# INSTITUTO POLITECNICO NACIONAL

## CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

## MEMORIAS ASOCIATIVAS ALFA-BETA SIMPLIFICADAS

## T E S I S

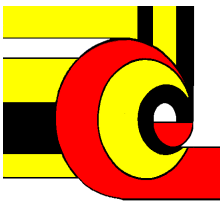
QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

**EDGAR ARMANDO CATALAN SALGADO**

DIRECTOR DE TESIS:

**DR. CORNELIO YÁÑEZ MÁRQUEZ**



MÉXICO, D.F. JUNIO 2007

# Índice General

# Indice Tablas y Figuras

# Resumen

Las memorias asociativas son modelos computacionales que permiten relacionar patrones de entrada y salida, y posteriormente recuperar el patrón correspondiente de salida a partir del de entrada, aunque éste haya sido alterado.

Uno de los modelos sobresalientes de hoy en día, son las memorias asociativas Alfa-Beta, las cuales se encuentran entre las mejores reportadas en la literatura científica actual, debido a su capacidad de almacenamiento, velocidad y tolerancia a patrones alterados.

Aunque las memorias asociativas Alfa-Beta estén entre las más rápidas de las conocidas actualmente, al momento de usarlas con problemas reales, que incluyen cientos o miles de características, todavía se nota la lentitud, ya sea al momento de entrenarlas o de recuperar información de ellas.

Por lo anterior mencionado, en esta tesis se presentan nuevos algoritmos que reducen drásticamente el tiempo de aprendizaje y recuperación. Estos nuevos algoritmos están basados en los operadores alfa y beta y por consiguiente ninguna de las cualidades de las memorias es afectada; en otras palabras, se conserva su capacidad de almacenamiento y su tolerancia ante patrones alterados.



# Abstract

The associative memories are computational models that allows to associate input patterns with output patterns, and later recover the corresponding output pattern from the input pattern although this was alterated in some way.

One of the outstanding models of the associative memories are the Alpha Beta associative memories, this are between the bests reported in the actual scientific literature due to their storage capacity, velocity and noise tolerance.

Although the Alpha-Beta associative memories are one of the fastest models of these days, at the moment of use it with real problems, that have thousands of characteristics, still are slow, whichever at the training phase or the recall phase

Due to the above mentionated, in this thesis are proposed new methods that reduce drastically the learning and recall time. This new algorithms are based on the alpha and beta operators and due to this no one of the qualities of the memories is affected, in other words their storage capacity and their tolerance to alterated patterns are conserved.

# CAPÍTULO 1

# Introducción

En esta tesis se desarrollan nuevos algoritmos que permiten, a las memorias asociativas Alfa-Beta, reducir drásticamente la cantidad de tiempo necesario para aprender y recuperar datos. Esta reducción se logra sin sacrificar la eficiencia de estas memorias.

## *1.1 Antecedentes*

En décadas recientes, algunos equipos de investigación científica que desarrollan proyectos relacionados con las ciencias de la computación, han puesto atención al área de las memorias asociativas, y las han incorporado en sus algoritmos, principalmente en tareas concernientes a la teoría del reconocimiento de patrones y a sus aplicaciones.

El propósito fundamental de una memoria asociativa es recuperar correctamente patrones completos a partir de patrones de entrada, los cuales pueden estar alterados respecto de los patrones de aprendizaje [1]. En el diseño de una memoria asociativa, previamente a la fase de recuperación de patrones, se requiere la fase de aprendizaje, que es el proceso mediante el cual se crea la memoria asociativa a través de la formación de asociaciones de patrones, las cuales son parejas de patrones, uno de entrada y uno de salida. Si en cada asociación sucede que el patrón de entrada es igual al de salida, la memoria es autoasociativa; en caso contrario, la memoria es heteroasociativa: esto significa que una memoria autoasociativa puede considerarse como un caso particular de una memoria heteroasociativa [2].

Las memorias asociativas están íntimamente ligadas a las redes neuronales, debido principalmente a que prácticamente todo lo que las redes neuronales pueden hacer también lo pueden hacer las memorias asociativas (a excepción de las regresiones), por lo tanto comparten en gran parte su historia.

El primer modelo de red neuronal conocido es el de McCulloch–Pitts [3], presentado en 1943. Este modelo estaba basado en las redes neuronales cerebrales y era bueno con problemas linealmente separables; debido a esto, las redes neuronales artificiales tuvieron gran auge durante los siguientes años. Posteriormente a partir de este modelo, Rosenblatt desarrolló el perceptron [5], el cual inspiro otros modelos que todavía a principios de la década de los 90 del siglo pasado eran objeto de estudios [6,7].

En 1969 Minsky y Papert, científicos altamente reconocidos en su tiempo (a Minsky se le considera como uno de los padres de la inteligencia artificial), publican el libro *Perceptrons* [9], en el cual se puso a prueba a las redes neuronales tratando de resolver el problema del XOR, pero aunque fuera un problema sencillo, por no ser linealmente separable no se podía resolver, anotando como conclusión en su libro que al no poder resolver un problema tan sencillo, cualquier investigación sobre redes neuronales sería estéril. Esto desalentó a los

investigadores que estaban trabajando en ellas y durante los siguientes 10 años no hubo ningún avance relevante.

En los años 80 del siglo pasado, Hopfield saca su modelo de memoria asociativa la cual era a su vez también una red neuronal [8] basándose en un modelo físico y dándole un fuerte sustento matemático. La gran cualidad de este modelo es que revivió el interés en las redes neuronales; sin embargo, el modelo Hopfield tenía algunos problemas: en primer lugar que sólo es autoasociativa, en segundo lugar, la baja capacidad de almacenamiento y recuperación, la cual es de sólo el 15% de la dimensión de los patrones y finalmente la iteratividad en la fase de salida, en la cual es necesaria una convergencia que podría no llegar a presentarse (este problema fue resuelto por Catalán–Yáñez en el 2006 [48]). Sin embargo, a pesar de los problemas que tenía este modelo dio pie al resurgimiento de la investigación, tanto en las redes neuronales como en las memorias asociativas.

Analizando por otro lado las memorias asociativas, en 1961 aparece la primera: La Lernmatrix [10], desarrollada por Karl Steinbuch, la cual fue una memoria heteroasociativa y fue usada para clasificar patrones binarios.

En 1969 los investigadores escoceses Willshaw, Buneman y Longuet Higgins presentan el *Correlograph* [11], dispositivo óptico elemental capaz de funcionar como una memoria asociativa.

Un 1972 Anderson[12] y Kohonen[13] presentan 2 modelos clásicos de memorias asociativas; sin embargo, debido a las similitudes en los conceptos involucrados ambos reciben el nombre genérico de linear asociator.

Para darle la capacidad a la memoria Hopfield de asociar patrones diferentes, Bart Kosko presenta en 1988 la memoria asociativa bidirecciona [14] (BAM por sus siglas en inglés); sin embargo, este modelo siguió conservando los demás problemas de la memoria Hopfield, así como también los modelos siguientes [15-18]. Avances notables en la capacidad de almacenamiento y tiempo de procesamiento se obtuvieron con la Alfa Beta BAM [21-23] cuando se superaron estos problemas, la cual es capaz de aprender $2^n$ asociaciones, no tiene los problemas de la iteratividad y es capaz de trabajar en modo heteroasociativo.

En 1998 surgen las memorias asociativas morfológicas[19], las cuales dan un giro radical a la forma en que se habían estado concibiendo las memorias asociativas hasta ese momento, debido a que sustituyeron la suma y la resta de productos por máximos y mínimos de sumas y restas. Con esto lograron mejorar significativamente la capacidad de almacenaje, recuperación y el tiempo necesario para ello, superando significativamente a todos los modelos existentes hasta el momento.

Finalmente en el año 2002, surgen las memorias asociativas Alfa Beta [20], inspiradas por las memorias morfológicas, sólo que éstas en vez de usar máximos y mínimos de sumas y restas, usan máximos y mínimos de operaciones lógicas, lo cual les otorga una ventaja en cuanto al tiempo de procesamiento.

## 1.2 Justificación

Como se muestra en el documento original de las memorias asociativas alfa beta, la tesis doctoral del doctor Yáñez-Márquez [20], las memorias asociativas Alfa-Beta igualan, y en ocasiones superan, la capacidad de aprendizaje y almacenamiento de las memorias asociativas morfológicas, además de otras ventajas; lo cual las coloca entre las mejores reportadas en la literatura científica actual; sin embargo, a pesar de estar entre las memorias asociativas mas rápidas de nuestros días, el tiempo de procesamiento se eleva demasiado cuando queremos aprender o recuperar una gran cantidad patrones o incrementamos la dimensión de éstos. El desarrollo de algoritmos más rápidos, tanto de aprendizaje como de recuperación de patrones, justifica este trabajo de tesis.

## 1.3 Objetivo

Desarrollar algoritmos que permitan, a las memorias asociativas Alfa-Beta, aprender y recuperar patrones en una cantidad de tiempo menor, sin perder ninguna de sus ventajas; es decir, conservando su capacidad de almacenamiento y recuperación así como su robustez ante patrones alterados.

## 1.4 Contribuciones

Las contribuciones de este trabajo de tesis son:

- El desarrollo de nuevos algoritmos de aprendizaje que permiten reducir el tiempo de aprendizaje drásticamente, ya sea para una gran cantidad de patrones o para patrones de gran dimensión.

- El desarrollo de nuevos algoritmos de recuperación que permiten reducir el tiempo de recuperacion drásticamente, ya sea para una gran cantidad de patrones o para patrones de gran dimensión.

- El desarrollo del operado $\varphi$, necesario para la generación del operador $\psi$ .y para el desarrollo de teoremas de conversión de las memorias asociativas Alfa-Beta.

- El desarrollo de un teorema que permite convertir una memoria autoasociativa Max a una memoria autoasociativa Min y viceversa, reduciendo el tiempo de aprendizaje a la mitad más el tiempo que se tarde en realizar la conversión.

- El desarrollo de un teorema que permite la conversión de las memorias asociativas Max y Min obtenidas a partir de las asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$, en memorias asociativas Max y Min que proceden de la relacion contraria $(\mathbf{y}^{\mu}, \mathbf{x}^{\mu})$; esto es, la genaración de una BAM a partir de las memorias Min y Max en un sentido.

- El desarrollo del operador $\psi$ (original de esta tesis), el cual nos permite operar una memoria Min autoasociativa para obtener el resultado de la memoria tipo Max y viceversa.

- Lo anterior se puede hacer con las memorias heteroasociativas a partir del operador $\psi$, pero en este caso obtendríamos el resultado de la relacion contraria.

- La generación de una clase para C++ builer, en la cual están implantados tanto los algoritmos originales como los simplificados.

## *1.5 Organización del documento*

En este capítulo se han presentado: los antecedentes, el problema a resolver, el objetivo del trabajo de tesis y las contribuciones. El resto del documento de tesis está organizado de la siguiente manera:

En el capítulo 2 se presentan los conceptos básicos de las memorias asociativas, así como el estado del arte de los modelos más representativos de memorias asociativas previos a las Alfa-Beta.

El capítulo 3 contiene todo lo referente al marco teórico, el cual en este caso está conformado por las memorias asociativas Alfa-Beta. En este capítulo se muestra la forma en que las memorias asociativas funcionan, así como algunos de sus teoremas más interesantes o útiles para esta tesis.

En el capítulo 4 se presentan las aportaciones de esta tesis. Se incluyen las nuevas definiciones y teoremas sobre los que están basados los nuevos algoritmos de aprendizaje y recuperación mostrados en este mismo capítulo.

En el capítulo 5 se muestran comparativas entre los algoritmos originales y los simplificados, haciendo primeramente un análisis entre el número de operaciones necesarias, para posteriormente ilustrar, mediante gráficas, el tiempo que tardaría cada uno de los algoritmos ante determinadas circunstancias.

En el capítulo 6, se presentan las conclusiones obtenidas durante el desarrollo de esta tesis y sugerencias para el trabajo futuro.

Finalmente, se presentan 5 apéndices, el primero con la simbología usada en este documento; el segundo muestra los conceptos básicos mediante un ejemplo; el tercero con los algoritmos de las memorias asociativas clásicas; el cuarto con ejemplos de los algoritmos originales de las memorias asociativas Alfa-Beta y el último contiene el código de los algoritmos implementados en C++; para este propósito se generó una clase, la cual se utilizó durante la experimentación (capítulo 5). Por último, se pone la bibliografía.

# CAPÍTULO 2

# Conceptos básicos y estado del arte

Este capítulo consta de dos secciones: en la primera se presentan los conceptos básicos relacionados con las memorias asociativas, y en la segunda se incluye el estado del arte de los modelos más representativos de memorias asociativas previos a las Alfa-Beta.

## *2.1 Conceptos básicos*

Los conceptos presentados en esta sección se han tomado de las referencias que, a nuestro juicio, son las más representativas [1, 2, 20, 21, 28].

Para poder entender la forma en que funcionan las memorias asociativas Alfa Beta, es necesario estar familiarizado conconceptos tales como característica, patron y conjunto fundamental. Si no esta familiarizado con estos conceptos consulte el apéndice B, en el cual estos conceptos son introducidos a traves de un ejemplo.

A continuacion se da la explicacion de una memoria asociativa, durante esta explicacion se daran las notaciones que se usaran a lo largo de esta tesis.

Una **Memoria Asociativa** puede formularse como un sistema de entrada y salida, idea que se esquematiza a continuación:

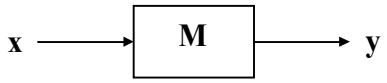



**Fig 2.1** Esquema de una memoria asociativa

En este esquema, los patrones de entrada y salida están representados por vectores columna denotados por **x** y **y**, respectivamente. Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida, la cual es similar a una pareja ordenada; por ejemplo, los patrones **x** y **y** del esquema anterior forman la asociación (**x**,**y**). A continuación se propone una notación que se usará en la descripción de los conceptos básicos sobre memorias asociativas, y en el resto de los capítulos de esta tesis.

Los patrones de entrada y salida se denotarán con las letras negrillas, **x** y **y**, agregándoles números naturales como superíndices para efectos de discriminación simbólica. Por ejemplo, a un patrón de entrada $\mathbf{x}^1$ le corresponderá el patrón de salida $\mathbf{y}^1$, y ambos formarán la asociación $(\mathbf{x}^1,\mathbf{y}^1)$; del mismo modo, para un número entero positivo $k$ específico, la asociación correspondiente será $(\mathbf{x}^k,\mathbf{y}^k)$.

La memoria asociativa **M** se representa mediante una matriz, la cual se genera a partir de un conjunto finito de asociaciones conocidas de antemano: este es el **conjunto fundamental de aprendizaje**, o simplemente **conjunto fundamental**.

El conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$$

donde $p$ es un número entero positivo que representa la cardinalidad del conjunto fundamental.

A los patrones que conforman las asociaciones del conjunto fundamental se les llama **patrones fundamentales**. La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas:

Una memoria es **Autoasociativa** si se cumple que $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \ \forall \mu \in \{1, 2, ..., p\}$, lo que tiene como consecuencia que $n = m$.

Una memoria **Heteroasociativa** es aquella en donde $\exists \mu \in \{1, 2, ..., p\}$ para el que se cumple que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$. Nótese que puede haber memorias heteroasociativas con $n = m$.

En los problemas donde intervienen las memorias asociativas, se consideran dos fases importantes: La fase de aprendizaje, que es donde se genera la memoria asociativa a partir de las $p$ asociaciones del conjunto fundamental, y la fase de recuperación que es donde la memoria asociativa opera sobre un patrón de entrada, a la manera del esquema que aparece al inicio de esta sección.

A fin de especificar las componentes de los patrones, se requiere la notación para dos conjuntos a los que llamaremos arbitrariamente $A$ y $B$. Las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, serán elementos del conjunto $A$, y las entradas de la matriz **M** serán elementos del conjunto $B$.

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Esto significa que el número de posibilidades para escoger $A$ y $B$ es infinito.

