cada EP de F_B sólo reciba información de los EP de F_A asociados al mismo patrón.

Para la propagación de las señales desde la capa F_A hasta la capa F_B se usará como función Υ^C el producto. Dada la asignación de pesos de las conexiones entre las capas, con el producto logramos que cada EP de la capa F_B reciba sólo las señales procedentes de los elementos asociados con el mismo patrón. Como función Ψ^C se usará la media aritmética.

El estado de activación de los EP de F_B será proporcional a la semejanza existente entre el patrón de entrada y el patrón almacenado correspondiente. Estableciendo una competición en que el elemento ganador tome valor uno logramos que el único EP activo de F_B sea el representante del patrón almacenado más semejante al de entrada. Para ello daremos la misma importancia a todos los elementos de la competición con el ajuste apropiado de los pesos $\beta_{ij}^C = 1 \quad \forall i, j$.

Recuperación

El proceso de recuperación comienza cuando ya se ha determinado cuál es el patrón almacenado que se va a recuperar. El

unificador se limitará a transmitir la selección a las CCLAM asociadas a las variables para que éstas recuperen la información del patrón seleccionado.

Unificador En la recuperación el unificador transmite la selección realizada a las CCLAM asociadas a las variables.

La competición se anula porque los elementos ya han competido al final de la fase de clasificación y sólo uno de ellos se encuentra activo. Por tanto, los pesos de la competición se establecen a cero $\beta_{ij}^R = 0 \quad \forall i, j$.

Los pesos de las conexiones que se establecen entre las capas F_A y F_B son iguales a los establecidos en la fase de clasificación, es decir, los EP asociados al mismo patrón están unidos por conexiones con peso uno mientras que los demás están unidos por conexiones con peso nulo.

Usando como función Υ^R el producto, cada elemento de F_A recibirá información procedente sólo de los EP asociados al mismo patrón. De este modo, los elementos de F_A asociados al patrón seleccionado recibirán una señal activa procedente de F_B. Los elementos asociados a patrones no seleccionados recibirán todas las señales a cero. Usando el máximo como función Ψ^R logramos que sólo los elementos de F_A asociados al patrón a recuperar permanezcan activos.

CCLAM asociada a variable lingüística. Cada CCLAM asociada a una variable lingüística recibe del unificador una señal que deja en su capa F_B un único EP activo. Dicho elemento, que tiene estado uno, es el que representa al patrón a recuperar. Por tanto no es necesario establecer una competición entre los EP de F_B de modo que todos los pesos se pondrán a cero $\beta_{ij}^R = 0 \quad \forall i, j$.

Las funciones Υ^R y Ψ^R que se usarán son la t-norma del mínimo y la t-conorma del máximo respectivamente. Con esto se logra que todos los EP inactivos de F_B envíen señales de activación nulas y que el estado de activación final de los elementos de F_A sea el recibido procedente del elemento representante del patrón que se recuperará.

4.3.5.3 Almacenamiento de reglas difusas

Cuando el tipo de información que necesitamos almacenar está formada por reglas difusas esperamos un comportamiento diferente al obtenido al almacenar atributos de objetos.

El almacenamiento de reglas implica la memorización de pares de patrones (antecedente, consecuente) mientras que el almacenamiento de información de objetos implica la memorización de patrones individuales.

Cuando almacenamos atributos de objetos deseamos que el sistema recupere el patrón almacenado más cercano al patrón de entrada. Cuando almacenamos reglas al suministrar un patrón como antecedente esperamos obtener un patrón consecuente que no tiene que ser necesariamente uno de los almacenados sino el resultado de la aplicación de todas las reglas que se disparen.

Para el almacenamiento de reglas dispondremos de una CCLAM por cada una de las variables lingüísticas del antecedente y otra CCLAM para el consecuente. Las memorias asociadas al antecedente recibirán como entrada un patrón que determinará el estado del sistema sobre el que se aplicarán las reglas. Cada una de estas memorias calculará el grado con que el valor de la variable lingüística que representa provoca el disparo de cada una de las reglas. Una CCLAM adicional unificará toda la información hallando el grado total con que se dispara cada regla. Esta información es entonces enviada a la memoria que almacena los consecuentes para que se recupere la combinación apropiada de éstos.

Podemos ver este esquema de una forma más detallada. Supongamos que las variables lingüísticas $A_1, A_2, \ldots, A_n$ forman el antecedente y que el consecuente lo forma la variable lingüística C.

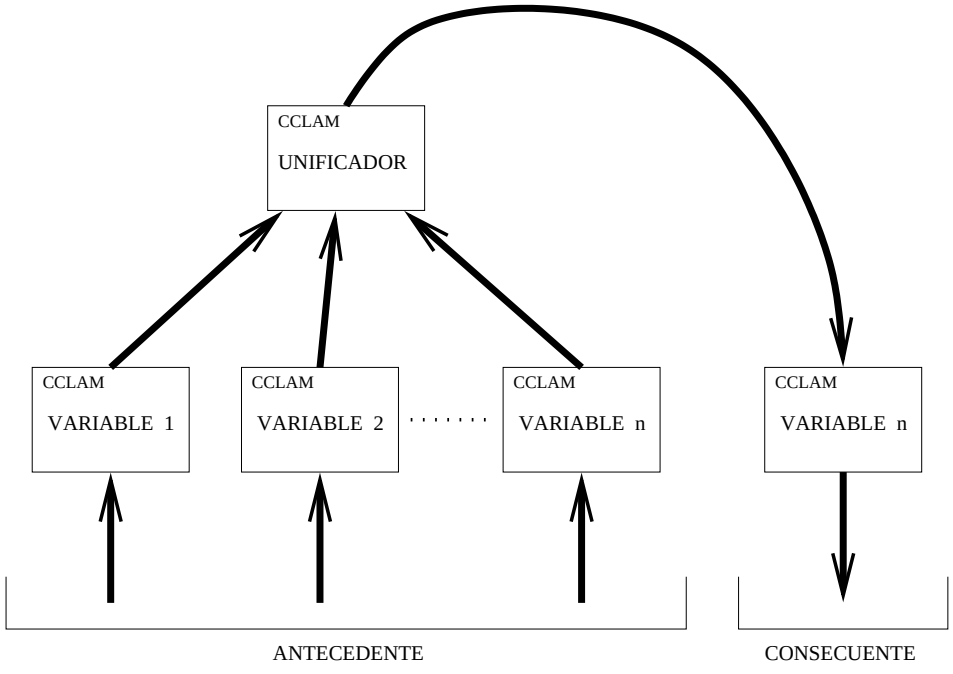
Los valores que toma la variable A_i se expresan mediante el conjunto de q_i términos lingüísticos $t_{i1}, t_{i2}, \ldots, t_{iq_i}$.

Los términos que expresan el valor de la variable C son $c_1, c_2, \ldots, c_m$.

El conjunto de reglas que se almacenarán son $R_1, R_2, \ldots, R_p$, donde cada regla es de la forma $a_1 \wedge a_2 \wedge \ldots \wedge a_n \Rightarrow b$, siendo:

$$\begin{array}{c} a_j \in \left\{t_{j1}, \ldots, t_{jq_j}\right\} \\ b \in \left\{c_1, \ldots, c_m\right\}. \end{array} \tag{4.10}$$

Figura 4.14: Almacenamiento de reglas usando CCLAM



El sistema de CCLAM que almacenará el conjunto de reglas será el mostrado en la figura 4.15.

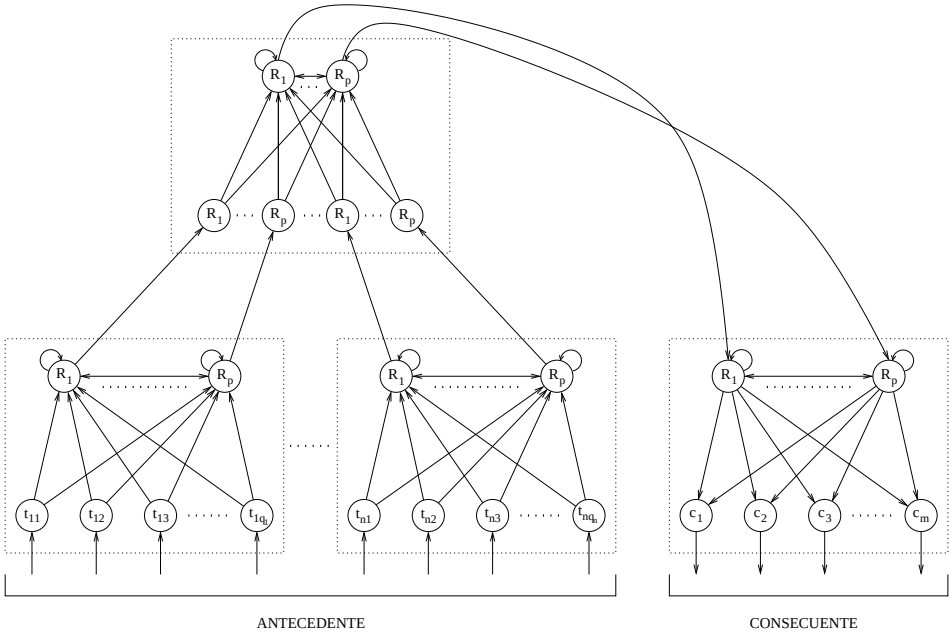
A continuación se especificará el comportamiento de las CCLAM asociadas tanto al antecedente como al consecuente.