Por convención, cada vector columna que representa a un patrón de entrada tendrá $n$ componentes cuyos valores pertenecen al conjunto $A$, y cada vector columna que representa a un patrón de salida tendrá $m$ componentes cuyos valores pertenecen también al conjunto $A$; es decir:

$$\mathbf{x}^{\mu} \in A^n \text{ y } \mathbf{y}^{\mu} \in A^m \ \forall \mu \in \{1, 2, ..., p\}$$

La $j$-ésima componente de un vector columna se indicará con la misma letra del vector, pero sin negrilla, colocando a $j$ como subíndice ($j \in \{1, 2, ..., n\}$ o $j \in \{1, 2, ..., m\}$ según corresponda). La $j$-ésima componente del vector columna $\mathbf{x}^{\mu}$ se representa por: $x_j^{\mu}$

Con los conceptos básicos ya descritos y con la notación anterior, es posible expresar las dos fases de una memoria asociativa:

1. **Fase de Aprendizaje** (Generación de la memoria asociativa). Encontrar los operadores adecuados y una manera de generar una matriz **M** que almacene las $p$ asociaciones del conjunto fundamental $\{(\mathbf{x}^1,\mathbf{y}^1), (\mathbf{x}^2,\mathbf{y}^2), ..., (\mathbf{x}^p,\mathbf{y}^p)\}$, donde $\mathbf{x}^\mu \in A^n$ y $\mathbf{y}^\mu \in A^m \ \forall \mu \in \{1, 2, ..., p\}$. Si $\exists \mu \in \{1, 2, ..., p\}$ tal que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$, la memoria será heteroasociativa; si $m = n$ y $\mathbf{x}^\mu = \mathbf{y}^\mu \ \forall \mu \in \{1, 2, ..., p\}$, la memoria será autoasociativa.

2. **Fase de Recuperación** (Operación de la memoria asociativa). Hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida $\mathbf{y}^\mu$, cuando se opera la memoria **M** con el patrón fundamental de entrada $\mathbf{x}^\mu$; lo anterior para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo.

Se dice que una memoria asociativa **M** exhibe **recuperación correcta** si al presentarle como entrada, en la fase de recuperación, un patrón $\mathbf{x}^\omega$ con $\omega \in \{1, 2, ..., p\}$, ésta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^\omega$.

## 2.2 Estado del arte

A continuación, en esta sección se muestra una tabla con los modelos de memorias asociativas más representativos, los cuales sirvieron de base para la creación de modelos matemáticos que sustentan el diseño y operación de memorias asociativas más complejas. e En el apéndice C se incluyen las fases de aprendizaje y recuperación de todos estos modelos, en forma detallada.

Los cinco modelos clásicos están basados en el anillo de los números racionales con las operaciones de multiplicación y adición: *Lernmatrix*, *Correlograph*, *Linear Associator* , Memoria Hopfield y su secuela, la BAM de Kosko; además, se presentan tres modelos basados en paradigmas diferentes a la suma de productos, a saber: memorias asociativas Morfológicas, memorias asociativas Alfa-Beta y memorias asociativas Mediana.

| Modelo | Año |
|---|---|
| *Lernmatrix* de Steinbuch | 1961 |
| *Correlograph* de Willshaw, Buneman y Longuet-Higgins | 1969 |
| Linear Associator de Anderson-Kohonen | 1972 |
| La memoria asociativa Hopfield | 1982 |
| Memoria Asociativa Bidireccional (BAM) de Kosko. | 1988 |
| Memorias Asociativas Morfológicas | 1998 |
| Memorias Asociativas Alfa-Beta | 2002 |
| Memorias Asociativas Mediana | 2004 |

**Tabla 2.1** Memorias asociativas clásicas

# CAPÍTULO 3

# Materiales y Métodos

En este capítulo se describe el modelo matemático de las Memorias Asociativas Alfa-Beta, las cuales son la base fundamental del algoritmo que da lugar al nuevo modelo de memoria asociativa propuesto en este trabajo de tesis.

## *3.1 Memorias Asociativas Alfa-Beta*

En esta sección se presenta el fundamento teórico que sustenta a las memorias asociativas Alfa-Beta tal como aparece en [20]; para ello, se presentan las definiciones y propiedades de las operaciones $\alpha$ y $\beta$, las operaciones matriciales que se derivan de estas operaciones originales, y se describen las fases de aprendizaje y recuperación de la memorias heteroasociativas y autoasociativas Alfa-Beta, tanto *Max,* denotadas por **M**, como *Min*, denotadas por **W,** ejemplos del funcionamiento de estas memorias pueden ser consultados en el apéndice D.

La numeración de los Lemas y Teoremas que se presentan en este capítulo, corresponde a la numeración original que aparece en la tesis [20].

### 3.1.1 Operaciones binarias $\alpha$ y $\beta$: definiciones y propiedades

Las memorias Alfa-Beta utilizan máximos y mínimos, y dos operaciones binarias originales $\alpha$ y $\beta$ de las cuales heredan el nombre.

Para la definición de las operaciones binarias $\alpha$ y $\beta$ se deben especificar los conjuntos $A$ y $B$, los cuales son:

$$\mathbf{A} = \{0, 1\} \quad \text{y} \quad \mathbf{B} = \{0, 1, 2\}$$

La operación $\alpha$: $\mathbf{A} \times \mathbf{A} \rightarrow \mathbf{B}$ se define como se muestra en la Tabla 1.

| $x$ | $y$ | $\alpha(x, y)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 2 |
| 1 | 1 | 1 |

**Tabla 3.1** Operación $\alpha$: $\mathbf{A} \times \mathbf{A} \rightarrow \mathbf{B}$

La operación $\beta$: **B** x **A** $\rightarrow$ **A** se define como se muestra en la Tabla 2

| $x$ | $y$ | $\beta(x, y)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 0 | 1 |
| 2 | 1 | 1 |

**Tabla 3.2** Operación $\beta$: **B** x **A** $\rightarrow$ A

Los conjuntos **A** y **B,** las operaciones binarias $\alpha$ y $\beta$ junto con los operadores $\wedge$ (mínimo) y $\vee$ (máximo) usuales conforman el sistema algebraico (**A**, **B**, $\alpha$, $\beta$, $\wedge$, $\vee$) en el que están inmersas las memorias asociativas Alfa-Beta [20, 37].

Se requiere la definición de cuatro operaciones matriciales, de las cuales se usarán sólo 4 casos particulares:

Operación $\boldsymbol{\alpha max}$: $P_{mxr} \nabla_{\alpha} Q_{rxn} = \left[ f_{ij}^{\alpha} \right]_{mxn}$, donde $f_{ij}^{\alpha} = \bigvee_{k=1}^{r} \alpha(p_{ik}, q_{kj})$

Operación $\boldsymbol{\beta max}$: $P_{mxr} \nabla_{\beta} Q_{rxn} = \left[ f_{ij}^{\beta} \right]_{mxn}$, donde $f_{ij}^{\beta} = \bigvee_{k=1}^{r} \beta(p_{ik}, q_{kj})$

Operación $\boldsymbol{\alpha min}$: $P_{mxr} \Delta_{\alpha} Q_{rxn} = \left[ h_{ij}^{\alpha} \right]_{mxn}$, donde $h_{ij}^{\alpha} = \bigwedge_{k=1}^{r} \alpha(p_{ik}, q_{kj})$

Operación $\boldsymbol{\beta min}$: $P_{mxr} \Delta_{\beta} Q_{rxn} = \left[ h_{ij}^{\beta} \right]_{mxn}$, donde $h_{ij}^{\beta} = \bigwedge_{k=1}^{r} \beta(p_{ik}, q_{kj})$

El siguiente lema muestra los resultados obtenidos al utilizar las operaciones que involucran al operador binario $\alpha$ con las componentes de un vector columna y un vector fila dados.

**Lema 3.9.** (*Numeración tal como aparece en* [20]). *Sean* $\mathbf{x} \in \mathbf{A}^n$ y $\mathbf{y} \in \mathbf{A}^m$*; entonces* $\mathbf{y} \nabla_{\alpha} \mathbf{x}^t$ *es una matriz de dimensiones m x n, y además se cumple que:* $\mathbf{y} \nabla_{\alpha} \mathbf{x}^t = \mathbf{y} \Delta_{\alpha} \mathbf{x}^t$.

**Demostración.**

$$\mathbf{y} \nabla_{\alpha} \mathbf{x}^t = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \nabla_{\alpha} (x_1, x_2, \ldots, \mathrm{x_n})$$

$$= \begin{pmatrix} \bigvee_{k=1}^{1} \alpha(y_1, x_1) & \bigvee_{k=1}^{1} \alpha(y_1, x_2) & \cdots & \bigvee_{k=1}^{1} \alpha(y_1, x_n) \\ \bigvee_{k=1}^{1} \alpha(y_2, x_1) & \bigvee_{k=1}^{1} \alpha(y_2, x_2) & \cdots & \bigvee_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \bigvee_{k=1}^{1} \alpha(y_m, x_1) & \bigvee_{k=1}^{1} \alpha(y_m, x_2) & \cdots & \bigvee_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \begin{pmatrix} \alpha(y_1, x_1) & \alpha(y_1, x_2) & \cdots & \alpha(y_1, x_n) \\ \alpha(y_2, x_1) & \alpha(y_2, x_2) & \cdots & \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \alpha(y_m, x_1) & \alpha(y_m, x_2) & \cdots & \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \begin{pmatrix} \bigwedge_{k=1}^{1} \alpha(y_1, x_1) & \bigwedge_{k=1}^{1} \alpha(y_1, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_1, x_n) \\ \bigwedge_{k=1}^{1} \alpha(y_2, x_1) & \bigwedge_{k=1}^{1} \alpha(y_2, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \bigwedge_{k=1}^{1} \alpha(y_m, x_1) & \bigwedge_{k=1}^{1} \alpha(y_m, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \mathbf{y} \Delta_{\alpha} \mathbf{x}^{t}$$

En efecto, resulta que $\mathbf{y} \nabla_{\alpha} \mathbf{x}^{t}$ es una matriz de dimensiones *m* x *n*, y que $\mathbf{y} \nabla_{\alpha} \mathbf{x}^{t} = \mathbf{y} \Delta_{\alpha} \mathbf{x}^{t}$.

Dado el resultado del lema anterior, es conveniente escoger un símbolo único, digamos el símbolo $\otimes$, que represente a las dos operaciones $\nabla_{\alpha}$ y $\Delta_{\alpha}$ cuando se opera un vector columna de dimensión *m* con un vector fila de dimensión *n*:

$$\mathbf{y} \nabla_{\alpha} \mathbf{x}^{t} = \mathbf{y} \otimes \mathbf{x}^{t} = \mathbf{y} \Delta_{\alpha} \mathbf{x}^{t}$$

La *ij*-ésima componente de la matriz está $\mathbf{y} \otimes \mathbf{x}^{t}$ dada por:

$$[\mathbf{y} \otimes \mathbf{x}^{t}]_{ij} = \alpha(y_i, x_j)$$

Dado un índice de asociación $\mu$, la expresión anterior indica que la *ij*-ésima componente de la matriz $\mathbf{y}^{\mu} \otimes (\mathbf{x}^{\mu})^{t}$ se expresa de la siguiente manera:

$$[\mathbf{y}^{\mu} \otimes (\mathbf{x}^{\mu})^{t}]_{ij} = \alpha\left(y_i^{\mu}, x_j^{\mu}\right)$$

Ahora se analizará el caso en el que se opera una matriz de dimensiones *m* x *n* con un vector columna de dimensión *n* usando las operaciones$\nabla_{\beta}$ y $\Delta_{\beta}$. En los lemas 3.11 y 3.12 se obtiene la forma que exhibirán las *i*-ésimas componentes de los vectores columna resultantes de dimensión *m*, a partir de ambas operaciones $\nabla_{\beta}$ y $\Delta_{\beta}$.

**Lema 3.11.** (*Numeración tal como aparece en* [20]). *Sean* $\mathbf{x} \in \mathbf{A}^n$ y $\mathbf{P}$ *una matriz de dimensiones m x n. La operación* $\mathbf{P}_{mxn} \nabla_\beta \mathbf{x}$ *da como resultado un vector columna de dimensión m, cuya i-ésima componente tiene la siguiente forma:* $\left(\mathbf{P}_{mxn} \nabla_\beta \mathbf{x}\right)_i = \bigvee_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$

**Demostración.**

$$\mathbf{P}_{mxn} \nabla_\beta \mathbf{x} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \nabla_\beta \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{P}_{mxn} \nabla_\beta \mathbf{x} = \begin{pmatrix} \beta(p_{11}, x_1) \vee \beta(p_{12}, x_2) \vee \ldots \vee \beta(p_{1n}, x_n) \\ \beta(p_{21}, x_1) \vee \beta(p_{22}, x_2) \vee \ldots \vee \beta(p_{2n}, x_n) \\ \vdots \\ \beta(p_{m1}, x_1) \vee \beta(p_{m2}, x_2) \vee \ldots \vee \beta(p_{mn}, x_n) \end{pmatrix} = \begin{pmatrix} \bigvee_{j=1}^{n} \beta(p_{1j}, x_j) \\ \bigvee_{j=1}^{n} \beta(p_{2j}, x_j) \\ \vdots \\ \bigvee_{j=1}^{n} \beta(p_{mj}, x_j) \end{pmatrix}$$

Se obtiene un vector columna de dimensión *m* cuya *i*-ésima componente es

$$\left(\mathbf{P}_{mxn} \nabla_\beta \mathbf{x}\right)_i = \bigvee_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$$

**Lema 3.12.** (*Numeración tal como aparece en* [20]). *Sean* $\mathbf{x} \in \mathbf{A}^n$ y $\mathbf{P}$ *una matriz de dimensiones m x n. La operación* $\mathbf{P}_{mxn} \Delta_\beta \mathbf{x}$ *da como resultado un vector columna de dimensión m, cuya i-ésima componente tiene la siguiente forma:* $\left(\mathbf{P}_{mxn} \Delta_\beta \mathbf{x}\right)_i = \bigwedge_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$

**Demostración.**

$$\mathbf{P}_{mxn} \Delta_\beta \mathbf{x} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \Delta_\beta \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{P}_{mxn} \Delta_\beta \mathbf{x} = \begin{pmatrix} \beta(p_{11}, x_1) \wedge \beta(p_{12}, x_2) \wedge \ldots \wedge \beta(p_{1n}, x_n) \\ \beta(p_{21}, x_1) \wedge \beta(p_{22}, x_2) \wedge \ldots \wedge \beta(p_{2n}, x_n) \\ \vdots \\ \beta(p_{m1}, x_1) \wedge \beta(p_{m2}, x_2) \wedge \ldots \wedge \beta(p_{mn}, x_n) \end{pmatrix} = \begin{pmatrix} \bigwedge_{j=1}^{n} \beta(p_{1j}, x_j) \\ \bigwedge_{j=1}^{n} \beta(p_{2j}, x_j) \\ \vdots \\ \bigwedge_{j=1}^{n} \beta(p_{mj}, x_j) \end{pmatrix}$$

Se obtiene un vector columna de dimensión *m* cuya *i*-ésima componente es

$$\left(\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x}\right)_i = \bigwedge_{j=1}^{n}\beta\left(p_{ij}, x_j\right)$$

### 3.1.2 Memorias Heteroasociativas Alfa-Beta

Se tienen dos tipos de memorias heteroasociativas Alfa-Beta: tipo Max denotada por **M** y su ij-ésima componente como $m_{ij}$ y las tipo Min, denotadas por **W** y su ij-ésima componente como $w_{ij}$. En la generación de ambos tipos de memorias se usará el operador $\otimes$ el cual tiene la siguiente forma:

$$\left[y^{\mu}\otimes\left(x^{\mu}\right)^t\right]_{ij} = \alpha\left(y_i^{\mu}, x_j^{\mu}\right), \mu\in\{1,2,\cdots,p\}, i\in\{1,2,\cdots,m\}, j\in\{1,2,\cdots,n\}$$

**Algoritmo Memorias Alfa-Beta Tipo Max**

**Fase de Aprendizaje**

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se construye la matriz

$$\left[\mathbf{y}^{\mu}\otimes\left(\mathbf{x}^{\mu}\right)^t\right]_{mxn}$$

**Paso 2.** Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso 1:

$$\mathbf{M} = \bigvee_{\mu=1}^{p}\left[\mathbf{y}^{\mu}\otimes\left(\mathbf{x}^{\mu}\right)^t\right]$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$m_{ij} = \bigvee_{\mu=1}^{p}\alpha\left(y_i^{\mu}, x_j^{\mu}\right)$$

**Fase de Recuperación**

Se presenta un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$, a la memoria heteroasociativa $\alpha\beta$ tipo **M** y se realiza la operación $\Delta_{\beta}$: $\mathbf{M}\ \Delta_{\beta}\ \mathbf{x}^{\omega}$.