Antecedente

El antecedente lo constituyen las CCLAM asociadas a las variables lingüísticas del antecedente además de la CCLAM unificadora. En estas memorias sólo se usará el proceso de clasificación, es decir, la información solamente fluirá desde las capas F_A hacia las capas F_B.

CCLAM asociada a variable lingüística del antecedente. En las memorias asociadas a las variables lingüísticas del antecedente

Figura 4.15: CCLAM que almacena el conjunto de reglas



se determinará el grado en que es aplicable cada una de las reglas almacenadas.

Puesto que las reglas están codificadas en los pesos de las conexiones existentes entre las capas F_A y F_B se usará como función Υ^C una t-norma. De este modo, para que un EP de F_B reciba una señal no nula es necesario que el patrón de entrada active algún EP de F_A que forme parte de la regla representada por el elemento de F_B.

La función Ψ^C que se usa es una t-conorma para que en F_B sólo queden activos los elementos que recibieron de F_A alguna señal no nula. El estado de activación resultante en los EP de F_B refleja el grado de cumplimiento de cada una de las reglas almacenadas.

La competición de F_B se anulará porque deseamos saber todas las las reglas que se dispararán y no solamente aquella que lo hará en mayor medida. Por tanto los pesos de las conexiones de la

competición serán $\beta_{ij}^C = 0 \quad \forall i, j$.

Unificador La CCLAM unificadora tendrá en F_B tantos EP como reglas memorizadas y en F_A habrá por cada variable lingüística del antecedente tantos EP como reglas. Los pesos de las conexiones que se establecen entre las capas F_A y F_B son tales que los EP asociados a la misma regla están unidos por conexiones con peso cero mientras que los demás están unidos por conexiones con peso uno. Esta asignación de pesos se debe a que el valor uno es el elemento neutro de las t-normas, lo cual será muy útil en la propagación de la información desde F_A hasta F_B.

El grado en que se dispara una regla se calcula mediante la aplicación de una t-norma a los grados en que se cumplen las variables del antecedente. Por ello se usará una t-norma, normalmente el mínimo, como función Ψ^C. Esa t-norma se aplicará a la información procedente de los EP de F_A. Para que sólo se tenga en cuenta el estado de activación de los elementos asociados a la misma regla es necesario que, desde los EP asociados a reglas distintas se reciba el elemento neutro de la t-norma. Dada la asignación de pesos de las conexiones que existen entre las capas F_A y F_B, la función Υ^C seleccionada será una t-conorma, por ejemplo el máximo. Los elementos asociados a distintas reglas, al estar conectados con pesos unitarios, envían hacia F_B el elemento neutro de la t-norma; los elementos asociados a la misma regla, al estar conectados con pesos nulos y ser éstos los elementos neutros de la t-conorma, envían hacia F_B su propio estado de activación.

Eliminamos de F_B la competición ya que no buscamos una regla ganadora sino el grado en que se dispara cada una de ellas. Por tanto los pesos de la competición serán $\beta_{ij}^C = 0 \quad \forall i, j$.

Los EP de F_B tienen finalmente por estado de activación el grado en que se debe disparar la regla que representa cada uno de ellos.

Consecuente

La CCLAM asociada al consecuente recibe como entrada el estado de activación de los EP de la capa F_B del unificador. Es decir, cada EP toma como estado de activación el grado en que se debe

disparar la regla a la que está asociado. El resultado que se espera obtener es el que se obtiene de la aplicación de todas las reglas que se deben disparar.

No deseamos que se produzca una competición entre los EP de F_B porque no buscamos un elemento ganador. Por tanto todos los pesos de la competición se anulan $\beta_{ij}^R = 0 \quad \forall i, j$.

Como función Υ^R se seleccionará una t-norma. De este modo, las reglas que no se disparen, es decir, aquellas cuyos elementos representantes tienen estado de activación nulo, enviarán hacia la capa F_A el valor cero, que es el valor neutro de las t-conormas. Seleccionando como función Ψ^R una t-conorma obtenemos en F_A el resultado de la aplicación de todas las reglas que se disparan.

4.3.6 Ejemplos

Para mostrar el funcionamiento de la CCLAM para el almacenamiento de distintos tipos de información se exponen unos sencillos ejemplos.

4.3.6.1 Almacenamiento de la altura de un conjunto de individuos

Tomemos la variable lingüística ALTURA. En esta variable lingüística vamos a usar las etiquetas lingüísticas MUY BAJO (MB), BAJO (B), NORMAL (N), ALTO (A) y MUY ALTO (MA) para expresar la altura de tres personas:

Nombre	MB	B	N	A	MA
Toni	0	0	0	0.5	0.5
David	0	0	0.9	0.1	0
Ana	0	0	0.2	0.8	0

El proceso de aprendizaje da lugar a las siguientes matrices de

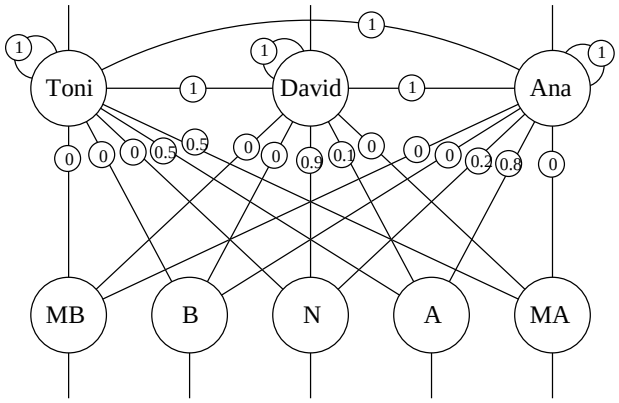
pesos:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0 \end{pmatrix} \tag{4.11}$$

$$\beta^C = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \beta^R = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.12}$$

El aspecto que tendrá la CCLAM será el mostrado en la figura 4.16.

Figura 4.16: CCLAM que almacena la altura de tres personas



Supongamos que se suministra como entrada el patrón:

$$X = (x_{MB}, x_B, x_N, x_A, x_{MA}) = (0, 0, 0, 0.6, 0.4)\,. \tag{4.13}$$

Las funciones que se utilizan para transmitir la información desde F_A hacia F_B son $\Upsilon^C(x, y) = 1 - |x - y|$ y Ψ^C=media aritmética.

En la celda correspondiente a la fila S y la columna T de la siguiente tabla se muestra el resultado de la función $\Upsilon^C\left(x_T, m^C_{T,S}\right)$. Al final de la fila, se muestra el valor que devuelve Ψ^C al ser aplicada sobre todos los valores de dicha fila.

Υ^C	MB	B	N	A	MA	Ψ^C
Toni	1	1	1	0.9	0.9	0.96
David	1	1	0.1	0.5	0.6	0.64
Ana	1	1	0.8	0.8	0.6	0.84

Los valores recibidos en la capa F_B son $Y_{Toni} = 0.96$, $Y_{David} = 0.64$ e $Y_{Ana} = 0.84$.

Todos los pesos de la competición valen uno. El elemento ganador de la competición pasa al estado uno mientras que el resto quedan desactivados. Por tanto, el estado de los elementos de F_B tras la competición será $Y_{Toni} = 1$, $Y_{David} = 0$ e $Y_{Ana} = 0$. Esto quiere decir que el patrón almacenado más cercano al de entrada es el correspondiente a la altura de Toni.

En el proceso de recuperación no se establece competición, es decir, los pesos se han ajustado de modo que no se modifica ningún estado de activación. Al ser los pesos $\beta_{ij}^R = 0 \quad \forall i, j$, ningún elemento cambia de estado tal y como se puede comprobar en la expresión que calcula el nuevo estado del i-ésimo EP:

$$\begin{cases} Y_i & \text{si } Y_i \geq Y_j \beta_{ji} \quad \forall j \\ 0 & \text{en otro caso .} \end{cases} \tag{4.14}$$

Para la propagación de la señal desde F_B hacia F_A se usa como función Υ^R el mínimo y como función Ψ^R el máximo. En la celda correspondiente a la fila S y la columna T de la siguiente tabla aparece el valor devuelto por la función $\Upsilon^R\left(Y_T, m_{T,S}^R\right)$. Al final de la fila se muestra el resultado de Ψ^R al ser aplicada sobre los valores de dicha fila.