Dado que las dimensiones de la matriz **M** son de *m* x *n* y $\mathbf{x}^{\omega}$ es un vector columna de dimensión *n*, el resultado de la operación anterior debe ser un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$\left(\mathbf{M}\Delta_{\beta}\mathrm{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n}\beta\left(m_{ij}, x_j^{\omega}\right)$$

**Teorema 4.7.** (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,\ldots,p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ *representada por* $\mathbf{M}$. *Si* $\omega$ *es un valor de índice arbitrario tal que* $\omega \in \{1,2,\ldots,p\}$ *, y si además para cada* $i \in \{1,\ldots,m\}$ *se cumple que* $\exists j = j_0 \in \{1,\ldots,n\}$ *, el cual depende de* $\omega$ *y de* $i$ *, tal que* $m_{ij_0} = \alpha\left(y_i^\omega, x_{j_0}^\omega\right)$ *, entonces la recuperación* $\mathbf{M}\Delta_\beta \mathbf{x}^\omega$ *es correcta; es decir* $\mathbf{M}\Delta_\beta \mathbf{x}^\omega = \mathbf{y}^\mu$.

**Demostración.** Ver detalles de la demostración en [20].

## Algoritmo Memorias Alfa-Beta Tipo Min

### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se construye la matriz

$$\left[\mathbf{y}^\mu \otimes \left(\mathbf{x}^\mu\right)^t\right]_{mxn}$$

**Paso 2.** Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas en el paso 1:

$$\mathbf{W} = \bigwedge_{\mu=1}^{p}\left[\mathbf{y}^\mu \otimes \left(\mathbf{x}^\mu\right)^t\right]$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$w_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right)$$

### Fase de Recuperación

Se presenta un patrón $\mathbf{x}^\omega$, con $\omega \in \{1, 2, ..., p\}$, a la memoria heteroasociativa $\alpha\beta$ tipo Min y se realiza la operación $\nabla_\beta$: $\mathbf{W} \nabla_\beta \mathbf{x}^\omega$.

Dado que las dimensiones de la matriz Min $\mathbf{W}$ son de *m* x *n* y $\mathbf{x}^\omega$ es un vector columna de dimensión *n*, el resultado de la operación anterior debe ser un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$\left(\mathbf{W}\nabla_\beta \mathbf{x}^\omega\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^\mu\right)$$

**Teorema 4.20.** (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,\ldots,p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ *Min representada por* $\mathbf{W}$. *Si* $\omega$ *es un valor arbitrario fijo tal que* $\omega \in \{1,2,\ldots,p\}$ *, y si además para cada* $i \in \{1,\ldots,m\}$ *se cumple que* $\exists j = j_0 \in \{1,\ldots,n\}$ *, el cual depende de* $\omega$ *y de* $i$ *, tal*

*que* $w_{ij} = \alpha\left(y_i^{\omega}, x_{j_0}^{\omega}\right)$, *entonces la recuperación* $\mathbf{W}\nabla_{\beta}\mathbf{x}^{\omega}$ *es correcta; es decir* $\mathbf{W}\nabla_{\beta}\mathbf{x}^{\omega} = \mathbf{y}^{\omega}$.

**Demostración.** Ver detalles de la demostración en [20].

### Algoritmo Memorias Alfa-Beta combinado

#### Fase de Aprendizaje

Como se puede observar el paso 1 es el mismo para los 2 tipos de memorias, asi que si queremos aprender las 2, este paso solo se hace una vez y se realiza el paso 2 de ambas memorias, entonces el algoritmo queda como se muestra a continuación:

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se construye la matriz

$$\left[\mathbf{y}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]_{mxn}$$

**Paso 2.** Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso 1:

$$\mathbf{M} = \bigvee_{\mu=1}^{p}\left[\mathbf{y}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]_{mxn}$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$m_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(y_i^{\mu}, x_j^{\mu}\right)$$

**Paso 3.** Se aplica el operador binario mínimo Min a las matrices obtenidas en el paso 1:

$$\mathbf{W} = \bigwedge_{\mu=1}^{p}\left[\mathbf{y}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]_{mxn}$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$w_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(y_i^{\mu}, x_j^{\mu}\right)$$

En la fase de recuperación los algoritmos se mantienen sin cambios

### 3.1.3 Memorias Autoasociativas Alfa-Beta

Si a una memoria heteroasociativa se le impone la condición de que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}\ \forall\mu \in \{1, 2, ..., p\}$ entonces, deja de ser heteroasociativa y ahora se le denomina autoasociativa.

A continuación se enlistan algunas de las características de las memorias autoasociativas Alfa-Beta :

1. El conjunto fundamental toma la forma $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$
2. Los patrones fundamentales de entrada y salida son de la misma dimensión; denotémosla por $n$.
3. La memoria es una matriz cuadrada, para ambos tipos, $\mathbf{V}$ y $\mathbf{\Lambda}$. Si $\mathbf{x}^{\mu} \in \mathbf{A}^{n}$ entonces

$$\mathbf{M} = \lfloor m_{ij} \rfloor_{nxn} \text{ y } \mathbf{W} = \lfloor w_{ij} \rfloor_{nxn}$$

## Algoritmo Memorias Autoasociativas Alfa-Beta tipo Max

Las fases de aprendizaje y recuperación son similares a las memorias heteroasociativas Alfa-Beta.

### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ se construye la matriz

$$\left[\mathbf{x}^{\mu} \otimes (\mathbf{x}^{\mu})^{t}\right]_{nxn}$$

**Paso 2.** Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso 1:

$$\mathbf{M} = \bigvee_{\mu=1}^{p} \left[\mathbf{x}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]_{mxn}$$

La entrada $ij$-ésima de la memoria está dada así:

$$m_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$$

y de acuerdo con que $\alpha$: $\mathbf{A}$ x $\mathbf{A} \rightarrow \mathbf{B}$, se tiene que $m_{ij} \in \boldsymbol{B}$, $\forall i \in \{1, 2, ..., n\}$. $\forall j \in \{1, 2, ..., n\}$.

**Fase de Recuperación.** La fase de recuperación de las memorias autoasociativas Alfa-Beta tipo Max tiene dos casos posibles. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino la versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\tilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, ..., p\}$, que corresponde al patrón fundamental respecto del cual $\tilde{\mathbf{x}}$ es una versión alterada con alguno de los tres tipos de ruido: aditivo, sustractivo o mezclado.

**CASO 1: Patrón fundamental.** Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa Alfa-Beta tipo Max, denotada por $\mathbf{M}$ y se realiza la operación $\Delta_{\beta}$:

$$\mathbf{M}\Delta_\beta \mathbf{x}^\omega$$

El resultado de la operación anterior será el vector columna de dimensión *n*.

$$\left(\mathbf{M}\Delta_\beta \mathbf{x}^\omega\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j^\mu\right)$$

$$\left(\mathbf{M}\Delta_\beta \mathbf{x}^\omega\right)_i = \bigwedge_{j=1}^{n} \beta\left[\left[\bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_j^\mu\right)\right], x_j^\omega\right]$$

**CASO 2: Patrón alterado.** Se presenta el patrón binario $\tilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^\omega$) que es un vector columna de dimensión *n*, a la memoria autoasociativa Alfa-Beta tipo V y se realiza la operación

$$\mathbf{M}\Delta_\beta \tilde{\mathbf{x}}$$

Al igual que en el caso 1, el resultado de la operación anterior es un vector columna de dimensión *n*, cuya *i*-ésima componente se expresa de la siguiente manera:

$$\left(\mathbf{M}\Delta_\beta \tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, \tilde{x}_j\right)$$

$$\left(\mathbf{M}\Delta_\beta \tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left[\left[\bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_j^\mu\right)\right], \tilde{x}_j\right]$$

**Lema 4.27.** (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo Max tiene únicamente unos en la diagonal principal*.

**Demostración**. La *ij*-ésima entrada de una memoria autoasociativa Alfa-Beta tipo Max **M** está dada por $m_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_j^\mu\right)$. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$:

$$m_{ii} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_i^\mu\right), \forall i \in \{1,2,...n\}$$

Por la propiedad de la tabla se tiene que $\alpha(x_i^\mu, x_i^\mu) = 1$, por lo que la expresión anterior se transforma en:

$$m_{ii} = \bigvee_{\mu=1}^{p} (1), \forall i \in \{1,2,...n\}$$

**Teorema 4.28.** (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo* Max **M** *recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje.*

**Demostración.** Sea $\omega = \{1, 2, ..., p\}$ arbitrario. De acuerdo con el lema 4.27, para cada $i \in \{1, ..., n\}$ escogida arbitrariamente

$$m_{ii} = 1 = \alpha\left(x_i^{\omega}, x_i^{\omega}\right)$$

Es decir, para $i \in \{1, ..., n\}$ escogida arbitrariamente, $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$m_{ij_0} = 1 = \alpha\left(x_i^{\omega}, x_{j_0}^{\omega}\right)$$

Por lo tanto, de acuerdo con el Teorema 4.7

$$\mathbf{M}\Delta_{\beta}\mathbf{x}^{\omega} = \mathbf{x}^{\omega}, \forall\, \omega \in \{1,2,\cdots, p\}$$

Esto significa que la memoria autoasociativa Alfa-Beta tipo Max recupera de manera correcta el conjunto fundamental completo.

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre *p,* que es la cardinalidad del conjunto fundamental; y esto quiere decir que el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo V, con recuperación correcta, es máximo.

El Teorema 4.28 se puede enunciar desde un enfoque matricial de la siguiente manera: dado que para cada asociación ($\mathbf{x}^{\omega}$, $\mathbf{x}^{\omega}$) del conjunto fundamental de una memoria autoasociativa Alfa-Beta **M**, se cumple que cada fila de la matriz $\mathbf{M} - \mathbf{x}^{\omega} \otimes (\mathbf{x}^{\omega})^t$ contiene una entrada cero, entonces de la memoria V recupera el conjunto completo de patrones fundamentales en forma correcta.

**Teorema 4.30.** (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta representada por* **M**, *y sea* $\tilde{\mathbf{x}} \in \mathbf{A}^n$ *un patrón alterado con ruido aditivo respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* **M** *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* $\omega$ *y de i tal* que $m_{ij} \leq \alpha\left(x^{\omega}, \tilde{x}_{j_0}\right)$, *entonces la recuperación* $\mathbf{M}\Delta_{\beta}\tilde{\mathbf{x}}$ *es correcta, es decir,* $\mathbf{M}\Delta_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

**Demostración.** Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}\ \forall \mu \in \{1, 2, ..., p\}$ y, por consiguiente, $m = n$. Al establecer estas dos condiciones en el Teorema 4.13 (Ver referencia), se obtiene el resultado: $\mathrm{M}\Delta_{\beta}\tilde{\mathbf{x}} = \mathrm{x}^{\omega}$

El Teorema 4.30 nos dice que las memorias autoasociativas Alfa-Beta tipo Max son inmunes a cierta cantidad de ruido aditivo.

## Algoritmo Memorias Autoasociativas Alfa-Beta tipo Max

### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ se construye la matriz

$$\left[\mathbf{x}^{\mu} \otimes (\mathbf{x}^{\mu})^{t}\right]_{nxn}$$

**Paso 2.** Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas en el paso 1, para obtener la memoria Min $\mathbf{W}$:

$$\mathbf{W} = \bigwedge_{\mu=1}^{p}\left[\mathbf{x}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]$$

La entrada $ij$-ésima de la memoria está dada así:

$$w_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$$

y de acuerdo con que $\alpha$: $A$ x $A \rightarrow B$, se tiene que $\lambda_{ij} \in B$, $\forall i \in \{1, 2, ..., n\}$. $\forall j \in \{1, 2, ..., n\}$.

**Fase de Recuperación.** La fase de recuperación de las memorias autoasociativas $\alpha\beta$ tipo $\Lambda$ tiene dos casos posibles. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino la versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\tilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, ..., p\}$, que corresponde al patrón fundamental respecto del cual $\tilde{\mathbf{x}}$ es una versión alterada de alguno de los tres tipos: aditivo, sustractivo o mezclado.

**CASO 1: Patrón fundamental.** Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa $\alpha\beta$ tipo Min, denotada por $\mathbf{W}$ y se realiza la operación $\nabla_{\beta}$:

$$\mathbf{W}\ \Delta_{\beta}\ \mathbf{x}^{\omega}$$

El resultado de la operación anterior será el vector columna de dimensión $n$.

$$\left(\mathbf{W}\nabla_{\beta}\mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^{\mu}\right)$$

$$\left(\mathbf{W}\nabla_{\beta}\mathrm{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left[\left[\bigwedge_{\mu=1}^{n} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)\right], x_j^{\omega}\right]$$

**CASO 2: Patrón alterado.** Se presenta el patrón binario $\tilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^{\omega}$) que es un vector columna de dimensión $n$, a la memoria autoasociativa $\alpha\beta$ tipo $\Lambda$ y se realiza la operación:

$$\mathbf{W}\nabla_{\beta}\tilde{\mathbf{x}}$$

Al igual que en el caso 1, el resultado de la operación anterior es un vector columna de dimensión $n$, cuya $i$-ésima componente se expresa de la siguiente manera:

$$\left(\mathbf{W}\nabla_{\beta}\tilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, \tilde{x}_j\right)$$

$$\left(\mathbf{W}\nabla_{\beta}\tilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta\left[\left[\bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)\right], \tilde{x}_j\right]$$

**Lema 4.31.** (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo* **W** *tiene únicamente unos en la diagonal principal*.

**Demostración**. La $ij$-ésima entrada de una memoria autoasociativa Alfa-Beta tipo Min **W** está dada por $w_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$:

$$w_{ii} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_i^{\mu}\right), \forall i \in \{1,2,\cdots,n\}$$

Por la propiedad de la tabla se tiene que $\alpha(x_i^{\mu}, x_i^{\mu}) = 1$, por lo que la expresión anterior se transforma en:

$$w_{ii} = \bigwedge_{\mu=1}^{p} \alpha(1), \forall i \in \{1,2,\cdots,n\}$$

**Teorema 4.32.** (*Numeración tal como aparece en* [20]). *Una memoria autoasociativa Alfa-Beta tipo Min* **W** *recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje.*

**Demostración.** Sea $\omega = \{1, 2, ..., p\}$ arbitrario. De acuerdo con el lema 4.31, para cada $i \in \{1, ..., n\}$ escogida arbitrariamente $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$w_{ii} = 1 = \alpha\left(x_i^{\omega}, x_i^{\omega}\right)$$

Es decir, para $i \in \{1, ..., n\}$ escogida arbitrariamente, $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$w_{ij_0} = \alpha\left(x_i^{\omega}, x_{j_0}^{\omega}\right)$$

Por lo tanto, de acuerdo con el Teorema 4.20

$$\mathbf{W} \nabla_{\beta} \mathbf{x}^{\omega} = \mathbf{x}^{\omega}, \forall\, \omega \in \{1, 2, ..., p\}$$

Esto significa que la memoria autoasociativa Alfa-Beta tipo Min recupera de manera correcta el conjunto fundamental completo.

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre $p$ que es la cardinalidad del conjunto fundamental; y esto quiere decir que el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo Min **W**, con recuperación correcta, es máximo.

El Teorema 4.32 se puede enunciar desde un enfoque matricial de la siguiente manera: dado que para cada asociación $(\mathbf{x}^{\omega}, \mathbf{x}^{\omega})$ del conjunto fundamental de una memoria autoasociativa Alfa-Beta Min **W**, se cumple que cada fila de la matriz $\mathbf{x}^{\omega} \otimes (\mathbf{x}^{\omega})^t$ - **W** contiene una entrada cero, entonces de la memoria **W** recupera el conjunto completo de patrones fundamentales en forma correcta.