Υ^R	Toni	David	Ana	Ψ^R
MB	0	0	0	0
B	0	0	0	0
N	0	0	0	0
A	0.5	0	0	0.5
MA	0.5	0	0	0.5

Por tanto, el patrón más parecido al patrón de entrada que se recupera $(0, 0, 0, 0.5, 0.5)$ es el correspondiente a la altura de Toni.

4.3.6.2 Almacenamiento de la altura, peso y edad de un conjunto de individuos

Tomemos la variable lingüística ALTURA. En esta variable lingüística vamos a usar las etiquetas lingüísticas MUY BAJO (MB), BAJO (B), NORMAL (N), ALTO (A) y MUY ALTO (MA).

Sea también la variable lingüística PESO con la que utilizaremos las etiquetas lingüísticas MUY LIGERO (ML), LIGERO (L), NORMAL (N), PESADO (P) y MUY PESADO (MP).

Por último tomaremos la variable lingüística EDAD. Se usarán las etiquetas lingüísticas NIÑEZ (N), PUBERTAD (P), JUVENTUD (J), MEDIANA EDAD (ME), EDAD AVANZADA (EA) y VEJEZ (V).

Los patrones que se quiere almacenar son los correspondientes a tres personas:

	ALTURA				
	MB	B	N	A	MA
Toni	0	0	0	0.5	0.5
David	0	0	0.9	0.1	0
Ana	0	0	0.2	0.8	0

	PESO				
	ML	L	N	P	MP
Toni	0	0	0	0.9	0.1
David	0	0.3	0.7	0	0
Ana	0	0	0.6	0.4	0

	EDAD					
	N	P	J	ME	EA	V
Toni	0	0	0.8	0.2	0	0
David	0	0	0	0.4	0.6	0
Ana	0	0.9	0.1	0	0	0

El proceso de aprendizaje dará lugar a las siguientes matrices de pesos:

$$M_{ALTURA} = \begin{pmatrix} 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0 \end{pmatrix} \qquad \beta_{ALTURA} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.15}$$

$$M_{PESO} = \begin{pmatrix} 0 & 0 & 0 & 0.9 & 0.1 \\ 0 & 0.3 & 0.7 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0 \end{pmatrix} \qquad \beta_{PESO} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.16}$$

$$M_{EDAD} = \begin{pmatrix} 0 & 0 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.6 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 & 0 \end{pmatrix} \qquad \beta_{EDAD} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.17}$$

$$M_{UNIF.} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad \beta_{UNIF.} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{4.18}$$

Figura 4.17: CCLAM asociada a la altura

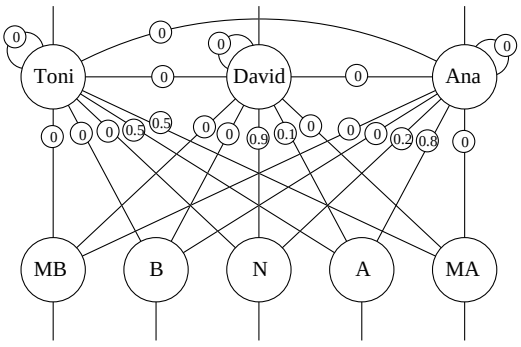


Figura 4.18: CCLAM asociada al peso

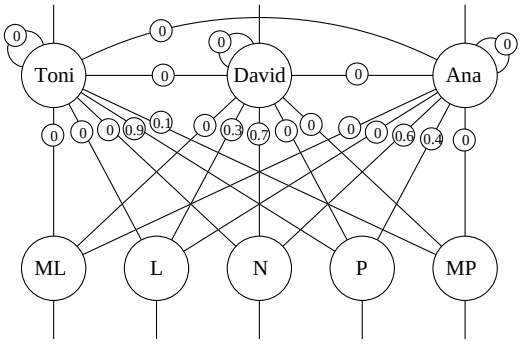
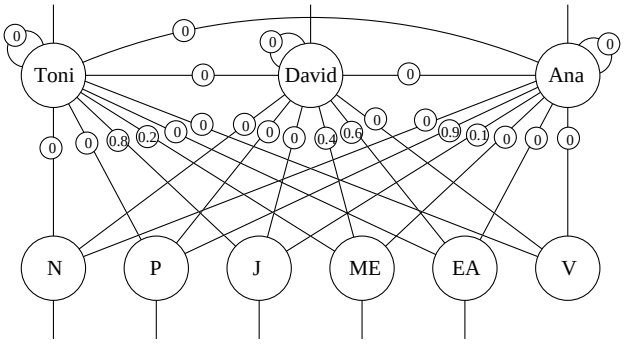


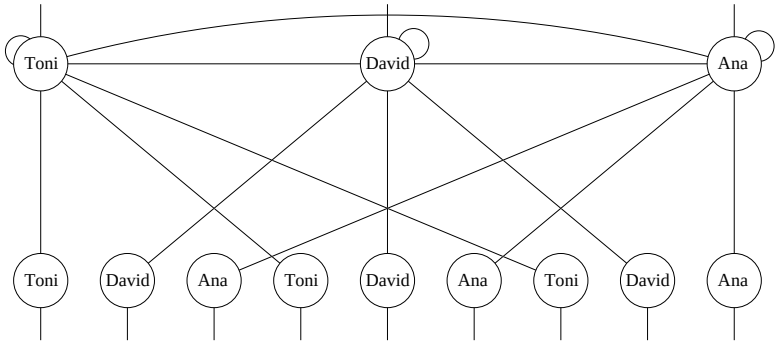
Figura 4.19: CCLAM asociada a la edad



Supongamos que se suministra como entrada el patrón (H, P, E) donde H es el subpatrón que expresa la altura, P es el que expresa el peso y E es el que expresa la edad. La altura, peso y edad indicados a la entrada son:

$$H = (h_{MB}, h_B, h_N, h_A, h_{MA}) = (0, 0, 1, 0, 0)$$
$$P = (p_{ML}, p_L, p_N, p_P, p_{MP}) = (0, 0, 0.5, 0.5, 0)$$
$$E = (e_N, e_P, e_J, e_{ME}, e_{EA}, e_V) = (0, 1, 0, 0, 0, 0)\,.$$

Figura 4.20: CCLAM unificadora (sólo se muestran las conexiones con peso uno; el resto tiene peso cero)



Por tanto el patrón presentado a la entrada es:

$$(0, 0, 1, 0, 0,\ \ 0, 0, 0.5, 0.5, 0,\ \ 0, 1, 0, 0, 0, 0)\ .$$

Las funciones que se utilizan para transmitir la información desde la capa F_A hacia F_B en las CCLAM asociadas a las variables lingüísticas son $\Upsilon^C(x, y) = 1 - |x - y|$ y Ψ^C=media aritmética.

El proceso de clasificación que se realiza en la CCLAM asociada a la altura da lugar a los resultados mostrados en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado de la función $\Upsilon^C\left(h_T, m^C_{T,S}\right)$. Al final de la fila, se muestra el valor devuelto por Ψ^S al ser aplicado sobre todos los valores de dicha fila.

Υ^C	MB	B	N	A	MA	Ψ^C
Toni	1	1	0	0.5	0.5	0.6
David	1	1	0.9	0.9	1	0.96
Ana	1	1	0.2	0.2	1	0.68

Los valores recibidos en la capa F_B son $Y_{Toni} = 0.6$, $Y_{David} = 0.96$

e $Y_{Ana} = 0.68$.

Todos los pesos de la competición valen cero de modo que todos los EP de la capa F_B conservan su estado.

El proceso de clasificación que se realiza en la CCLAM asociada al peso da lugar a los resultados mostrados en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado de la función $\Upsilon^C\left(p_T, m^C_{T,S}\right)$. Al final de la fila, se muestra el valor devuelto por Ψ^C al ser aplicado sobre todos los valores de dicha fila.

Υ^C	ML	L	N	P	MP	Ψ^C
Toni	1	1	0.5	0.6	0	0.62
David	1	0.7	0.8	0.5	1	0.8
Ana	1	1	0.9	0.9	1	0.96

Los valores recibidos en la capa F_B son $Y_{Toni} = 0.62$, $Y_{David} = 0.8$ e $Y_{Ana} = 0.96$.

Todos los pesos de la competición valen cero de modo que todos los EP de la capa F_B conservan su estado.

El proceso de clasificación que se realiza en la CCLAM asociada a la edad da lugar a los resultados mostrados en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado de la función $\Upsilon^C\left(e_T, m^C_{T,S}\right)$. Al final de la fila, se muestra el valor devuelto por Ψ^C al ser aplicado sobre todos los valores de dicha fila.

Υ^C	N	P	J	EM	EA	V	Ψ^C
Toni	1	0	0.2	0.8	1	1	0.67
David	1	0	1	0.6	0.4	1	0.67
Ana	1	0.9	0.9	1	1	1	0.97

Los valores recibidos en la capa F_B son $Y_{Toni} = 0.67$, $Y_{David} = 0.67$

e $Y_{Ana} = 0.97$.