**Teorema 4.33.** (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta Min representada por* **W**, *y sea* $\tilde{\mathbf{x}} \in A^n$ *un patrón alterado con ruido sustractivo respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* **W** *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* $\omega$ *y de i tal que* $w_{ij_0} \leq \alpha\left(x^{\omega}, \tilde{x}_{j_0}\right)$, *entonces la recuperación* $\mathbf{W} \nabla_{\beta} \tilde{\mathbf{x}}$ *es correcta, es decir,* $\mathbf{W} \nabla_{\beta} \tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

**Demostración.** Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}\ \forall \mu \in \{1, 2, ..., p\}$ y, por consiguiente, $m = n$. Al establecer estas dos condiciones en el Teorema 4.23 (Ver referencia), se obtiene el resultado: $\mathbf{W} \nabla_{\beta} \tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

El Teorema 4.33 nos dice que las memorias autoasociativas Alfa-Beta tipo Min son inmunes a cierta cantidad de ruido sustractivo.

### 3.1.4 Memorias Asociativas Alfa Beta Bidireccionales

Cuando entrenamos una memoria sociativa Alfa-beta, mediante las asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ la entrenamos en un sentido y somos capaces de recuperar el patrón de salida $\mathbf{y}^{\mu}$ a partir del patrón de entrada $\mathbf{x}^{\mu}$, si queremos tener también la capacidad de recuperar $\mathbf{x}^{\mu}$ a partir

de $\mathbf{y}^{\mu}$ entonces también tenemos que aprender las memorias en el sentido inverso; esto es, la que contiene las relaciones $(\mathbf{y}^{\mu}, \mathbf{x}^{\mu})$.

Una memoria que es capaz de recuperar tanto en un sentido como en el otro se conoce como memoria asociativa bidireccional, las cuales son de 2 tipos Min y Max. Gráficamente se pueden representar como:

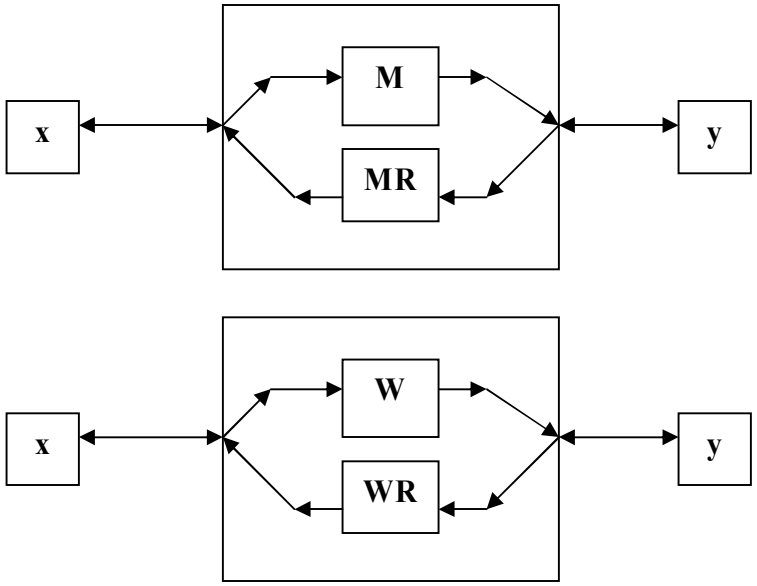



**Fig 3.1** Esquema de una memoria BAM

Los algoritmos de aprendizaje no cambian en absoluto. Primero se realiza el aprendizaje en un sentido, introduciendo las asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$, para obtener las memorias **M** y **W;** posteriormente se realiza el aprendizaje en el otro sentido, introduciendo las asociaciones $(\mathbf{y}^{\mu}, \mathbf{x}^{\mu})$, para obtener las memorias **MR** y **WR.**

El algoritmo de recuperación tiene un paso extra, que se realiza al principio: tiene que determinar en qué sentido se operará la memoria, y esto se logra comparando la dimensión de los patrones; en caso de que la dimension del patrón de salida sea igual al de entrada, deberá dar el resultado en ambas direcciones.

# CAPÍTULO 4

# Modelo Propuesto

Este capítulo es el más relevante del presente documento de tesis. Aquí se dan las nuevas definiciones y teoremas necesarios para los nuevos algoritmos de aprendizaje y recuperación, los cuales, como se muestra en la siguiente sección, demuestran ser más rápidos que los originales. En esta sección también se muestra un algoritmo que convierte una memoria autoasociativa Max en una Min y viceversa.

## *4.1 Definiciones*

Para poder realizar los nuevos algoritmos de aprendizaje y recuperación, es necesario dar las siguientes definiciones; algunas nos ayudarán en la elaboración de teoremas, y otras nos ayudarán en los nuevos algoritmos.

***Definición 1***, Sea $\mathbf{x}^{\mu}$ un patrón de entrada de dimensión $n$ y sea $q$ un número entero positivo tal que $0 \leq q \leq n$ que indica el número de ceros en el patrón $\mathbf{x}^{\mu}$. Se define el conjunto de posiciones de ceros en $\mathbf{x}^{\mu}$, denotado por $\mathbf{X0s}^{\mu}$ y su i-ésima componente por $X0s_i^{\mu}$, como un conjunto de enteros positivos de dimensión $q$ que contiene todas las posiciones con valor 0 en el vector de entrada $\mathbf{x}^{\mu}$. Esto es:

$$\mathbf{X0s}^{\mu} = \left\{X0s_1^{\mu}, X0s_2^{\mu}, \cdots, X0s_i^{\mu}, \cdots, X0s_q^{\mu}\right\}, donde\ X0s_i^{\mu} \in \{1,2,\cdots,n\}\ y\ \mathbf{x}^{\mu}_{X0s_i^{\mu}} = 0$$

A continuación se muestran algunos ejemplos para patrones de dimensión 4, esto es n=4

$$\mathbf{x}^1 = \begin{pmatrix}1\\0\\1\\0\end{pmatrix} \quad \begin{matrix}\mathbf{X0s}^1 = \{2,4\}\\ q = 2\end{matrix} \qquad \mathbf{x}^2 = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad \begin{matrix}\mathbf{X0s}^2 = \{2,3,4\}\\ q = 3\end{matrix} \qquad \mathbf{x}^3 = \begin{pmatrix}1\\1\\1\\1\end{pmatrix} \quad \begin{matrix}\mathbf{X0s}^3 = \phi\\ q = 0\end{matrix}$$

***Definición 2***, Sea $\mathbf{x}^{\mu}$ un patrón de entrada de dimensión $n$ y sea $r$ un número entero positivo tal que $0 \leq r \leq n$ que indica el número de unos en el patrón $\mathbf{x}^{\mu}$. Se define el conjunto de posiciones de unos en $\mathbf{x}^{\mu}$, denotado por $\mathbf{X1s}^{\mu}$ y su i-ésima componente como $X1s_i^{\mu}$. como un conjunto de enteros positivos de dimensión $r$ que contiene todas las posiciones con valor 1 en el vector de entrada $\mathbf{x}^{\mu}$. Esto es:

$$\mathbf{X1s}^{\mu} = \left\{X1s_1^{\mu}, X1s_2^{\mu}, \cdots, X1s_i^{\mu}, \cdots, X1s_r^{\mu}\right\}, donde\ X1s_i^{\mu} \in \{1,2,\cdots,n\}\ y\ \mathbf{x}^{\mu}_{X1s_i^{\mu}} = 1$$

A continuación se muestran algunos ejemplos para patrones de dimensión 4, esto es n=4

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{X1s}^1 = \{1,3\} \quad r = 2 \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{X1s}^2 = \{1\} \quad r = 1 \qquad \mathbf{x}^3 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{X1s}^3 = \{1,2,3,4\} \quad r = 4$$

**Nota**: dado que el vector **x** es binario y por tanto sólo puede tener los valores 0 y 1, la suma de la magnitud de los conjuntos $\mathbf{X0s}^\mu$ y $\mathbf{X1s}^\mu$ es igual a la dimensión del vector **x**. Esto es $\left|\mathbf{X0s}^\mu\right| + \left|\mathbf{X1s}^\mu\right| = \left|\mathbf{x}^\mu\right|$; además, dado que $\left|\mathbf{X0s}^\mu\right| = q, \left|\mathbf{X1s}^\mu\right| = r \; y \; \left|\mathbf{x}^\mu\right| = n$, tenemos que $q + r = n$

***Definición 3***, Sea $\mathbf{y}^\mu$ un patrón de salida de dimensión *m* y sea *s* un número entero positivo tal que $0 \le s \le n$ que indica el número de ceros en el patrón $\mathbf{y}^\mu$. Se define el conjunto de posiciones de ceros en $\mathbf{y}^\mu$, denotado por $\mathbf{Y0s}^\mu$ y su i-ésima componente por $Y0s_i^\mu$, como un conjunto de enteros positivos de dimensión *s* que contiene todas las posiciones con valor 0 en el patrón de salida $\mathbf{y}^\mu$. Esto es:

$$\mathbf{Y0s}^\mu = \left\{Y0s_1^\mu, Y0s_2^\mu, \cdots, Y0s_i^\mu, \cdots, Y0s_s^\mu\right\}, donde \; Y0s_i^\mu \in \{1,2,\cdots,m\} \; y \; \mathbf{y}^\mu_{Y0s_i^\mu} = 0$$

A continuación se muestran algunos ejemplos para patrones de dimensión 3, esto es n=3

$$\mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{Y0s}^1 = \{2\} \quad s = 1 \qquad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{Y0s}^2 = \{1,2,3\} \quad s = 3 \qquad \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{Y0s}^3 = \{2,3\} \quad s = 2$$

***Definición 4***, Sea $\mathbf{y}^\mu$ un patrón de salida de dimensión *m* y sea *t* un número entero positivo tal que $0 \le t \le n$ que indica el número de unos en el patrón $\mathbf{y}^\mu$. Se define el vector de posiciones de unos en $\mathbf{y}^\mu$, denotado por $\mathbf{Y1s}^\mu$ y su i-ésima componente por $Y1s_i^\mu$, como un vector de enteros positivos de dimensión *t* que contiene todas las posiciones con valor 0 en el patrón de salida $\mathbf{Y1s}^\mu$. Esto es:

$$\mathbf{Y1s}^\mu = \left\{Y1s_1^\mu, Y1s_2^\mu, \cdots, Y1s_i^\mu, \cdots, Y1s_t^\mu\right\}, donde \; Y1s_i^\mu \in \{1,2,\cdots,m\} \; y \; \mathbf{y}^\mu_{Y1s_i^\mu} = 1$$

$$\mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{Y1s}^1 = \{1,3\} \quad t = 2 \qquad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{Y1s}^2 = \phi \quad t = 0 \qquad \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{Y1s}^3 = \{1\} \quad s = 1$$

**Nota**: dado que el vector **y** es binario y por tanto solo puede tener los valores de 0 y 1, la suma de la magnitud de los conjunto $\mathbf{Y0s}^\mu$ y $\mathbf{Y1s}^\mu$ es igual a la dimensión del vector **y**.

Esto es $\left|\mathbf{Y0s}^{\mu}\right|+\left|\mathbf{Y1s}^{\mu}\right|=\left|\mathbf{y}^{\mu}\right|$ dado que $\left|\mathbf{Y0s}^{\mu}\right|=s,\left|\mathbf{Y1s}^{\mu}\right|=t\ y\ \left|\mathbf{y}^{\mu}\right|=m$, tenemos que $s+t=m$

***Definición 5***, Sea $\mathbf{x}^{\mu} \mid \mu \in \{1, 2, \ldots, p\}$ el conjunto fundamental de patrones de entrada de dimensión *n*. Se define el vector de existencia de ceros en $\mathbf{x}^{\mu}$ o vector mínimo de los patrones $\mathbf{x}^{\mu}$, denotado por **EX0s** y su i-ésima componente por $EX0s_i$, como un vector binario que indica si en la posición i ha existido algún 0 para algún patrón miembro del conjunto fundamental; es el equivalente a hacer la operación mínimo entre todos los patrones de entrada del conjunto fundamental y sacar el complemento al vector resultante. Esto es:

$$\mathbf{EX0s} = [EX0s_1, EX0s_2, \cdots, EX0s_i, \cdots, EX0s_n]$$

Donde:

$$EX0s_i = \begin{cases} 1 & \text{si } x_i^{\mu} = 0 \text{ para algun } \mu \\ 0 & \text{de otra forma} \end{cases}$$

Ejemplo, del siguiente conjunto fundamental obtendremos **EX0s**

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\mathbf{EX0s} = [0,1,1,1]$$

***Definición 6***, Sea $\mathbf{x}^{\mu} \mid \mu \in \{1, 2, \ldots, p\}$ el conjunto fundamental de patrones de entrada de dimensión *n*. se define el vector de existencia de unos en $\mathbf{x}^{\mu}$ o vector máximo de los patrones $\mathbf{x}^{\mu}$, denotado por **EX1s** y su i-ésima componente por EX1s, como un vector binario que indica si en la posición i ha existido algún 1 para algún patrón miembro del conjunto fundamental, es el equivalente a hacer la operación máximo entre todos los patrones de entrada del conjunto fundamental. matemáticamente esta dado por:

$$\mathbf{EX1s} = [EX1s_1, EX1s_2, \cdots, EX1s_i, \cdots, EX1s_n]$$

Donde:

$$EX1s_i = \begin{cases} 1 & \text{si } x_i^{\mu} = 1 \text{ para algun } \mu \\ 0 & \text{de otra forma} \end{cases}$$

Ejemplo, sean $\mathbf{x}^1$, $\mathbf{x}^2$, $\mathbf{x}^3$ como en el ejemplo de la definición anterior, se tiene que:

$$\mathbf{EX1s} = [1,1,1,1]$$

***Definición 7***, Sea $\mathbf{y}^{\mu} \mid \mu \in \{1, 2, \ldots, p\}$ el conjunto fundamental de patrones de salida de dimensión m. se define el vector de existencia de 0s en $\mathbf{y}^{\mu}$ o vector mínimo de los patrones $\mathbf{y}^{\mu}$, denotado por **EY0s** y su i-ésima componente por $EY0s_i$, como un vector binario que indica si en la posición i ha existido algún 0 para algún patrón miembro del conjunto fundamental, es el equivalente a hacer la operación mínimo entre todos los patrones de salida del conjunto fundamental y sacar el complemento al vector resultante. Esto es:

$$\mathbf{EY0s} = [EY0s_1, EY0s_2, \cdots, EY0s_i, \cdots, EY0s_m]$$

Donde:

$$EY0s_i = \begin{cases} 1 & \text{si } y_i^{\mu} = 0 \text{ para algun } \mu \\ 0 & \text{de otra forma} \end{cases}$$

Ejemplo, del siguiente conjunto fundamental obtendremos **EY0s**

$$\mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{EY0s} = [1,1,1]$$

***Definición 8***, Sea $\mathbf{y}^{\mu} \mid \mu \in \{1, 2, \ldots, p\}$ el conjunto fundamental de patrones de salida de dimensión m se define el vector de existencia de unos en $\mathbf{y}^{\mu}$o vector máximo de los patrones $\mathbf{y}^{\mu}$, denotado por **EX1s** y su i-ésima componente por $EX1s_i$, como un vector binario que indica si en la posición i ha existido algún 1 para algún patrón miembro del conjunto fundamental, es el equivalente a hacer la operación máximo entre todos los patrones de salida del conjunto fundamental. Esto es:

$$\mathbf{EY1s} = [EY1s_1, EY1s_2, \cdots, EY1s_i, \cdots, EY1s_m]$$

Donde:

$$EY1s_i = \begin{cases} 1 & \text{si } y_i^{\mu} = 1 \text{ para algun } \mu \\ 0 & \text{de otra forma} \end{cases}$$

Ejemplo, sean $\mathbf{y}^1$, $\mathbf{y}^2$, $\mathbf{y}^3$ como en el ejemplo de la definición anterior, se tiene que:

$$\mathbf{EY1s} = [1,0,1]$$

Las siguientes definiciones nos permiten simplificar las memorias **M** y **W,** lo cual nos permite reducir el tiempo en la fase de recuperación.

***Definición 9***, Sea **M** una memoria asociativa Max de dimensión $m$x$n$ y $m_{ij}$ su ij-ésima posición. Se define el vector de existencia de ceros en el renglón i de la memoria asociativa Max M, denotado por **EM0s** y su i-ésima componente por $EM0s_i$, como un vector binario

que indica si en determinado renglón i de la matriz de la memoria asociativa Max **M** existe algún 0 en alguna de las posiciones del renglón.

$$\mathbf{EM0s} = [EM0s_1, EM0s_2, \cdots, EM0s_i, \cdots, EM0s_m]$$

Donde:

$$EM0s_i = \begin{cases} 1 & \text{si } m_{ij} = 0 \text{ para algun } \mu \\ 0 & \text{de otra forma} \end{cases}$$

Ejemplo, a partir de la siguiente memoria asociativa **M** obtendremos **EM0s**

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix} \quad \mathbf{EM0s} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

***Definición 10*** Sea **M** una memoria asociativa Max de dimensión *m*x*n* y $m_{ij}$ su ij-ésima posición, sea i un número entero positivo tal que $1 \le i \le m$ que indica un renglón particular, y sea $\mathbf{F}=[f_1, f_2, \ldots, f_i, \ldots, f_m]$ un vector de enteros positivos en donde su i-ésima componente, denotada por $f_i$ tal que $0 \le f_i \le n$, indica el número de unos en el renglón i. Se define el conjunto de posiciones de unos en el renglón i de la memoria asociativa Max **M**, denotado por $\mathbf{M1s}^i$ cuya c-ésima componente es $M1s_c^i$ como un conjunto de enteros positivos de dimensión $f_i$ que contiene las posiciones de los unos en el renglón i de la memoria asociativa Max,. Esto es:

$$\mathbf{M1s}^i = \{M1s_1^i, M1s_2^i, \cdots, M1s_c^i, \cdots, M1s_{f_i}^i\}, \text{donde } f_i \in \{0,1,2\cdots,n\} \text{ y } M1s_c^i = j \text{ si } m_{ij} = 1,$$

Ejemplo, a partir de la siguiente memoria asociativa **M** obtendremos $\mathbf{M1s}^i$

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix} \quad \begin{aligned} \mathbf{M1s}^1 &= \{1,3\} \\ \mathbf{M1s}^2 &= \{2,3,4\} \\ \mathbf{M1s}^3 &= \{1,3\} \end{aligned}$$

***Definición 11*** Sea **W** una memoria asociativa Min de dimension *m*x*n* y $w_{ij}$ su ij-ésima posicion, sea i un número entero positivo tal que $1 \le i \le m$ que indica un renglón particular, y sea $\mathbf{G}=[g_1, g_2, \ldots, g_i, \ldots, g_m]$ un vector de enteros positivos en donde su iésima componente denotada por $g_i$ tal que $0 \le g_i \le n$ indica el número de unos en el renglón i, entonces se define el conjunto de posiciones en de unos en el renglón i en la memoria Min, denotado como $\mathbf{W1s}^i$ y su d-ésima posición como $W1s^i_d$, como un conjunto de enteros positivos de dimensión $g_i$ que contiene las posiciones de los unos en el renglón i de la matriz de la memoria asociativa Min. Esto es:

$$\mathbf{W1s}^i = \{W1s_1^i, W1s_2^i, \cdots, W1s_d^i, \cdots, W1s_{g_i}^i\}, \textit{donde } g_i \in \{0,1,2\cdots,n\} \textit{ y } W1s_d^i = j \textit{ si } W1s_d^i = j$$

Ejemplo, a partir de la siguiente memoria asociativa Min **W** obtendremos **W1s**$^{i}$

$$\mathbf{W}=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&0&0&0\end{pmatrix}\quad\begin{matrix}\mathbf{W1s}^{1}=\{2,3,4\}\\ \mathbf{W1s}^{2}=\phi\\ \mathbf{W1s}^{3}=\phi\end{matrix}$$

***Definición 12***, Sea **W** una memoria asociativa Min de dimensión *m*x*n* y $w_{ij}$ su ij ésima posición, entonces se define el vector de existencia de dos en el renglón i de la memoria asociativa Min **W**, denotado por **EW2s** y su iésima componente por $EW2s_{i,}$ como un vector binario que indica si en determinado renglón i de la matriz de la memoria asociativa Min **W** existe algún 2 en alguna de las posiciones del renglón. Esto es:

$$\mathbf{EW2s}=[EW2s_1,EW2s_2,\cdots,EW2s_i,\cdots,EW2s_m]$$

Donde:

$$EW2s_i=\begin{cases}1 & \text{si } w_{ij}=2 \text{ para algun } \mu\\ 0 & \text{de otra forma}\end{cases}$$

$$\mathbf{W}=\begin{pmatrix}0&1&1&1\\0&0&0&0\\0&0&0&0\end{pmatrix}\quad\mathbf{EW2s}=\begin{pmatrix}0\\0\\0\end{pmatrix}$$

## *4.2 Teoremas*

Los teoremas propuestos en esta sección formalizan algunas de las ideas planteadas en la introducción de este capítulo, y estos teoremas son la parte fundamental de los nuevos algoritmos de aprendizaje y recuperación.

Sean $\mathbf{y}^{\mu}$ y $\mathbf{y}^{k}$ patrones del conjunto fundamental, tales que $k,\mu\in\{1,2,...p\}$, y $y_i^k, y_i^{\mu}$ sus respectivas i-ésimas componentes, **M** una memoria asociativa con $m_{ij}$ como su ijésima componente y **W** una memoria asociativa Min con $w_{ij}$ como su ij-ésima componente.

***Teorema 1***. Si $y_i^k=1$ para algún $k\in\{1, 2, \ldots, p\}$ entonces la memoria **M** en el renglón i sólo puede tener los valores de 1 o 2 pero nunca 0; esto es:

$$y_i^{\mu}=1\rightarrow m_{ij}=1 \text{ o } m_{ij}=2 \text{ y } m_{ij}\neq 0\ \ \forall\ j$$

Demostración:

$$m_{ij}=\bigvee_{\mu=1}^{p}\alpha\left(y_i^{\mu},x_j^{\mu}\right)$$

Como existe un $k \in \{1,2,...p\}$ tal que $y_i^k = 1$, tenemos 3 casos posibles: $k = 1$, $1 < k < p$ y $k = p$. A continuación se muestra cada caso y su demostracion:

i) $k = 1$

$$\rightarrow m_{ij} = \alpha\left(y_i^k, x_j^k\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right)$$

Pero $x_j^k \in \{0,1\}$, por lo tanto tenemos 2 casos $x_j^k = 0$ $o$ $x_j^k = 1$

1er caso $x_j^k = 0$

$$\rightarrow m_{ij} = \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(1,0)$$

$$\rightarrow m_{ij} = \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee 2$$

$$\rightarrow m_{ij} = 2$$

2do caso $x_j^k = 1$

$$\rightarrow m_{ij} = \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(1,1)$$

$$\rightarrow m_{ij} = \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee 1$$

En este caso tenemos 3 posibilidades

1 Que $\underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) = 0$ en este caso

$$\rightarrow m_{ij} = 0 \vee 1$$

$$\rightarrow m_{ij} = 1$$

2 Que $\underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) = 1$, en este caso

$$\rightarrow m_{ij} = 1 \vee 1$$

$$\rightarrow m_{ij} = 1$$

3 Que $\underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) = 2$ en este caso

$$\rightarrow m_{ij} = 2 \vee 1$$

$$\rightarrow m_{ij} = 2$$

ii) $1 < k < p$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha\left(y_i^k, x_j^k\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right)$$

Reorganizando

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee \alpha\left(y_i^k, x_j^k\right)$$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee \alpha\left(1, x_j^k\right)$$

Pero $x_j^k \in \{0,1\}$, por lo tanto tenemos 2 casos $x_j^k = 0$ *o* $x_j^k = 1$

1er caso $x_j^k = 0$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee \alpha(1,0)$$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee 2$$

$$\rightarrow m_{ij} = 2$$

2do caso $x_j^k = 1$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee \alpha(1,1)$$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee 1$$

En este caso tenemos 3 posibilidades

1 Que $\left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) = 0$ en este caso

$$\rightarrow m_{ij} = 0 \vee 1$$

$$\rightarrow m_{ij} = 1$$

2 Que $\left(\underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right)\underset{\mu=k+1}{\overset{p}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right)\right)$=1, en este caso

$$\rightarrow m_{ij} = 1 \vee 1$$

$$\rightarrow m_{ij} = 1$$

3 Que $\left(\underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right)\underset{\mu=k+1}{\overset{p}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right)\right)$=2 en este caso

$$\rightarrow m_{ij} = 2 \vee 1$$

$$\rightarrow m_{ij} = 2$$

iii) $k = p$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha\left(y_i^k, x_j^k\right)$$

Pero $x_j^k \in \{0,1\}$, por lo tanto tenemos 2 casos $x_j^k = 0$ *o* $x_j^k = 1$

1er caso $x_j^k = 0$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(1,0)$$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right) \vee 2$$

$$\rightarrow m_{ij} = 2$$

2do caso $x_j^k = 1$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(1,1)$$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right) \vee 1$$

En este caso tenemos 3 posibilidades

1 Que $\underset{\mu=1}{\overset{k-1}{\vee}}\alpha\left(y_i^\mu, x_j^\mu\right)$ =0 en este caso

$$\rightarrow m_{ij} = 0 \vee 1$$

$$\rightarrow m_{ij} = 1$$

2 Que $\bigvee_{\mu=1}^{k-1} \alpha\left(y_i^\mu, x_j^\mu\right)$=1, en este caso

$$\rightarrow m_{ij} = 1 \vee 1$$
$$\rightarrow m_{ij} = 1$$

3 Que $\bigvee_{\mu=1}^{k-1} \alpha\left(y_i^\mu, x_j^\mu\right)$=2 en este caso

$$\rightarrow m_{ij} = 2 \vee 1$$
$$\rightarrow m_{ij} = 2$$

En cualquier caso el valor máximo es 1 o 2 pero nunca 0.

***Teorema 2***. Si $x_j^k$=0 para algun k $\in$ {1, 2, …,p} entonces la memoria **M** en la columna j solo puede tener los valores de 1 o 2 pero nunca 0, esto es:

$$x_j^k = 1 \rightarrow m_{ij} = 1 \text{ o } m_{ij} = 2 \text{ y } m_{ij} \neq 0 \ \ \forall \text{ i}$$

Demostración:

$$m_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right)$$

Como existe un $k \in \{1,2,...p\}$ tal que $y_i^k = 1$, tenemos 3 casos posibles: $k = 1$, $1 < k < p$ y $k = p$. A continuacion se muestra cada caso y su demostracion:

i) $k = 1$

$$\rightarrow m_{ij} = \alpha\left(y_i^k, x_j^k\right) \bigvee_{\mu=k+1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right)$$

Pero $y_i^k \in \{0,1\}$, por lo tanto tenemos 2 casos $y_i^k = 0$ *o* $y_i^k = 1$

1er caso $y_i^k$=1

$$\rightarrow m_{ij} = \bigvee_{\mu=k+1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(1,0)$$
$$\rightarrow m_{ij} = \bigvee_{\mu=k+1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right) \vee 2$$
$$\rightarrow m_{ij} = 2$$

2do caso $y_i^k$=0

$$\rightarrow m_{ij} = \bigvee_{\mu=k+1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(0,0)$$

$$\rightarrow m_{ij} = \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \vee 1$$

En este caso tenemos 3 posibilidades

1 Que $\underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) = 0$ en este caso

$$\rightarrow m_{ij} = 0 \vee 1$$
$$\rightarrow m_{ij} = 1$$

2 Que $\underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) = 1$, en este caso

$$\rightarrow m_{ij} = 1 \vee 1$$
$$\rightarrow m_{ij} = 1$$

3 Que $\underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) = 2$ en este caso

$$\rightarrow m_{ij} = 2 \vee 1$$
$$\rightarrow m_{ij} = 2$$

ii) $1 < k < p$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \vee \alpha\left(y_i^{k}, x_j^{k}\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right)$$

Reorganizando

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \right) \vee \alpha\left(y_i^{k}, x_j^{k}\right)$$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \right) \vee \alpha\left(y_i^{k}, 0\right)$$

Pero $y_i^k \in \{0,1\}$, por lo tanto tenemos 2 casos $y_i^k = 0$ *o* $y_i^k = 1$

1er caso $y_i^k = 1$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \right) \vee \alpha(1,0)$$
$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^{\mu}, x_j^{\mu}\right) \right) \vee 2$$
$$\rightarrow m_{ij} = 2$$

$2^{do}$ caso $y_i^k = 0$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee \alpha(0,0)$$

$$\rightarrow m_{ij} = \left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) \vee 1$$

En este caso tenemos 3 posibilidades

1 Que $\left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) = 0$ en este caso

$$\rightarrow m_{ij} = 0 \vee 1$$

$$\rightarrow m_{ij} = 1$$

2 Que $\left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) = 1$, en este caso

$$\rightarrow m_{ij} = 1 \vee 1$$

$$\rightarrow m_{ij} = 1$$

3 Que $\left( \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \underset{\mu=k+1}{\overset{p}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \right) = 2$ en este caso

$$\rightarrow m_{ij} = 2 \vee 1$$

$$\rightarrow m_{ij} = 2$$

iii) $k = p$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha\left(y_i^k, x_j^k\right)$$

Reorganizando

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha\left(y_i^k, x_j^k\right)$$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha\left(y_i^k, 0\right)$$

Pero $y_i^k \in \{0,1\}$, por lo tanto tenemos 2 casos $y_i^k = 0$ *o* $y_i^k = 1$

$1^{er}$ caso $y_i^k = 1$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(1,0)$$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee 2$$

$$\rightarrow m_{ij} = 2$$

$2^{do}$ caso $y_i^k = 0$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee \alpha(0,0)$$

$$\rightarrow m_{ij} = \underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) \vee 1$$

En este caso tenemos 3 posibilidades

1 Que $\underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) = 0$ en este caso

$$\rightarrow m_{ij} = 0 \vee 1$$

$$\rightarrow m_{ij} = 1$$

2 Que $\underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) = 1$, en este caso

$$\rightarrow m_{ij} = 1 \vee 1$$

$$\rightarrow m_{ij} = 1$$

3 Que $\underset{\mu=1}{\overset{k-1}{\vee}} \alpha\left(y_i^\mu, x_j^\mu\right) = 2$ en este caso

$$\rightarrow m_{ij} = 2 \vee 1$$

$$\rightarrow m_{ij} = 2$$

En cualquier caso el valor máximo es 1 o 2 pero nunca 0.