Todos los pesos de la competición valen cero de modo que todos los EP de la capa F_B conservan su estado.

La CCLAM unificadora recibe a la entrada la señal procedente de las capas F_B de las CCLAM asociadas a la altura, peso y edad. Al propagar esta señal hacia F_B se obtienen los resultados mostrados en la siguiente tabla. En la celda correspondiente a la fila S y la columna T se muestra el resultado de la función $\Upsilon^C\left(y_T, \beta^C_{T,S}\right)$. Al final de la fila, se muestra el resultado devuelto por Ψ^C al ser aplicado sobre todos los valores de dicha fila.

	ALTURA			PESO			EDAD			
Υ^C	Toni	David	Ana	Toni	David	Ana	Toni	David	Ana	Ψ^C
Toni	0.6	0	0	0.62	0	0	0.67	0	0	0.21
David	0	0.96	0	0	0.8	0	0	0.67	0	0.27
Ana	0	0	0.68	0	0	0.96	0	0	0.97	0.29

Todos los pesos de la competición valen uno. El elemento ganador de la competición pasa al estado uno mientras que el resto quedan desactivados. Por tanto, el estado de los elementos de F_B tras la competición será $Y_{Toni} = 0$, $Y_{David} = 0$ e $Y_{Ana} = 1$. Esto quiere decir que el patrón almacenado más cercano al de entrada es el correspondiente a Ana.

En la recuperación el unificador transmite la selección realizada a las CCLAM asociadas a las variables.

La competición está anulada al establecerse $\beta^R_{ij} = 0 \quad \forall i, j$ por lo que los EP de F_B conservan su estado.

Como los pesos de las conexiones que se establecen entre las capas F_A y F_B son iguales a los establecidos en la fase de clasificación y se utiliza el producto como función Υ^R, los EP asociados al mismo patrón son los que reciben señales distintas de cero. De este modo, la señal recibida en la capa F_B de la CCLAM asociada a la altura es $Y_{Toni} = 0$, $Y_{David} = 0$ e $Y_{Ana} = 1$, la recibida en la CCLAM asociada al peso es $Y_{Toni} = 0$, $Y_{David} = 0$ e $Y_{Ana} = 1$ y la recibida en la CCLAM asociada a la edad es $Y_{Toni} = 0$, $Y_{David} = 0$ e $Y_{Ana} = 1$.

En el proceso de recuperación no hay competición en las CCLAM

asociadas a variables lingüísticas, es decir, los pesos se han ajustado de modo que la competición no modifica ningún estado de activación. Al ser los pesos $\beta_{ij}^R = 0 \quad \forall i, j$, ningún elemento cambia de estado.

En las siguientes tablas se muestra la señal propagada desde F_B hacia F_A en la CCLAM de la altura, peso y edad. En la celda correspondiente a la fila S y la columna T aparece el resultado devuelto por la función $\Upsilon^R\left(Y_T, m_{T,S}^R\right)$. Al final de la fila se encuentra el valor devuelto por Ψ^R al ser aplicada sobre los valores de dicha fila.

Υ^R	Toni	David	Ana	Ψ^R
MB	0	0	0	0
B	0	0	0	0
N	0	0	0.2	0.2
A	0	0	0.8	0.8
MA	0	0	0	0

Υ^R	Toni	David	Ana	Ψ^R
ML	0	0	0	0
L	0	0	0	0
N	0	0	0.6	0.6
P	0	0	0.4	0.4
MP	0	0	0	0

Υ^R	Toni	David	Ana	Ψ^R
N	0	0	0	0
P	0	0	0.9	0.9
J	0	0	0.1	0.1
ME	0	0	0	0
EA	0	0	0	0
V	0	0	0	0

Por tanto, el patrón más parecido al patrón de entrada que se

recupera $(0, 0, 0.2, 0.8, 0,\ 0, 0, 0.6, 0.4, 0,\ 0, 0.9, 0.1, 0, 0, 0)$ es el correspondiente a la información almacenada de Ana.

4.3.6.3 Almacenamiento de reglas de frenado de un móvil

Supongamos que deseamos almacenar mediante CCLAM el conjunto de reglas que identifican el sistema que tiene por objetivo el frenado de un móvil mediante la aplicación de una fuerza que depende de la posición y velocidad del mismo.

Supongamos que la posición la expresamos mediante la variable lingüística POSICIÓN usando para ello las etiquetas lingüísticas NEGATIVA ALTA (NA), NEGATIVA BAJA (NB), CERO (CE), POSITIVA BAJA (PB) y POSITIVA ALTA (PA).

La velocidad se expresará mediante la variable lingüística VELOCIDAD usando las etiquetas lingüísticas NEGATIVA (NE), CERO (CE) y POSITIVA (PO).

La respuesta ofrecida por el sistema para el frenado del móvil corresponderá a la fuerza que es necesario aplicarle. Ésta se representará por la variable FUERZA mediante las etiquetas lingüísticas NEGATIVA ALTA (NA), NEGATIVA BAJA (NB), CERO (CE), POSITIVA BAJA (PB) y POSITIVA ALTA (PA).

El conjunto de reglas que caracterizan el sistema es:

1. Si posición positiva alta y velocidad cero entonces fuerza negativa alta.
2. Si posición positiva baja y velocidad positiva entonces fuerza negativa baja.
3. Si posición positiva baja y velocidad negativa entonces fuerza cero.
4. Si posición negativa alta y velocidad cero entonces fuerza positiva alta.
5. Si posición negativa baja y velocidad negativa entonces fuerza positiva baja.
6. Si posición negativa baja y velocidad positiva entonces fuerza cero.
7. Si posición cero y velocidad cero entonces fuerza cero.

El conjunto de CCLAM que almacenará las reglas memorizará tríos de patrones (P, V, F) donde P y V son los patrones que representan la posición y velocidad del móvil respectivamente y F es el patrón que representa la fuerza que es necesario aplicar para detenerlo. La codificación de las reglas dará lugar al siguiente conjunto de patrones:

	Posición					Velocidad			Fuerza				
R.	NA	NB	CE	PB	PA	NE	CE	PO	NA	NB	CE	PB	PA
1	0	0	0	0	1	0	1	0	1	0	0	0	0
2	0	0	0	1	0	0	0	1	0	1	0	0	0
3	0	0	0	1	0	1	0	0	0	0	1	0	0
4	1	0	0	0	0	0	1	0	0	0	0	0	1
5	0	1	0	0	0	1	0	0	0	0	0	1	0
6	0	1	0	0	0	0	0	1	0	0	1	0	0
7	0	0	1	0	0	0	1	0	0	0	1	0	0

Para la memorización de las reglas utilizaremos cuatro CCLAM. Dos de ellas se usarán para almacenar los antecedentes de las reglas, es decir, los valores de posición y velocidad. El resultado de estas dos memorias lo unificará otra CCLAM y la información obtenida se enviará a una cuarta y última CCLAM encargada de recuperar la información relativa a la fuerza que se debe aplicar.

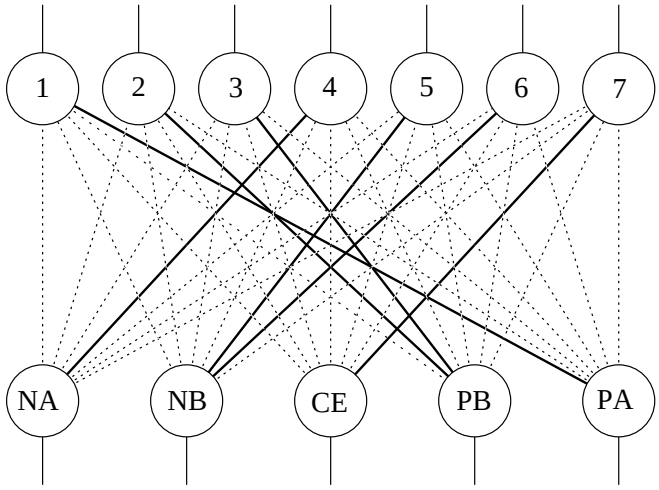
Tras el proceso de aprendizaje, la CCLAM asociada a la posición tendrá las siguientes matrices de pesos:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad \beta^C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.19}$$

La figura 4.21 muestra el aspecto de la CCLAM asociada a la posición tras la memorización de las reglas. Para simplificar el grá-

fico se han omitido las conexiones de peso cero de la competición. Las conexiones existentes entre las capas F_A y F_B se muestran con linea continua si tienen peso uno y con linea discontinua si tienen peso cero.

Figura 4.21: CCLAM asociada a la posición



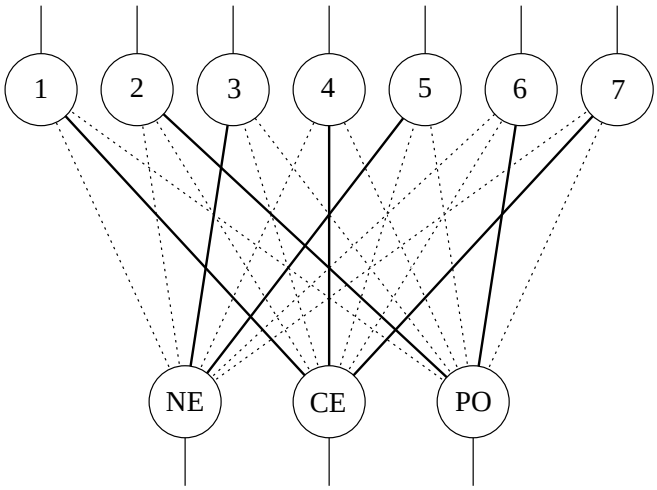
Las matrices de pesos de la CCLAM asociada a la velocidad son:

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \beta^C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.20}$$

La figura 4.22 muestra el aspecto de la CCLAM asociada a la velocidad tras la memorización de las reglas.

La información de las memorias asociadas a posición y velocidad se propaga hacia la CCLAM unificadora. Ésta tiene catorce EP en F_A (siete por cada una de las dos variables lingüísticas del

Figura 4.22: CCLAM asociada a la velocidad



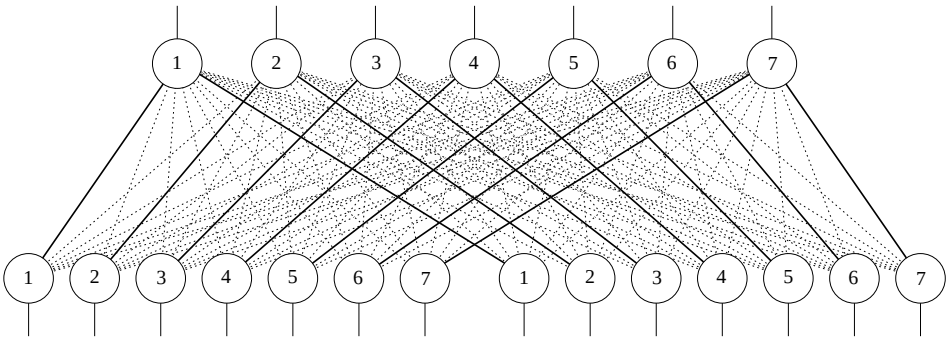
antecedente) y siete elementos en F_B. Las matrices de pesos son:

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \beta^C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.21}$$

La figura 4.23 muestra el aspecto de la CCLAM asociada al unificador (en este caso se muestran con trazo continuo las conexiones con peso nulo y con trazo discontinuo las de peso uno).

Las CCLAM asociadas a la posición y velocidad y la CCLAM unificadora son las que intervienen en el proceso de clasificación. A partir de un patrón de entrada en que se especifica la posición y velocidad del móvil se obtiene el grado en que se dispara cada una de las reglas del sistema. Dicha información se envía a una CCLAM que recuperará un patrón que representa la fuerza que el

Figura 4.23: CCLAM unificadora



sistema de reglas ha determinado que se debe aplicar. Las matrices de pesos de dicha CCLAM asociada a la fuerza son:

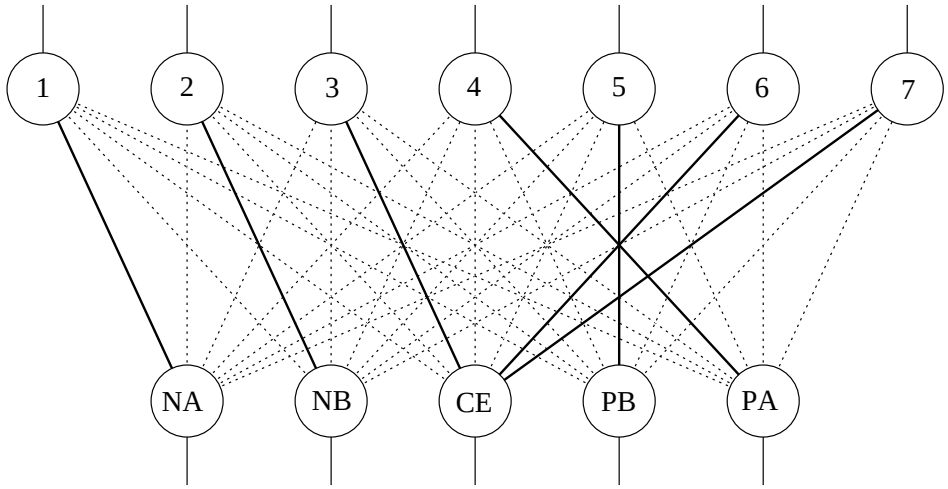
$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad \beta^R = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.22}$$

La figura 4.24 muestra el aspecto de la CCLAM asociada a la fuerza tras la memorización de las reglas.

En las CCLAM asociadas a la posición, velocidad y fuerza las conexiones con peso nulo se representan mediante lineas discontinuas y las conexiones con peso uno se representan con lineas continuas. En el caso del unificador son las conexiones con peso uno las que se representan con lineas discontinuas mientras que las de peso cero se muestran con trazo continuo.

Teniendo en cuenta lo dicho anteriormente, el conjunto de las CCLAM utilizadas para la memorización de las reglas se muestra en la figura 4.25 (para facilitar la representación se ha girado la CCLAM asociada a la fuerza de modo que su capa F_A está arriba y la F_B está abajo).

Figura 4.24: CCLAM asociada a la fuerza



Para determinar el valor de la fuerza necesaria para parar el móvil utilizaremos la t-norma del mínimo y la t-conorma del máximo. Por tanto, la función Υ de las CCLAM asociadas a variables lingüísticas será el mínimo y la función Ψ será el máximo.

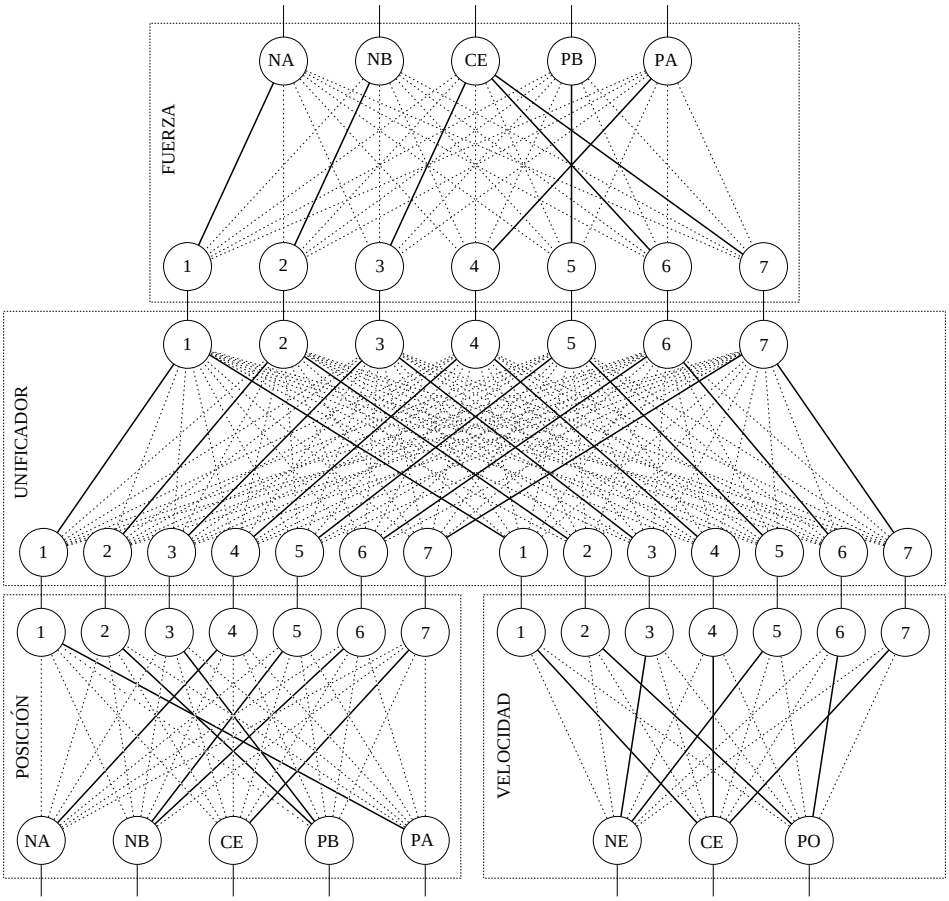
Supongamos que la posición y velocidad del móvil que deseamos frenar son:

- Posición $\frac{1}{4}$ negativa alta, $\frac{3}{4}$ negativa baja,
- Velocidad $\frac{1}{2}$ negativa, $\frac{1}{2}$ cero.