***Teorema 3***. La única forma en que una memoria asociativa **M** en su posición $m_{ij}$ pueda tener el valor de 0 es cuando $y_i^\mu \neq 1$ y $x_j^\mu \neq 0$ para toda $\mu \in \{1, 2, \ldots, p\}$

$$m_{ij} = 0 \leftrightarrow y_i^\mu \neq 1 \text{ y } x_j^\mu \neq 0 \ \forall \ \mu \in \{1,2,...,p\}$$

Demostración

$$m_{ij} = \vee \ \alpha\left(y_i^\mu, x_j^\mu\right)$$

A) $m_{ij} = 0 \rightarrow y_i^\mu \neq 1$ y $x_j^\mu \neq 0 \ \forall \ \mu \in \{1,2,...,p\}$

$$m_{ij} = 0$$

$$\rightarrow \alpha\left(y_i^\mu, x_j^\mu\right) = 0 \ \forall \ \mu \in \{1,2,...,p\}$$

$$\rightarrow y_i^\mu = 0 \text{ y } x_j^\mu = 1 \ \forall \ \mu \in \{1,2,...,p\}$$

$$\rightarrow y_i^\mu \neq 1 \text{ y } x_j^\mu \neq 0 \ \forall \ \mu \in \{1,2,...,p\}$$

B) $y_i^\mu \neq 1$ y $x_j^\mu \neq 0 \; \forall \; \mu \in \{1,2,...,p\} \rightarrow m_{ij} = 0$

$$y_i^\mu \neq 1 \;\; \forall \;\; \mu \rightarrow y_i^\mu = 0 \;\; \forall \;\; \mu \text{ tal que } \mu \in \{1,2,...,p\}$$
$$x_i^\mu \neq 0 \;\; \forall \;\; \mu \rightarrow y_i^\mu = 1 \;\; \forall \;\; \mu \text{ tal que } \mu \in \{1,2,...,p\}$$
$$\rightarrow m_{ij} = \bigvee_{\mu=1}^{p} \alpha(0,1)$$
$$\rightarrow m_{ij} = 0$$

***Teorema 4***. Si $y_j^k$=0 para algún k $\in$ {1, 2, …,p} entonces la memoria **W** en el renglón i sólo puede tener los valores de 0 o 1 pero nunca 2, esto es:

$$y_i^\mu = 0 \rightarrow w_{ij} = 0 \text{ o } w_{ij} = 1 \text{ y } w_{ij} \neq 2 \;\; \forall \; j$$

Demostración: Este teorema es el dual del teorema 1

***Teorema 5***. Si $y_j^k$=0 para algún k $\in$ {1, 2, …,p} entonces la memoria **W** en la columna j sólo puede tener los valores de 0 o 1 pero nunca 2, esto es:

$$x_j^k = 1 \rightarrow w_{ij} = 0 \text{ o } w_{ij} = 1 \text{ y } w_{ij} \neq 2 \;\; \forall \; i$$

Demostración: Este teorema es el dual del teorema 2

***Teorema 6***. La única forma en que una posición $w_{ij}$ en la memoria asociativa **W** pueda valer 2 es cuando $y_i^\mu \neq 0$ y $x_j^\mu \neq 1$ para toda $\mu \in \{1, 2, \ldots, p\}$

$$w_{ij} = 2 \leftrightarrow y_i^\mu \neq 0 \text{ y } x_j^\mu \neq 1 \;\; \forall \;\; \mu \in \{1,2,...,p\}$$

Demostración. Este es el dual del teorema 3

***Teorema 7.*** Las posiciones que tienen el valor de 2 en la memoria asociativa Max **M**, están dadas por el producto cruz entre $\mathbf{Y1s}^\mu$ y $\mathbf{X0s}^\mu$, esto es

$$m_{ij} = 2 \leftrightarrow i \in Y1s^\mu \;\; y \;\; j \in X0s^\mu$$

$$\mathbf{Y1s}^\mu \times \mathbf{X0s}^\mu = \{(Y1s_1^\mu, X0s_1^\mu),(Y1s_1^\mu, X0s_2^\mu),...,(Y1s_1^\mu, X0s_w^\mu),...,(Y1s_2^\mu, X0s_1^\mu),...,(Y1s_r^\mu, X0s_w^\mu)\}$$

Demostración

Por definición $\mathbf{Y1s}^\mu$ es el vector que contiene las posiciones de cada 1 en el patrón de salida $\mathbf{y}^\mu$, de la misma forma $\mathbf{X0s}^\mu$ es el vector que contiene las posiciones de de cada 0 del patrón de entrada $\mathbf{X}^\mu$. Por lo tanto $\mathbf{Y1s}^\mu \times \mathbf{X0s}^\mu$ es el conjunto de coordinadas en las cuales $\alpha(X_i^\mu, X_i^\mu) = \alpha(1,0) = 2$

***Teorema 8.*** Las posiciones que tienen el valor de 0 en la memoria asociativa Min **W**, estan dadas por el producto cruz entre $\mathbf{Y0s}^{\mu}$ y $\mathbf{X1s}^{\mu}$, esto es

$$w_{ij} = 0 \leftrightarrow i \in \mathbf{Y0s}^{\mu} \;\; y \;\; j \in \mathbf{X1s}^{\mu}$$

$$\mathbf{Y0s}^{\mu} \times \mathbf{X1s}^{\mu} = \left\{(Y0s_1^{\mu}, X1s_1^{\mu}),(Y0s_1^{\mu}, X1s_2^{\mu}),...,(Y0s_1^{\mu}, X1s_w^{\mu}),...,(Y0s_2^{\mu}, X1s_1^{\mu}),...,(Y0s_r^{\mu}, X1s_w^{\mu})\right\}$$

Demostración

Por definición $\mathbf{Y0s}^{\mu}$ es el vector que contiene las posiciones de cada 0 en el patron de salida $\mathbf{y}^{\mu}$, de la misma forma $\mathbf{X1s}^{\mu}$ es el vector que contiene las posiciones de de cada 1 del patrón de entrada $\mathbf{x}^{\mu}$. Por lo tanto $\mathbf{Y0s}^{\mu} \times \mathbf{X1s}^{\mu}$ es el conjunto de coordinadas en las cuales $\alpha\left(Y_j^{\mu}, X_i^{\mu}\right) = \alpha(0,1) = 0$

Los siguientes teoremas son usados para simplificar la fase de recuperación

***Teorema 9*** Si $m_{ik}$=0 para algún k$\in$ {1, 2, …, n} entonces el valor de salida del vector **y** en la posición $y_i$ es 0. Esto es

$$\exists m_{ik} = 0 \rightarrow y_i = 0 \;\; donde \;\; k \in \{1,2,..,n\}$$

Demostración

$$y_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j\right) y \;\exists \text{ k tal que } m_{ik} = 0$$

$$y_i = \bigwedge_{j=1}^{n-1} \beta\left(m_{ij}, x_j\right) \wedge \beta\left(m_{ik}, x_k\right)$$

$$\rightarrow y_i = \bigwedge_{j=1}^{n-1} \beta\left(m_{ij}, x_j\right) \wedge \beta\left(0, x_k\right)$$

Pero $x_k \in \{0,1\}$, por tanto $\beta(0, x_k)$ tiene 2 posibilidades $\beta(0,0) = 0$ y $\beta(0,1) = 0$, en cualquier caso el valor de salida es 0 . Por lo tanto tenemos:

$$\rightarrow y_i = \bigwedge_{j=1}^{n-1} \beta\left(m_{ij}, x_j\right) \wedge \beta(0,0)$$

$$\rightarrow y_i = \bigwedge_{j=1}^{n-1} \beta\left(m_{ij}, x_j\right) \wedge 0$$

$$\rightarrow y_i = 0$$

***Teorema 10*** Si $m_{ik}$ =2 para todo k$\in$ {1, 2, …, n} entonces el valor de salida del vector **y** en la posición $y_i$ es 1. Esto es

$$m_{ik} = 2 \;\; \forall k \rightarrow Y_i = 1 \;\; donde \;\; k \in \{1,2,..,n\}$$

Demostración

$$y_i = \bigwedge_{j=1}^{n} \beta(m_{ij}, x_j), m_{ij} = 2 \ \forall_j \in \{1,2,...,n\}$$

$$\rightarrow y_i = \bigwedge_{j=1}^{n} \beta(2, x_j)$$

$x_j \in \{0,1\}$, por tanto $\beta(2, x_j)$ tiene 2 posibilidades $\beta(2,0) = 1$ o $\beta(2,1) = 1$, en cualquier caso el valor es 1. Por lo tanto tenemos:

$$\rightarrow y_i = \bigwedge_{j=1}^{n} 1$$

$$\rightarrow y_i = 1$$

De esta forma este teorema queda demostrado

***Teorema 11*** Si $m_{ij} \neq 0$ para todo j$\in$ {1, 2, …, n} y existe un k$\in$ {1, 2, …, n} tal que $m_{ik}$= y $xk$=1entonces $y_i$=1, de otra forma $y_i$=0. Esto es:

$$m_{ij} \neq 0 \ \forall \ j, \exists \ k \mid m_{ik} = 1, x_k = 0 \rightarrow y_i = 0 \ \ donde \ \ j,k \in \{1,2,..,n\}$$

Demostración

$$y_i = \bigwedge_{j=1}^{n} \beta(m_{ij}, x_j)$$

$$m_{ij} \neq 0 \ \forall \ j \in \{1,2,..,n\} \rightarrow m_{ij} = 1 \ o \ m_{ij} = 2$$

Extraemos el caso k

$$y_i = \bigwedge_{j=1}^{n-1} \beta(m_{ij}, x_j) \wedge \beta(m_{ik}, x_k), donde \ \ j,k \in \{1,2,...,n\}$$

Como $m_{ij} \neq 0$, entonces $m_{ij} = 1$ o $m_{ij} = 2$,

A) $m_{ik} = 1$

$$\rightarrow y_i = \bigwedge_{j=1}^{n-1} \beta(m_{ij}, x_j) \wedge \beta(1, x_k)$$

Entonces, dado que $x_k \in \{0,1\}$, tenemos 2 casos:

1er caso $x_k = 0$

$$\rightarrow y_i = \bigwedge_{j=1}^{n-1} \beta(m_{ij}, x_j) \wedge \beta(1,0)$$

$$\rightarrow y_i = \bigwedge_{j=1}^{n-1} \beta(m_{ij}, x_j) \wedge 0$$

$$\rightarrow y_i = 0$$

2do caso $x_k = 1$

$$\rightarrow y_i = \underset{j=1}{\overset{n-1}{\wedge}} \beta\left(M_{ij}, x_j\right) \wedge \beta(1,1) \rightarrow y_i = \underset{j=1}{\overset{n-1}{\wedge}} \beta\left(M_{ij}, x_j\right) \wedge 1$$

B) $m_{ik} = 2$

$$\rightarrow y_i = \underset{j=1}{\overset{n-1}{\wedge}} \beta\left(m_{ij}, x_j\right) \wedge \beta\left(2, x_k\right)$$

Por definición $\beta\left(2, x_k\right) = 1$ cualquiera que sea el valor de $x_k$, por tanto es imposible que si $m_{ik} = 2$ este nos dé un valor de 0. Por tanto, la única forma de que $y_i$ valga 0, dado que $m_{ik} \neq 0$ es cuando $\exists m_{ij} = 1 \; y \; x_j = 0$ pero este caso en es el demostrado en el caso A de esta demostración.

***Teorema 12*** Si $w_{ik} \neq 2$ para algún $k \in \{1, 2, \ldots, n\}$ entonces el valor de salida y en la posición i es 1. Esto es

$$\exists w_{ik} = 2 \rightarrow y_i = 1 \; donde \; k \in \{1,2,..,n\}$$

Demostración

Este es el dual del teorema 9

***Teorema 13*** Si $w_{ik} = 0$ para todo $k \in \{1, 2, \ldots, n\}$ entonces el valor de salida y en la posición i es 0. Esto es

$$w_{ik} = 0 \; \forall k \rightarrow y_i = 0 \; donde \; k \in \{1,2,..,n\}$$

Demostración

Este es el dual del teorema 10

***Teorema 14*** Si $w_{ij} \neq 2$ para todo $j \in \{1, 2, \ldots, n\}$ y existe un $k \in \{1, 2, \ldots, n\}$ tal que $w_{ik} = 1$ y $x_k = 1$entonces $y_i = 1$, de otra forma $y_i = 0$. Esto es:

$$w_{ij} \neq 2 \; \forall \; j, \exists \; k \mid m_{ik} = 1, x_k = 1 \rightarrow y_i = 1 \; donde \; j,k \in \{1,2,..,n\}$$

Demostración

Este es el dual del teorema 11

## *4.3 Memorias Asociativas Alfa Beta Simplificadas*

En esta sección se proponen los nuevos algoritmos de aprendizaje y recuperación de las memorias asociativas Alfa-Beta simplificadas. Estos nuevos algoritmos están basados en las definiciones y teoremas desarrollados en las secciones 4.1 y 4.2.

### 4.3.1 Memorias Heteroasociativas Simplificadas

Se tienen dos tipos de memorias heteroasociativas Alfa-Beta: tipo Max, denotadas por **M** y $m_{ij}$ como su ij-ésima componente y las tipo Min, denotadas por **W** y $w_{ij}$ como su ij-ésima componente. Como cada una de estas memorias puede dar una patrón de salida **y** diferente, para diferenciarlos definiremos a $\mathbf{YM} = [YM_1, YM_2, \cdots, YM_j, \cdots, YM_m]$, como el vector de dimensión *m* que resulta de operar el patrón **x** de entrada con la memoria asociativa $\alpha\beta$ **M**. Y a $\mathbf{YW} = [YW_1, YW_2, \cdots, YW_j, \cdots, YW_m]$ como el vector de dimensión *m*, que resulta de operar el patrón **x** de entrada con la memoria asociativa $\alpha\beta$ **W.**

#### 4.3.1.1 Memoria Heteroasociativa Max

##### Fase de Aprendizaje

**Paso 1.** Para cada $\mu$ = 1, 2, ..., *p*, a partir de la pareja ($\mathbf{x}^{\mu}$,$\mathbf{y}^{\mu}$) Obtenemos los vectores $\mathbf{X0s}^{\mu}$, $\mathbf{Y1s}^{\mu}$ para cada par en el conjunto fundamental de patrones {($\mathbf{x}^{\mu}$, $\mathbf{y}^{\mu}$)| μ=1, 2,..., p). Además también hacemos:

$EX0s_k$ =1 donde k ∈ $\mathbf{X0s}^{\mu}$
$EX1s_k$ =1 donde k ∈ $\mathbf{X1s}^{\mu}$
$EY0s_k$ =1 donde k ∈ $\mathbf{Y0s}^{\mu}$
$EY1s_k$ =1 donde k ∈ $\mathbf{Y1s}^{\mu}$

**Paso 2.** Para cada μ ∈ {1,2,3..p} hacemos:

$m_{ij} = 2$ ∀i,j ∈ { $\mathbf{Y1s}^{\mu}$ × $\mathbf{X0s}^{\mu}$ } por teorema 7

**Paso 3** Como paso final, después de haber realizado el paso 2 p veces, revisamos las memorias Max y Min para poner los valores que faltan.

$m_{ij} = 2 \leftrightarrow m_{ij} = 2$

$m_{ij} = 1 \leftrightarrow m_{ij} \neq 2$ y ($EY1s_i$ =1 o $EX0s_i$ =1) por teoremas 1-2

$m_{ij} = 0 \leftrightarrow m_{ij} \neq 2$ y ($EY1s_i \neq 1$ y $EX0i_i \neq 1$) por teorema 3

##### Fase de Recuperación

**Paso 1** De la memoria **M**, obtenemos los vectores **EM0s, M1s**$^{i}$, como se definió en 4.1
**Paso 2** Obtenemos el vector **X0s** como se definió en 4.1.

**Paso 3** El valor del patrón de salida **y** en la posición i, cuando el patrón de entrada **x** es operado sobre una memoria asociativa Max **M,** es:

$YM_i$=0 si $EM0s_i = 1$ por teorema 9
$YM_i$=1 si $EM0s_i = 0$ y $\mathbf{M1s}^i = \emptyset$ por teorema 10
$YM_i$=0 si $EM0s_i = 0$ y $\mathbf{M1s}^i \cap \mathbf{X0s} \neq \emptyset$ por teorema 11
$YM_i$=1 si $EM0s_i = 0$ y $\mathbf{M1s}^i \cap \mathbf{X0s} = \emptyset$ por teorema 11

### 4.3.1.2 Memoria Heteroasociativa Min

#### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ Obtener los vectores $\mathbf{X1s}^\mu$, $\mathbf{Y0s}^\mu$ para cada par en el conjunto fundamental de patrones $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu=1, 2,..., p\}$. Además también hacemos:

$EX0s_k$ =1 donde k $\in \mathbf{X0s}^\mu$
$EX1s_k$ =1 donde k $\in \mathbf{X1s}^\mu$
$EY0s_k$ =1 donde k $\in \mathbf{Y0s}^\mu$
$EY1s_k$ =1 donde k $\in \mathbf{Y1s}^\mu$

**Paso 2.** Para cada $\mu \in \{1,2,3..p\}$ hacemos:

$w_{ij} = 0 \quad \forall i,j \in \{ \mathbf{Y0s}^\mu \times \mathbf{X1s}^\mu \}$ por el teorema 8

**Paso 3** Como paso final, después de haber realizado el paso 2 *p* veces, revisamos la memoria Min para poner los valores que faltan.

$w_{ij} = 0 \leftrightarrow w_{ij} = 0$
$w_{ij} = 1 \leftrightarrow w_{ij} \neq 0$ y ($EY0s_i$ =1 o $EX1s_j$ =1) por los teoremas 4-5
$w_{ij} = 2 \leftrightarrow w_{ij} \neq 0$ y ($EY0s_i \neq 1$ y $EX1s_j \neq 1$ ) por teorema 6

#### Fase de Recuperación

**Paso 1** De la memoria Min **W**, obtenemos $\mathbf{W1s}^i$, **EW2s** como se definió en 4.1
**Paso 2** Obtenemos el vector **X1s** como se definió en 4.1.
**Paso 3** El valor del patrón de salida **y** en la posición i, cuando el patrón de entrada **x** es operado sobre una memoria asociativa **W**, es:

$YW_i$=1 si $EW2s_i = 1$ por teorema 12
$YW_i$=0 si $EW2s_i = 0$ y $\mathbf{W1s}^i = \emptyset$ por teorema 13
$YW_i$=1 si $EW2s_i = 0$ y $\mathbf{W1s}^i \cap \mathbf{X1s} \neq \emptyset$ por teorema 14
$YW_i$=0 si $EW2s_i = 0$ y $\mathbf{W1s}^i \cap \mathbf{X1s} = \emptyset$ por teorema 14

### 4.3.1.3 Algoritmo Combinado de las Memorias Heteroasociativas Simplificadas

#### Fase de Aprendizaje

Dado que el paso uno es el mismo en ambos tipos de memoria, es posible realizar ese paso una sola vez y utilizar los datos para ambos tipos de memoria. Entonces el algoritmo queda:

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{y}^{\mu})$ Obtener los vectores $\mathbf{X1s}^{\mu}$, $\mathbf{Y0s}^{\mu}$ para cada par en el conjunto fundamental de patrones $\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})| \mu=1, 2,..., p\}$. Además también hacemos:

$EX0s_k = 1$ donde $k \in \mathbf{X0s}^{\mu}$
$EX1s_k = 1$ donde $k \in \mathbf{X1s}^{\mu}$
$EY0s_k = 1$ donde $k \in \mathbf{Y0s}^{\mu}$
$EY1s_k = 1$ donde $k \in \mathbf{Y1s}^{\mu}$

**Paso 2.** Para cada $\mu \in \{1,2,3..p\}$ hacemos:
$m_{ij} = 2 \ \forall i,j \in \{ \mathbf{Y1s}^{\mu} \times \mathbf{X0s}^{\mu} \}$ por teorema 7
$w_{ij} = 0 \ \forall i,j \in \{ \mathbf{Y0s}^{\mu} \times \mathbf{X1s}^{\mu} \}$ por el teorema 8

**Paso 3** Como paso final, después de haber realizado el paso 2 *p* veces, revisamos las memorias Max y Min para poner los valores que faltan.

$m_{ij} = 2 \leftrightarrow m_{ij} = 2$
$m_{ij} = 1 \leftrightarrow m_{ij} \neq 2$ y $(EY1s_i = 1$ o $EX0s_j = 1)$ por teoremas 1-2
$m_{ij} = 0 \leftrightarrow m_{ij} \neq 2$ y $(EY1s_i \neq 1$ y $EX0s_j \neq 1)$ por teorema 3
$w_{ij} = 0 \leftrightarrow w_{ij} = 0$
$w_{ij} = 1 \leftrightarrow w_{ij} \neq 0$ y $(EY0s_i = 1$ o $EX1s_j = 1)$ por los teoremas 4-5
$w_{ij} = 2 \leftrightarrow w_{ij} \neq 0$ **y** $(EY0s_i \neq 1$ y $EX1s_j \neq 1)$ por teorema 6

## Ejemplo de memorias heteroasociativas Alfa-Beta Simplificadas

En esta sección se desarrollará un ejemplo usando las memorias asociativas simplificadas, y para este fin usaremos el mismo conjunto fundamental que se dio con los algoritmos originales en el apéndice D. Para este ejemplo utilizaremos patrones de entrada de dimensión 5 y patrones de salida de dimensión 6. Utilizaremos el siguiente conjunto fundamental:

$$\mathbf{x}^1 = \begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix} \quad \mathbf{y}^1 = \begin{pmatrix}1\\0\\1\\0\\1\\0\end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix}0\\1\\1\\0\\0\end{pmatrix} \quad \mathbf{y}^2 = \begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix}0\\0\\1\\0\\1\end{pmatrix} \quad \mathbf{y}^3 = \begin{pmatrix}1\\0\\1\\0\\0\\1\end{pmatrix}$$

**Fase de Aprendizaje:**

**Paso 1.** Obtenemos los conjuntos $\mathbf{X0s}^\mu, \mathbf{X1s}^\mu, \mathbf{Y0s}^\mu, \mathbf{Y1s}^\mu \forall \mu$ y los vectores $\mathbf{EX0s}, \mathbf{EX1s}, \mathbf{EY0s}, \mathbf{EY1s}$

$\mathbf{X0s}^1 = \phi,\ \mathbf{X1s}^1 = \{1,2,3,4,5\},\ \mathbf{Y0s}^1 = \{2,4,6\},\ \mathbf{Y1s}^1 = \{1,3,5\}$
$\mathbf{X0s}^2 = \{1,4,5\},\ \mathbf{X1s}^2 = \{2,3\},\ \mathbf{Y0s}^2 = \{1,2\},\ \mathbf{Y1s}^2 = \{3,4,5,6\}$
$\mathbf{X0s}^3 = \{1,2,4\},\ \mathbf{X1s}^3 = \{3,5\},\ \mathbf{Y0s}^3 = \{2,4,5\},\ \mathbf{Y1s}^3 = \{1,3,6\}$
$\mathbf{EX0s} = [1,1,0,1,1]$
$\mathbf{EX1s} = [1,1,1,1,1]$
$\mathbf{EY0s} = [1,1,0,1,1,1]$
$\mathbf{EY1s} = [1,0,1,1,1,1]$

**Paso 2.** Obtenemos $\mathbf{Y1s}^\mu \times \mathbf{X0s}^\mu$ para cada $\mu \in \{1,2,3..p\}$

$\mathbf{Y1s}^1 \times \mathbf{X0s}^1 = \{1,3,5\} \times \phi = \phi$
$\mathbf{Y1s}^2 \times \mathbf{X0s}^2 = \{3,4,5,6\} \times \{1,4,5\} = \{(3,1),(3,4),(3,5),(4,1),(4,4),(4,5),(5,1),(5,4),(5,5),(6,1),(6,4),(6,5)\}$
$\mathbf{Y1s}^3 \times \mathbf{X0s}^3 = \{1,3,6\} \times \{1,2,4\} = \{(1,1),(1,2),(1,4),(3,1),(3,2),(3,4),(6,1),(6,2),(6,4)\}$

Y ahora a la memoria Max **M**, le introducimos los valores en los que $m_{ij}=2$

$$\mathbf{M} = \begin{pmatrix} 2 & 2 & & 2 & \\ & & & & \\ 2 & 2 & & 2 & 2 \\ 2 & & & 2 & 2 \\ 2 & & & 2 & 2 \\ 2 & 2 & & 2 & 2 \end{pmatrix}$$

Obtenemos $\mathbf{Y0s}^\mu \times \mathbf{X1s}^\mu$ para cada $\mu \in \{1,2,3..p\}$
$\mathbf{Y0s}^1 \times \mathbf{X1s}^1 = \{2,4,6\} \times \{1,2,3,4,5\} = \{(2,1),(2,2),(2,3),(2,4),(2,5),(4,1),(4,2),(4,3),(4,4),(4,5),(6,1),(6,2),(6,3),(6,4),(6,5)\}$
$\mathbf{Y0s}^2 \times \mathbf{X1s}^2 = \{1,2\} \times \{2,3\} = \{(1,2),(1,3),(2,2),(2,3)\}$
$\mathbf{Y0s}^3 \times \mathbf{X1s}^3 = \{2,4,5\} \times [3,5] = \{(2,3),(2,5),(4,3),(4,5),(5,3),(5,5)\}$

Y ahora a la memoria Min **W**, le introducimos los valores en los que $w_{ij}$=0

$$\mathbf{W} = \begin{pmatrix} & 0 & 0 & & \\ 0 & 0 & 0 & 0 & 0 \\ & & & & \\ 0 & 0 & 0 & 0 & 0 \\ & & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

**Paso 3.** Ponemos los valores faltantes en las memoria, para facilitar la comprensión pondremos los vectores **EX0s**, **EX1s**, **EY0s**, **EY1s** junto con la memoria, dado que **EY0s**, **EY1s** afecta los renglones, pondremos la transpuesta de estos. Empezaremos con la memoria **M**

$$\mathbf{EX0s} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$(\mathbf{EY1s})^t = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 2 & 2 & & 2 & \\ & & & & \\ 2 & 2 & & 2 & 2 \\ 2 & & & 2 & 2 \\ 2 & & & 2 & 2 \\ 2 & 2 & & 2 & 2 \end{pmatrix}$$

El algoritmo nos dice que las posiciones $m_{ij}$=2 permanecen igual, y para las que no tienen valor si $\mathbf{EY1s}_i$=1 o $\mathbf{EX0s}_j$=1 entonces $m_{ij}$=1, realizando esta parte

$$\mathbf{EX0s} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$(\mathbf{EY1s})^t = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 \end{pmatrix}$$

Finalmente si $\mathbf{EY1s}_i$=0 y $\mathbf{EX0s}_j$=0 entonces $m_{ij}$=0, en este ejemplo tenemos únicamente la posición (2,3). Entonces la memoria M queda como se muestra a continuación

$$\mathbf{M} = \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 \end{pmatrix}$$

Ahora se ahora le ponemos los valores faltantes a la memoria **W**

$$\mathbf{EX1s} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$(\mathbf{EY0s})^t = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{W} = \begin{pmatrix} & 0 & 0 & & \\ 0 & 0 & 0 & 0 & 0 \\ & & & & \\ 0 & 0 & 0 & 0 & 0 \\ & & 0 & & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

El algoritmo nos dice que las posiciones $w_{ij}$=2 permanecen igual, y para las que no tienen valor si $\mathbf{EY0s}_i$=1 o $\mathbf{EX1s}_j$=1 entonces $w_{ij}$=1, realizando esta parte

$$\mathbf{EX1s} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$(\mathbf{EY0s})^t = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Como no existe una posición en la que $\mathbf{EY0s}_i$=0 y $\mathbf{EX1s}_j$=0, entonces no existen posiciones $m_{ij}$=2, por lo tanto la memoria queda:

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

### Fase de Recuperación Memoria Max

**Paso 1** De la memoria **M**, obtenemos los vectores **EM0s, M1s**$^i$, como se definió en 4.1, para ilustrarlo pondremos a la transpuesta de **EM0s** al lado de la memoria Max **M.**

$$\mathbf{M} = \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 \end{pmatrix} \rightarrow \ (\mathbf{EM0s})^t = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{aligned} \mathbf{M1s}^1 &= \{3,5\} \\ \mathbf{M1s}^2 &= \{1,2,4,5\} \\ \mathbf{M1s}^3 &= \{3\} \\ \mathbf{M1s}^4 &= \{2,3\} \\ \mathbf{M1s}^5 &= \{2,3\} \\ \mathbf{M1s}^6 &= \{3\} \end{aligned}$$

**Paso 2** Obtenemos los conjuntos **X0s** de los patrones de entrada como se definió en 4.1.

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \ \mathbf{X0s}^1 = \phi \qquad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \ \mathbf{X0s}^2 = \{1,4,5\} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \mathbf{X0s}^3 = \{1,2,4\}$$

**Paso 3** El valor del patron de salida **YM** en la posición $YM_i$, cuando el patron de entrada **x** es operado sobre una memoria asociativa Max **M,** es:

$YM_i$=0 si $EM0s_i$ = 1
$YM_i$=1 si $EM0s_i$ = 0 y $\mathbf{M1s}^i = \emptyset$
$YM_i$=0 si $EM0s_i$ = 0 y $\mathbf{M1s}^i \cap \mathbf{X0s} \neq \emptyset$
$YM_i$=1 si $EM0s_i$ = 0 y $\mathbf{M1s}^i \cap \mathbf{X0s} = \emptyset$

Empezaremos tratando de recuperar el patrón asociado a $\mathbf{x}^1$:

Para i=1 tenemos que $EM0s^1 = 0$ y $\mathbf{M1s}^1 \cap \mathbf{X0s}^1 = \{3,5\} \cap \phi = \phi$, entonces $YM_1$=1
Para i=2 tenemos que $EM0s^2 = 1$, por tanto $YM_2$=0
Para i=3 tenemos que $EM0s^3 = 0$ y $\mathbf{M1s}^3 \cap \mathbf{X0s}^1 = \{3\} \cap \phi = \phi$, entonces $YM_3$=1
Para i=4 tenemos que $EM0s^4 = 0$ y $\mathbf{M1s}^4 \cap \mathbf{X0s}^1 = \{2,3\} \cap \phi = \phi$ entonces $YM_4$=1
Para i=5 tenemos que $EM0s^5 = 0$ y $\mathbf{M1s}^5 \cap \mathbf{X0s}^1 = \{2,3\} \cap \phi = \phi$ entonces $YM_5$=1
Para i=6 tenemos que $EM0s^6 = 0$ y $\mathbf{M1s}^6 \cap \mathbf{X0s}^1 = \{3\} \cap \phi = \phi$ entonces $YM_5$=1

Entonces el vector final de salida es:

$$\mathbf{YM}(\mathbf{x}^1) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Comparando el vector de salida **YM** con el vector $\mathbf{y}^1$ vemos que difiere en las posiciones 4 y 6.

Ahora tratamos de recuperar el patrón asociado a $\mathbf{x}^2$:

Para i=1 tenemos que $EM0s^1 = 0$ y $\mathbf{M1s}^1 \cap \mathbf{X0s}^2 = \{3,5\} \cap \{1,4,5\} = \{5\}$, entonces $YM_1$=0
Para i=2 tenemos que $EM0s^2 = 1$, por tanto $YM_2$=0
Para i=3 tenemos que $EM0s^3 = 0$ y $\mathbf{M1s}^3 \cap \mathbf{X0s}^2 = \{3\} \cap \{1,4,5\} = \phi$, entonces $YM_3$=1
Para i=4 tenemos que $EM0s^4 = 0$ y $\mathbf{M1s}^4 \cap \mathbf{X0s}^2 = \{2,3\} \cap \{1,4,5\} = \phi$ entonces $YM_4$=1
Para i=5 tenemos que $EM0s^5 = 0$ y $\mathbf{M1s}^5 \cap \mathbf{X0s}^2 = \{2,3\} \cap \{1,4,5\} = \phi$ entonces $YM_5$=1
Para i=6 tenemos que $EM0s^6 = 0$ y $\mathbf{M1s}^6 \cap \mathbf{X0s}^2 = [3] \cap [1,4,5] = \phi$ entonces $YM_5$=1

Entonces el vector final de salida **YM** es:

$$\mathbf{YM}(\mathbf{x}^2) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Comparando el el vector de salida **YM** con el vector $\mathbf{y}^2$ vemos que es exactemente igual, lo cual indica una recuperación correcta.

Finalmente tratamos de recuperar el patrón asociado a $\mathbf{x}^3$:

Para i=1 tenemos que $EM0s^1 = 0$ y $\mathbf{M1s}^1 \cap \mathbf{X0s}^3 = \{3,5\} \cap \{1,2,4\} = \phi$, entonces $YM_1$=1
Para i=2 tenemos que $EM0s^2 = 1$, por tanto $YM_2$=0
Para i=3 tenemos que $EM0s^3 = 0$ y $\mathbf{M1s}^3 \cap \mathbf{X0s}^3 = \{3\} \cap \{1,2,4\} = \phi$, entonces $YM_3$=1
Para i=4 tenemos que $EM0s^4 = 0$ y $\mathbf{M1s}^4 \cap \mathbf{X0s}^3 = \{2,3\} \cap \{1,2,4\} = \{2\}$ entonces $YM_4$=0
Para i=5 tenemos que $EM0s^5 = 0$ y $\mathbf{M1s}^5 \cap \mathbf{X0s}^3 = \{2,3\} \cap \{1,2,4\} = \{2\}$ entonces $YM_5$=0
Para i=6 tenemos que $EM0s^6 = 0$ y $\mathbf{M1s}^6 \cap \mathbf{X0s}^3 = \{3\} \cap \{1,2,4\} = \phi$ entonces $YM_5$=1

Entonces el vector final de salida es:

$$\mathbf{YM}(\mathbf{x}^3) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Comparando el el vector de salida **YM** con el vector $\mathbf{y}^3$ vemos que es exactemente igual, lo cual indica una recuperación correcta.

Tenemos que se recuperaron correctamente 2 de los 3 patrones aprendidos, y que en el que falló sólo difiere en 2 posiciones, exactamente el mismo resultado que tenemos con los algoritmos originales

### Fase de Recuperación Memoria Min

**Paso 1** De la memoria **W**, obtenemos los vectores **EW2s, W1s**$^i$, como se definió en 4.1, para ilustrarlo pondremos a la transpuesta de **EW2s** al lado de la memoria Max **W**.