Los patrones que codifican esta situación son:

$$\begin{array}{c} P = \left(\frac{1}{4}, \frac{3}{4}, 0, 0, 0\right) \\ V = \left(\frac{1}{2}, \frac{1}{2}, 0\right) \end{array} \tag{4.23}$$

Figura 4.25: Conjunto de CCLAM que almacenan las reglas del sistema de frenado



Al ser suministrado el patrón P a la entrada de la CCLAM asociada a la posición la señal propagada desde F_A hacia F_B se muestra en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado devuelto por $\Upsilon^C\left(P_T, m^C_{T,S}\right) = \min\left(P_T, m^C_{T,S}\right)$. Al final de la fila se encuentra el valor proporcionado por Ψ^C al ser aplicada sobre los valores de dicha fila.

Υ^C	NA	NB	CE	PB	PA	Ψ^C
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	$\frac{1}{4}$	0	0	0	0	$\frac{1}{4}$
5	0	$\frac{3}{4}$	0	0	0	$\frac{3}{4}$
6	0	$\frac{3}{4}$	0	0	0	$\frac{3}{4}$
7	0	0	0	0	0	0

Del mismo modo, al ser suministrado el patrón V a la entrada de la CCLAM asociada a la velocidad la señal propagada desde F_A hacia F_B se muestra en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado de la función $\Upsilon^C\left(P_T, m^C_{T,S}\right) = \min\left(P_T, m^C_{T,S}\right)$. Al final de la fila se encuentra el valor devuelto por Ψ^C al ser aplicada sobre los valores de dicha fila.

Υ^C	NE	CE	PO	Ψ^C
1	0	$\frac{1}{2}$	0	$\frac{1}{2}$
2	0	0	0	0
3	$\frac{1}{2}$	0	0	$\frac{1}{2}$
4	0	$\frac{1}{2}$	0	$\frac{1}{2}$
5	$\frac{1}{2}$	0	0	$\frac{1}{2}$
6	0	0	0	0
7	0	$\frac{1}{2}$	0	$\frac{1}{2}$

El resultado obtenido en las dos CCLAM anteriores se envía al unificador. Procedente de la memoria asociada a la posición recibe el patrón $\left(0, 0, 0, \frac{1}{4}, \frac{3}{4}, \frac{3}{4}, 0\right)$ y procedente de la memoria asociada a la velocidad recibe $\left(\frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2}\right)$.

La señal propagada desde F_A hacia F_B se muestra en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado que proporciona $\Upsilon^C\left(P_T, m^C_{T,S}\right) = \max\left(P_T, m^C_{T,S}\right)$. Al final de la fila se encuentra el valor devuelto por $\Psi^C = \min$ al ser aplicada sobre los valores de dicha fila.

	POSICIÓN							VELOCIDAD							
Υ^C	1	2	3	4	5	6	7	1	2	3	4	5	6	7	Ψ^C
1	0	1	1	1	1	1	1	$\frac{1}{2}$	1	1	1	1	1	1	0
2	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0
3	1	1	0	1	1	1	1	1	1	$\frac{1}{2}$	1	1	1	1	0
4	1	1	1	$\frac{1}{4}$	1	1	1	1	1	1	$\frac{1}{2}$	1	1	1	$\frac{1}{4}$
5	1	1	1	1	$\frac{3}{4}$	1	1	1	1	1	1	$\frac{1}{2}$	1	1	$\frac{1}{2}$
6	1	1	1	1	1	$\frac{3}{4}$	1	1	1	1	1	1	0	1	0
7	1	1	1	1	1	1	0	1	1	1	1	1	1	$\frac{1}{2}$	0

El resultado obtenido por el unificador indica que las reglas aplicables son la cuarta y la quinta con grados $\frac{1}{4}$ y $\frac{1}{2}$ respectivamente. El patrón obtenido $\left(0, 0, 0, \frac{1}{4}, \frac{1}{2}, 0, 0\right)$ se envía a la CCLAM asociada a la fuerza para proceder allí a la recuperación de la información buscada.

La señal propagada desde F_B hacia F_A se muestra en la siguiente tabla. En la celda correspondiente a la fila S y la columna T aparece el resultado dado por la función $\Upsilon^R\left(P_T, m^R_{T,S}\right) = \min\left(P_T, m^R_{T,S}\right)$. Al final de la fila se encuentra el valor devuelto por $\Psi^R = \max$ al ser aplicada sobre los valores de dicha fila.

Υ^R	1	2	3	4	5	6	7	Ψ^R
NA	0	0	0	0	0	0	0	0
NB	0	0	0	0	0	0	0	0
CE	0	0	0	0	0	0	0	0
PB	0	0	0	0	$\frac{1}{2}$	0	0	$\frac{1}{2}$
PA	0	0	0	$\frac{1}{4}$	0	0	0	$\frac{1}{4}$

Por tanto, para frenar un móvil que se halla en una posición $\frac{1}{4}$ negativa alta, $\frac{3}{4}$ negativa baja y que se mueve con una velocidad $\frac{1}{2}$ negativa, $\frac{1}{2}$ cero, es necesario aplicarle una fuerza $\frac{1}{2}$ positiva baja, $\frac{1}{4}$ positiva alta.

4.4 Comparación de las dos CLAM difusas

En las secciones precedentes hemos construido dos memorias asociativas capaces de almacenar información imprecisa. A continuación compararemos la eficiencia de los dos modelos cuando almacenan patrones y reglas difusas.

4.4.1 Almacenamiento de patrones

Supongamos que queremos almacenar los atributos de N objetos, expresados mediante V variables lingüísticas. La variable A_i tomará valores en un conjunto de T_i etiquetas lingüísticas.

Usando el método CDI con P bits de precisión necesitaremos PT_i bits para codificar el valor de la variable A_i. Por tanto, los patrones que se almacenarán constarán de $P\sum_{i=1}^{V} T_i$ componentes. El número de patrones a almacenar será N.

El número de EP necesarios es igual al número de patrones más el número de componentes, es decir, $N + P\sum_{i=1}^{V} T_i$. El número de pesos presentes en la matriz de pesos es $NP\sum_{i=1}^{V} T_i$. El espacio total necesario es:

$$N + P\sum_{i=1}^{V} T_i + NP\sum_{i=1}^{V} T_i. \tag{4.24}$$

Usando CCLAM para almacenar los patrones necesitamos una memoria para cada variable más una memoria para el unificador. La CCLAM asociada a la variable A_i tendrá T_i elementos en la capa F_A y N elementos en la capa F_B. La memoria asociada al unificador tendrá VN elementos en la capa F_A y N elementos en F_B. Por tanto, el número de EP necesarios será $N\left(2V+1\right) + \sum_{i=1}^{V} T_i$, el número de conexiones entre las capas F_A y F_B será $N\left(VN + \sum_{i=1}^{V} T_i\right)$ y el número de conexiones para establecer la competición en las capas F_B será $\left(V+1\right)N^2$. El espacio total necesario es:

$$N\left(2V+1\right) + \sum_{i=1}^{V} T_i + N\left(VN + \sum_{i=1}^{V} T_i\right) + \left(V+1\right)N^2. \tag{4.25}$$

Volvamos al ejemplo mostrado en la página 140 con el almacenamiento de la altura y peso de cinco personas, usando cinco etiquetas para describir cada atributo y diez bits de precisión. Usando el método CDI necesitaremos almacenar 5 patrones de 100 componentes, en una CLAM de 105 elementos de proceso y un total de 500 conexiones entre las dos capas de EP.

Si queremos almacenar la misma información usando CCLAM necesitamos dos memorias de 5 EP en la capa F_A y 5 EP en la capa F_B para guardar la información de los dos atributos más otra memoria que unificará la información con 10 EP en la capa F_A y 5 EP en la capa F_B. En total serán necesarias 175 conexiones y 35 EP de una décima de precisión.

4.4.2 Almacenamiento de reglas

Supongamos que deseamos memorizar un conjunto de R reglas con M antecedentes y un consecuente. El antecedente A_i lo constituye una variable lingüística que toma valores en un conjunto de T_i etiquetas lingüísticas. El consecuente toma valores en un conjunto de T_C etiquetas lingüísticas.

Si utilizamos el método CDI para almacenar en una CLAM las reglas, necesitamos memorizar tantos patrones como sea necesario para codificar todas las situaciones posibles que se pueden presentar. Junto a los antecedentes que codifican dichas situaciones deben ir los consecuentes correspondientes al aplicar las reglas en cada situación.

Usando P bits de precisión para codificar las variables lingüísticas serán necesarios PT_i componentes para representar el valor de la variable i. Por tanto, los patrones que se memorizarán tendrán $P\left(T_C + \sum_{i=1}^{M} T_i\right)$ componentes.

El número de valores distintos que puede presentar el antecedente i es $P\left(T_i - 1\right) + 1$. Por tanto, el número de situaciones distintas que se pueden presentar al combinar todas las posibilidades que presentan los antecedentes es $\prod_{i=1}^{M}\left(P\left(T_i - 1\right) + 1\right)$. Para cada una de estas situaciones es necesario codificar un patrón con el consecuente adecuado a cada situación. Dicho consecuente se calculará a partir de las reglas aplicables en cada situación.

El número de elementos de proceso necesarios para formar la memoria será igual al número de componentes más el número de patrones almacenados, es decir:

$$P\left(T_C + \sum_{i=1}^{M} T_i\right) + \prod_{i=1}^{M} \left(P\left(T_i - 1\right) + 1\right). \tag{4.26}$$

La matriz de pesos tendrá $\prod_{i=1}^{M} \left(P\left(T_i - 1\right) + 1\right)$ filas mientras que el número de columnas será $P\left(T_C + \sum_{i=1}^{M} T_i\right)$. Por tanto el número total de elementos que formará la matriz es:

$$P\left(T_C + \sum_{i=1}^{M} T_i\right) \prod_{i=1}^{M} \left(P\left(T_i - 1\right) + 1\right). \tag{4.27}$$

Volviendo al sistema de frenado mostrado como ejemplo anteriormente, nos encontramos con 7 reglas en que el consecuente toma valores en un conjunto de 5 etiquetas y en que los dos antecedentes toman valores en conjuntos de 5 y 3 etiquetas respectivamente. Usando 10 bits de precisión necesitaremos almacenar 492 patrones de 130 componentes en una memoria de 622 elementos de proceso y un total de 63960 pesos.

Si utilizamos CCLAM para almacenar un conjunto de reglas necesitaremos una CCLAM para cada una de las variables lingüísticas además de la CCLAM usada como unificador.

La CCLAM asociada al antecedente A_i tendrá T_i componentes en la capa F_A y R componentes en la capa F_B. El unificador tendrá MR componentes en la capa F_A y R componentes en la capa F_B. La memoria asociada al consecuente tendrá T_C elementos en la capa F_A y R elementos en F_B. El número de elementos de proceso necesarios será:

$$\sum_{i=1}^{M} T_i + R + MR + R + T_C + R. \tag{4.28}$$

El número de componentes que formarán las matrices de pesos de las memorias asociadas a los antecedentes es RT_i. La matriz de pesos del unificador la formarán MR^2 componentes. La matriz

de pesos del consecuente la formarán RT_C componentes. El número de conexiones necesarias para la competición de las capas F_B es $(M+1)R^2$. Por tanto, el número de pesos que es necesario almacenar es:

$$R\sum_{i=1}^{M} T_i + MR^2 + RT_C + (M+1)R^2. \tag{4.29}$$

Siguiendo con el ejemplo del sistema de frenado, para almacenar todas las reglas necesitamos un conjunto de memorias con un total de 48 elementos de proceso y 336 pesos. La memoria asociada a la posición almacenará 7 patrones de 5 componentes, la asociada a la velocidad almacenará 7 patrones de 3 componentes, la asociada al consecuente almacenará 7 patrones de 5 componentes y el unificador almacenará 7 patrones de 14 componentes.

Para obtener una precisión semejante a los 10 bits usados anteriormente es necesario que todos los EP tengan una precisión de una décima.

4.4.3 Ventajas e inconvenientes

Empleo del método CDI con la CLAM

- Ventajas
  - Todos los elementos de proceso son discretos (bipolares y binarios).
  - Los cálculo son más simples con EP discretos.
- Inconvenientes
  - En el almacenamiento de reglas es necesario calcular a priori todos los posibles antecedentes y sus correspondientes consecuentes.
  - Almacenar reglas implica que el número de patrones a almacenar se dispare al aumentar la precisión de modo que no se podrá alcanzar una precisión elevada.

  - El tamaño de la matriz de pesos crece con la precisión por lo que los problemas que requieran alta precisión no podrán ser abordados de un modo eficiente.

Empleo de CCLAM

- Ventajas
  - No es necesario ningún preprocesamiento de las reglas a almacenar.
  - El número de patrones a almacenar no depende de la precisión por lo que no es una cantidad elevada.
  - El tamaño de las matrices de pesos no depende de la precisión por lo que no constituyen una limitación a la precisión con la que se aborde el problema.
- Inconvenientes
  - Los elementos de proceso son continuos.
  - Los cálculos son más complejos al no trabajar con EP discretos.

4.5 Notas finales

Basándonos en una CLAM y empleando dos métodos distintos hemos obtenido dos memorias asociativas difusas que permiten almacenar información imprecisa:

- Por un lado, el empleo de CDI nos permite utilizar una CLAM para el almacenamiento de información expresada en términos lingüísticos mediante la discretización de los respectivos grados de compatibilidad.
- Por otro lado hemos obtenido la CCLAM como fruto de la adaptación de una CLAM para la memorización de información continua. Dicha CCLAM puede ser empleada para almacenar los grados de compatibilidad con un conjunto de etiquetas lingüísticas.

Las dos alternativas constituyen excelentes herramientas en cualquier sistema de tratamiento de información empleado en razonamiento aproximado por su alta capacidad y por su habilidad para trabajar con información imprecisa.

En general, el empleo del método CDI con una CLAM está indicado en problemas que requieran baja precisión o en situaciones en que sea poco eficiente el uso de elementos de proceso continuos. Cuando no es problemático el uso de EP continuos y deseamos trabajar con alta precisión, sobre todo para almacenar reglas difusas, lo mejor es emplear CCLAM.

Conclusiones

Los modelos existentes de memorias asociativas están afectados por dos problemas, los estados espurios y la dependencia de los datos, que limitan su capacidad de almacenamiento e impiden que sean realmente útiles. Nos hemos propuesto, a la vista de ello, construir una memoria asociativa que no presente dichos inconvenientes. Esto ha sido plenamente conseguido mediante:

1. La introducción de CLAM, un modelo de memoria asociativa discreta de alta capacidad que no presenta estados espurios ni impone condiciones sobre los patrones que almacena.
2. Construcción de dos memorias asociativas basadas en la Memoria Asociativa Clasificadora capaces de procesar información lingüística.

Más concretamente, del trabajo de investigación desarrollado podemos extraer las siguientes conclusiones particulares a modo de resumen:

- La CLAM no recupera patrones espurios sino que siempre recupera algún patrón de los que fueron almacenados.
- La CLAM no impone ningún tipo de restricción sobre los patrones que se memorizan por lo que no existe dependencia de los datos, es decir, permite el almacenamiento de cualquier patrón.
- La topología dinámica de la CLAM permite almacenar cuantos patrones se deseen sin limitar la capacidad por un número estático de elementos de proceso.

- La CLAM mejora notablemente en eficiencia a modelos tan extendidos como la BAM de modo que, empleando menos recursos, obtiene mejores resultados .
- La no existencia de estados oscilatorios y el empleo de operaciones matemáticas simples en los procesos de aprendizaje y recuperación hacen que la CLAM sea una memoria muy rápida.
- El método de codificación de variables lingüísticas mediante discretización incremental nos ha permitido emplear una CLAM que trabaja con información precisa para la obtención de una CLAM difusa que trabaja con información imprecisa expresada en términos lingüísticos.
- El empleo del método de Codificación por Discretización Incremental a modo de filtro permite el uso de la CLAM para almacenar información imprecisa sin la necesidad de diseñar y estudiar nuevos modelos.
- La Memoria Asociativa Clasificadora Continua CCLAM permite almacenar y recuperar correctamente información imprecisa expresada en términos lingüísticos.
- Usando la CCLAM como elemento estructural podemos construir agrupaciones de CCLAM que permitan almacenar información de más de una variable lingüística. De igual modo podemos construir memorias asociativas formadas por agrupaciones de CCLAM que permitan almacenar reglas difusas.
- La CCLAM puede devolver respuestas interpoladas de modo que su empleo para el almacenamiento de reglas difusas nos permite no sólo la recuperación de las reglas, sino también la obtención del consecuente apropiado para unos antecedentes cualesquiera.

Futuras investigaciones

- De forma general, podemos decir que un campo de investigación con gran futuro es la utilización de la *Memoria Asociativa Clasificadora,* tanto el modelo discreto como el difuso, en el campo del razonamiento aproximado.
- Empleo de técnicas de agrupamiento de patrones para la obtención de clasificaciones más acertadas cuando se realiza aprendizaje no supervisado así como la obtención de clases formadas por grupos de patrones no compactos.
- Aumento del rendimiento de la *Memoria Asociativa Clasificadora* mediante un mejor aprovechamiento de la información almacenada aprendiendo relaciones de dependencia entre las componentes de los patrones almacenados.
- Estudio del comportamiento de la CLAM y la CCLAM cuando se utilizan otras funciones de activación en los elementos de proceso.
- Aplicación de los sistemas resultantes de ésta y futuras investigaciones sobre implementaciones concretas de sistemas con objetivos específicos tales como el reconocimiento de patrones o sistemas de control.

Bibliografía

[1] Anderson J. A memory storage model utilizing spatial correlation functions. *Kybernetik*, tomo 5:113–119, 1968.

[2] Anderson J. Two models for memory organization using interacting traces. *Mathematical Biosciences*, tomo 8:137–160, 1970.

[3] Anderson J. A simple neural network generating an interactive memory. *Mathematical Biosciences*, tomo 14:197–220, 1972.

[4] Anderson J. A theory for the recognition of items from short memorized lists. *Psycological Review*, tomo 80:417–438, 1973.

[5] Axelrod R. *Structure of Design*. Princeton University Press, 1976.

[6] Bailón A., Delgado M. y Fajardo W. Clasificación de información con memorias asociativas. *Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial*. 1997.

[7] Bailón A., Delgado M. y Fajardo W. Memoria asociativa clasificadora mejorada. *Actas de la VIII Conferencia de la Asociación Española para la Inteligencia Artificial*. 1999.

[8] Bailón A., Delgado M. y Fajardo W. Clam: A new model of associative memory. *International Journal of Intelligent Systems*, tomo 15:549–564, 2000.

[9] Blanco A. *Identificación de Sistemas Mediante Redes Neuronales*. Tesis Doctoral, Universidad de Granada, 1993.

[10] Blanco A., Delgado M. y Fajardo W. Extension from a linear associative memory to a linguistic linear associative memory. *International Journal of Intelligent Systems*, tomo 13(1):41–58, 1998.

[11] Blanco A., Delgado M. y Fajardo W. Representation model of information in linguistic terms. *Fuzzy Sets and Systems*, (107):277–287, 1999.

[12] Carpenter G. y Grossberg S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, tomo 37:54–115, 1987.

[13] Chetayav N. *The Stability of Motion*. Pergamon Press, Nueva York, 1961.

[14] Delgado M., Verdeay J. y Vila M. A linguistic version of the compositional rule of inference. *Symposium of Intelligent Components and Instruments for Control Applications*, págs. 141–148. 1992.

[15] Fajardo W. *Recuperación de Información Mediante Memorias Asociativas Difusas*. Tesis Doctoral, Universidad de Granada, 1995.

[16] Fausett L. *Fundamentals of Neural Networks. Architectures, Algorithms and Applications*. Prntice Hall, 1994.

[17] Grossberg S. *Studies of Mind and Brain: Neural Principles of Learning, Perception, Cognition and Motor Control*. Reidel Press, 1982.

[18] Haines K. y Hecht-Nielsen R. A bam with increased information storage capacity. *Proceedings of the IEEE International Conference on Neural Networks*, págs. 181–190. IEEE, San Diego, 1988.

[19] Hebb D. *The Organization of Behavior*. John Wiley & Sons, 1949.

[20] Hohonen T. Correlation matrix memories. Informe Técnico TKK-F-A130, Helsinki University of Technology, 1972.

[21] Hopfield J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of Natural Academic Science*, págs. 2254–2258. 1982.

[22] Hopfield J. Neurons with graded response have collective computational poperties like those of two state neurons. *Proceedings of the National Academy of Sciences*, págs. 2554–2558. 1982.

[23] Kohonen T. Correlation associative memory. *IEEE Transactions on Computers*, tomo C-21:353–359, 1972.

[24] Kohonen T. *Associative Memory - A System Theoretical Approach*. Springer-Verlag, 1977.

[25] Kohonen T. y Ruohonen M. Representation of associated data by matrix operators. *IEEE Transactions on Computer*, tomo C-22:701–702, 1973.

[26] Kosko B. Adaptive inference. Informe técnico, Verac, 1985.

[27] Kosko B. Fuzzy cognitive maps. *International Journal of Man-Machine Studies*, tomo 24:65–75, 1986.

[28] Kosko B. Fuzzy entropy and conditioning. *Information Sciences*, tomo 40:165–174, 1986.

[29] Kosko B. Fuzzy knowledge combination. *International Journal for Intelligent Systems*, tomo 1:293–320, 1986.

[30] Kosko B. Fuzzy associative memories. A.Kandel, ed., *Fuzzy Expert Systems*. Addison-Wesley, 1987.

[31] Kosko B. *IEEE Transactions on Systems, Man and Cybernetics*, tomo SMC-18:42–60, 1988.

[32] Kosko B. Bidirectional associative memories. *IEEE Transactions on Systems, Man and Cybernetics*, tomo SMC-18:42–60, 1988.

[33] Kosko B. *Neural Networks and Fuzzy Systems*. Prentice-Hall International Editions, 1992.

[34] Leung C. Encoding method for bidirectional associative memory using projection on convex sets. *IEEE Trabsactions on Neural Networks*, tomo 4:879–881, 1993.

[35] Loos H. Reflexive associative memories. D. Anderson, ed., *Proceedings of the IEEE Conference on Neural Information Processing Systems - Natural and Synthetic*, págs. 495–504. American Institute of Physics, New York, 1988.

[36] Mamdani E. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers*, tomo C-26(12):1182–1191, 1977.

[37] McElice R., Posner E., Rodemich E. y Venkatesh S. The capacity of the hopfield associative memory. *IEEE Transactions on Information Theory*, tomo IT-33:461–482, 1987.

[38] Pedrycz W. Numerical and applicational aspects of fuzzy relational equations. *Fuzzy Sets and Systems*, tomo 11:1–18, 1983.

[39] Pedrycz W. Applications of fuzzy relational equations for method of reasoning in presence of fuzzy data. *Fuzzy Sets and Systems*, tomo 16:163–175, 1985.

[40] Pedrycz W. *Fuzzy Sets Engineering*. CRC Press, 1995.

[41] Rosenblatt F. The perceptron: A perceiving and recognizing automation (project para). Informe técnico, Cornell Aeronautical Laboratory, 1957.

[42] Simpson P. *Artificial Neural Systems*. Pergamon Press, 1990.

[43] Taber W. y Siegel M. Estimation of expert weights using fuzzy cognitive maps. IEEE, ed., *Proceedings of the IEEE First International Conference on Neural Networks*, págs. 319–326. 1987.

[44] Wang T., Zhuang X. y Xing X. Weighted learning of bidirectional associative memories by globl minimization. *IEEE Transactions on Neural Networks*, tomo 3:1010–1018, 1992.

[45] Wang Y., Cruz J. y Mulligan J. On multiple training for bidirectional associative memory. *IEEE Transactions on Neural Networks*, tomo 1:275–276, 1990.

[46] Wang Y., Cruz J. y Mulligan J. Two coding strategies for bidirectional associative memory. *IEEE Transactios on Neural Networks*, tomo 1:81–92, 1990.

[47] Wang Y., Cruz J. y Mulligan J. Guaranteed recall of all training pairs for bidirectional associative memory. *IEEE Transactios on Neural Networks*, tomo 2:559–567, 1991.

[48] Wee W. Generalized inverse approach to adaptive multiclass pattern classification. *IEEE Transactions on Computers*, tomo C-17:1157–1164, 1968.

[49] Zadeh L. Outline of a new approach to the analysis of complex systems. *IEEE Transactions on Systems, Man and Cybernetics*, tomo SMC(3):82–84, 1973.

[50] Zadeh L. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems man and Cybernetics*, tomo SSMC-3, 1973.

[51] Zadeh L. The concept of a linguistic variable and its applications to approximate reasoning, part i. *Information Sciences*, tomo 8:199–249, 1975.

[52] Zadeh L. The concept of a linguistic variable and its applications to approximate reasoning, part ii. *Information Sciences*, tomo 8:301–357, 1975.

[53] Zadeh L. The concept of a linguistic variable and its applications to approximate reasoning, part ii. *Information Sciences*, tomo 9:43–80, 1975.

[54] Zadeh L. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, (1):3–28, 1978.

[55] Zadeh L. Fuzzy sets and information ganularity. *Advances in Fuzzy Sets Theory and Applications*, págs. 3–19, 1979.

[56] Zadeh L. Fuzzy sets and information granularity. M. et al., ed., *Advances in Fuzzy Sets Theory and Applications*. North Holland, Nueva York, 1979.

[57] Zadeh L. A theory of approximate reasoning. *Machine Intelligence*, tomo 9:149–194, 1979.

[58] Zadeh L. A theory of approximate reasoning. tomo 9, págs. 149–194. Elsevier, 1979.

[59] Zadeh L. A theory of commonsense knowledge. T. Skala Termini, ed., *Aspects of vagueness*, págs. 257–295. Reidel, 1984.

[60] Zhang B., Xu B. y Kwong C. Performance analysis of the bidirectional associative memory and a improved model from the matched-filtering viewpoint. *IEEE Transactions on Neural Networks*, tomo 4:864–872, 1993.