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (\mathbf{EW0s})^t = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{aligned} \mathbf{W1s}^1 &= \{1,4,5\} \\ \mathbf{W1s}^2 &= \phi \\ \mathbf{W1s}^3 &= \{1,2,3,4,5\} \\ \mathbf{W1s}^4 &= \phi \\ \mathbf{W1s}^5 &= \{1,2,4\} \\ \mathbf{W1s}^6 &= \phi \end{aligned}$$

**Paso 2** Obtenemos el vector **X1s** como se definió en 4.1.

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{X1s}^1 = \{1,2,3,4,5\} \qquad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{X1s}^2 = \{2,3\} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \mathbf{X1s}^3 = \{3,5\}$$

**Paso 3** El valor del patrón de salida **y** en la posición i, cuando el patrón de entrada **x** es operado sobre una memoria asociativa **W**, es:

$YW_i = 1$ si $EW2s_i = 1$ por teorema 12
$YW_i = 0$ si $EW2s_i = 0$ y $\mathbf{W1s}^i = \emptyset$ por teorema 13
$YW_i = 1$ si $EW2s_i = 0$ y $\mathbf{W1s}^i \cap \mathbf{X1s} \neq \emptyset$ por teorema 14

$YW_i$=0 si $EW2s_i$ = 0 y $\mathbf{W1s}^i \cap \mathbf{X1s} = \emptyset$ por teorema 14

Empezaremos tratando de recuperar $\mathbf{x}^1$

Para i=1 tenemos que $EW2s_1 = 0$ y $\mathbf{W1s}^1 \cap \mathbf{X1s}^1 = \{1,4,5\} \cap \{1,2,3,4,5\} = \{1,4,5\}$, entonces $YW_1$=1
Para i=2 tenemos que $EW2s_2 = 0$ y $\mathbf{W1s}^2 \cap \mathbf{X1s}^1 = \phi \cap \{1,2,3,4,5\} = \phi$, entonces $YW_2$=0
Para i=3 tenemos que $EW2s_3 = 0$ y $\mathbf{W1s}^3 \cap \mathbf{X1s}^1 = \{1,2,3,4,5\} \cap \{1,2,3,4,5\} = \{1,2,3,4,5\}$, entonces $YW_3$=1
Para i=4 tenemos que $EW2s_4 = 0$ y $\mathbf{W1s}^4 \cap \mathbf{X1s}^1 = \phi \cap \{1,2,3,4,5\} = \phi$, entonces $YW_4$=0
Para i=5 tenemos que $EW2s_5 = 0$ y $\mathbf{W1s}^5 \cap \mathbf{X1s}^1 = \{1,2,4\} \cap \{1,2,3,4,5\} = \{1,2,4\}$, entonces $YW_5$=1
Para i=6 tenemos que $EW2s_6 = 0$ y $\mathbf{W1s}^6 \cap \mathbf{X1s}^1 = \phi \cap \{1,2,3,4,5\} = \phi$, entonces $YW_6$=0

Entonces el vector final de salida es:

$$\mathbf{YW}(\mathbf{x}^1) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Comparando el el vector de salida **YM** con el vector $\mathbf{y}^1$ vemos que es exactemente igual, lo cual indica una recuperación correcta.

Ahora tratamos de recuperar $\mathbf{x}^2$

Para i=1 tenemos que $EW2s_1 = 0$ y $\mathbf{W1s}^1 \cap \mathbf{X1s}^2 = \{1,4,5\} \cap \{2,3\} = \phi$, entonces $YW_1$=0
Para i=2 tenemos que $EW2s_2 = 0$ y $\mathbf{W1s}^2 \cap \mathbf{X1s}^2 = \phi \cap \{2,3\} = \phi$, entonces $YW_2$=0
Para i=3 tenemos que $EW2s_3 = 0$ y $\mathbf{W1s}^3 \cap \mathbf{X1s}^2 = \{1,2,3,4,5\} \cap \{2,3\} = \{2,3\}$, entonces $YW_3$=1
Para i=4 tenemos que $EW2s_4 = 0$ y $\mathbf{W1s}^4 \cap \mathbf{X1s}^2 = \phi \cap \{2,3\} = \phi$, entonces $YW_4$=0
Para i=5 tenemos que $EW2s_5 = 0$ y $\mathbf{W1s}^5 \cap \mathbf{X1s}^2 = \{1,2,4]\} \cap \{2,3\} = \{2\}$, entonces $YW_5$=1
Para i=6 tenemos que $EW2s_6 = 0$ y $\mathbf{W1s}^6 \cap \mathbf{X1s}^2 = \phi \cap \{2,3\} = \phi$, entonces $YW_6$=0

Entonces el vector final de salida es:

$$\mathbf{YW}(\mathbf{x}^2) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Comparando el el vector de salida **YM** con el vector $\mathbf{y}^2$ vemos que es diferente en las posiciones 4 y 6.

Ahora tratamos de recuperar $\mathbf{x}^3$

Para i=1 tenemos que $EW2s_1 = 0$ y $\mathbf{W1s}^1 \cap \mathbf{X1s}^3 = \{1,4,5\} \cap \{3,5\} = \{5\}$, entonces $YW_1$=1
Para i=2 tenemos que $EW2s_2 = 0$ y $\mathbf{W1s}^2 \cap \mathbf{X1s}^3 = \phi \cap \{3,5\} = \phi$, entonces $YW_2$=0
Para i=3 tenemos que $EW2s_3 = 0$ y $\mathbf{W1s}^3 \cap \mathbf{X1s}^3 = \{1,2,3,4,5\} \cap \{3,5\} = \{3,5\}$, entonces $YW_3$=1
Para i=4 tenemos que $EW2s_4 = 0$ y $\mathbf{W1s}^4 \cap \mathbf{X1s}^3 = \phi \cap \{3,5\} = \phi$, entonces $YW_4$=0
Para i=5 tenemos que $EW2s_5 = 0$ y $\mathbf{W1s}^5 \cap \mathbf{X1s}^3 = \{1,2,4\} \cap \{3,5\} = \phi$, entonces $YW_5$=0
Para i=6 tenemos que $EW2s_6 = 0$ y $\mathbf{W1s}^6 \cap \mathbf{X1s}^3 = \phi \cap \{3,5\} = \phi$, entonces $YW_6$=0

$$\mathbf{YW}(\mathbf{x}^3) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Comparando el el vector de salida **YM** con el vector $\mathbf{y}^3$ vemos que es diferente en la posición 6.

Como se puede observar el único patrón que se recuperó correctamente es $\mathbf{x}^1$, exactamente los mismos resultados obtenidos aquí se obtienen con los algoritmos originales.

### 4.3.2 Memorias AutoAsociativas Alfa Beta Simplificadas

Las memorias autoasociativas tienen la forma de $(\mathbf{x}^\mu, \mathbf{x}^\mu)$ en todas sus asociaciones. Por lo tanto los vectores **X1s** y **Y1s** son iguales, al igual que **X0s** y **Y0s**. Como resultado de este hecho los vectores **EY0s** y **EY1s** son iguales a **EX0s** y **EX1s** respectivamente. Por este

hecho podemos tomar el algoritmo de aprendizaje de las memorias heteroasociativas y prescindir del uso de los conjuntos **Y0s**, **Y1s**, y de los vectores **EY0s** y **EY1s**,

Se tienen dos tipos de memorias autoasociativas Alfa-Beta: tipo Max, denotadas por **M** con $m_{ij}$ como su ij-ésima componente y las tipo Min, denotadas por **W** con $w_{ij}$ como su ij-ésima componente.

### 4.3.2.1 Memoria AutoAsociativa Max

#### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ Obtenemos los vectores $\mathbf{X0s}^{\mu}$, $\mathbf{X1s}^{\mu}$ para cada par en el conjunto fundamental de patrones $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu})| \mu=1, 2,..., p\}$. Además también hacemos:

$EX0s_k = 1$ donde $k \in \mathbf{X0s}^{\mu}$

$EX1s_k = 1$ donde $k \in \mathbf{X1s}^{\mu}$

**Paso 2.** Para cada $\mu \in \{1,2,3,...,p\}$ hacemos:

$m_{ij} = 2 \ \forall ij \in \{ \mathbf{X1s}^{\mu} \times \mathbf{X0s}^{\mu} \}$ por teorema 7

**Paso 3** Como paso final, después de haber realizado los pasos 1,2 $p$ veces, revisamos la memoria Max para poner los valores adecuados en las posiciones que faltan.

$m_{ij} = 2 \leftrightarrow m_{ij} = 2$

$m_{ij} = 1 \leftrightarrow m_{ij} \neq 2$ y $(EX1s_i = 1$ or $EX0s_j = 1)$ por teoremas 1-2

$m_{ij} = 0 \leftrightarrow m_{ij} \neq 2$ and $(EX1s_i \neq 1$ and $EX0s_j \neq 1)$ por teorema 3

#### Fase de Recuperación

**Paso 1** De la memoria **M**, obtenemos los vectores **M0s**, **M1s**, como se definio en 4.1

**Paso 2** Obtenemos el vector **X0s** como se definio en 4.1.

**Paso 3** El valor del patron de salida **y** en la posición i, cuando el patrón de entrada **x** es operado sobre una memoria asociativa Max **M**, es:

$YM_i = 0$ si $EM0s_i = 1$ por teorema 1

$YM_i = 1$ si $EM0s_i = 0$ y $\mathbf{M1s}^i = \emptyset$ por teorema 2

$YM_i = 0$ si $EM0s_i = 0$ y $\mathbf{M1s}^i \cap \mathbf{X0s} \neq \emptyset$ por teorema 3

$YM_i = 1$ si $EM0s_i = 0$ y $\mathbf{M1s}^i \cap \mathbf{X0s} = \emptyset$ por teorema 3

### 4.3.2.2 Memoria Autoasociativa Min

#### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{y}^{\mu})$ Obtener los vectores $\mathbf{X1s}^{\mu}$, $\mathbf{X0s}^{\mu}$ para cada par en el conjunto fundamental de patrones $\{(X^{\mu}, Y^{\mu})| \mu=1, 2,..., p)$
Además en este paso también hacemos:

$EX0s_k$ =1 donde k $\in$ $\mathbf{X0s}^{\mu}$
$EX1s_k$ =1 donde k $\in$ $\mathbf{X1s}^{\mu}$

**Paso 2.** Para cada $\mu \in \{1,2,3..p\}$ hacemos:
$w_{ij}$= 0 $\forall i,j \in \{ \mathbf{X0s}^{\mu} \times \mathbf{X1s}^{\mu} \}$ por el teorema 8

**Paso 3** Como paso final, después de haber realizado el paso 2 $\mu$ veces, revisamos la memoria Min para poner los valores adecuados en las posiciones que faltan .
$w_{ij}$= 0 $\leftrightarrow$ $w_{ij}$ =0
$w_{ij}$= 1 $\leftrightarrow$ $w_{ij} \neq 0$ and ($EX0s_i$ =1 or $EX1s_j$ =1) por teoremas 4-5
$w_{ij}$= 2 $\leftrightarrow$ $w_{ij} \neq 0$ and ($EX0s_i \neq 1$ and $EX1s_j \neq 1$ ) por teorema 6

### Fase de Recuperación

**Paso 1** De la memoria Min **W**, obtenemos **W1s**, **EW2s** como se definió en 4.1
**Paso 2** Obtenemos el vector **X1s** como se definió en 4.1.
**Paso 3** El valor del patron de salida **YW** en la posición $YM_i$, cuando el patrón de entrada xes operado sobre una memoria asociativa **W**, es:
$YW_i$ =1 si $EW2s_i$ = 1 por teorema 4
$YW_i$ =0 si $EW2s_i$ = 0 y $\mathbf{W1s}^i = \emptyset$ por teorema 5
$YW_i$ =1 si $EW2s_i$ = 0 y $\mathbf{W1s}^i \cap \mathbf{X1s} \neq \emptyset$ por teorema 6
$YW_i$ =0 si $EW2s_i$ = 0 y $\mathbf{W1s}^i \cap \mathbf{X1s} = \emptyset$ por teorema 6

## 4.3.2.3 Algoritmo Combinado de las Memorias Autoasociativas Simplificadas

### Fase de Aprendizaje

Dado que el paso uno es el mismo en ambos tipos de memoria es posible realizar ese paso una sola vez y utilizar los datos para ambos tipos de memoria. Entonces el algoritmo queda:

**Paso 1.** Para cada $\mu$ = 1, 2, ..., $p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{y}^{\mu})$ Obtener los vectores $\mathbf{X1s}^{\mu}$, $\mathbf{X0s}^{\mu}$ para cada par en el conjunto fundamental de patrones $\{(X^{\mu}, Y^{\mu})| \mu=1, 2,..., p)$
Además en este paso también hacemos:
$EX0s_k$ =1 donde k $\in$ $\mathbf{X0s}^{\mu}$
$EX1s_k$ =1 done k $\in$ $\mathbf{X1s}^{\mu}$

**Paso 2.** Para cada $\mu \in \{1,2,3..p\}$ hacemos:
$m_{ij}$= 2 $\forall ij \in \{ \mathbf{X1s}^{\mu} \times \mathbf{X0s}^{\mu} \}$ por teorema 7
$m_{ij}$= 0 $\forall i,j \in \{ \mathbf{X0s}^{\mu} \times \mathbf{X1s}^{\mu} \}$ por el teorema 8

**Paso 3** Como paso final, después de haber realizado los pasos 1,2 $p$ veces, revisamos las memorias Max y Min para poner los valores adecuados en las posiciones que faltan.
$m_{ij}$ = 2 $\leftrightarrow$ $m_{ij}$ =2
$m_{ij}$ = 1 $\leftrightarrow$ $m_{ij} \neq 2$ y ($EX1s_i$ =1 or $EX0s_j$ =1) por teoremas 1-2

$m_{ij} = 0 \leftrightarrow m_{ij} \neq 2$ and (EX1s $_i \neq 1$ and EX0s $_j \neq 1$) por teorema 3
$w_{ij} = 0 \leftrightarrow w_{ij} = 0$
$w_{ij} = 1 \leftrightarrow w_{ij} \neq 0$ and ($EX0s_i = 1$ or $EX1s_j = 1$) por teoremas 4-5
$w_{ij} = 2 \leftrightarrow w_{ij} \neq 0$ and ($EX0s_i \neq 1$ and $EX1s_j \neq 1$ ) por teorema 6

Dado que la fase de recuperación no tiene pasos en común, ésta permanece igual.

## Ejemplo de las memorias autoasociativas Alfa-Beta Simplificadas

El propósito de esta sección es la de ilustrar mediante un ejemplo el funcionamiento de las memorias autoasociativas Alfa-Beta Simplificadas de ambos tipos. Aprovechando el ejemplo para comparar los algoritmos simplificados con los originales, usaremos el mismo conjunto fundamental que se usó en el ejemplo de los algoritmos originalesen el apéndice D. Para este ejemplo utilizaremos patrones de entrada de dimensión 4 y el siguiente conjunto fundamental:

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

### Fase de Aprendizaje

**Paso 1.** Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ Obtener los vectores $\mathbf{X1s}^\mu$, $\mathbf{X0s}^\mu$ para cada par en el conjunto fundamental de patrones $\{(X^\mu, Y^\mu) | \mu=1, 2,..., p)$
Además en este paso también hacemos:
$EX0s_k = 1$ donde $k \in \mathbf{X0s}^\mu$
$EX1s_k = 1$ done $k \in \mathbf{X1s}^\mu$

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \begin{matrix} \mathbf{X0s}^1 = \{1,3\} \\ \mathbf{X1s}^1 = \{2,4\} \end{matrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{matrix} \mathbf{X0s}^2 = \{3,4\} \\ \mathbf{X1s}^2 = \{1,2\} \end{matrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{matrix} \mathbf{X0s}^3 = \{1,2,3\} \\ \mathbf{X1s}^3 = \{4\} \end{matrix}$$

**EX0s**=[1,1,1,1]
**EX1s**=[1,1,0,1]

**Paso 2.** Para toda $\mu \in \{1,2,3..p\}$ hacemos:
$m_{ij} = 2 \quad \forall i,j \in \{ \mathbf{X1s}^\mu \times \mathbf{X0s}^\mu \}$ por teorema 7
$w_{ij} = 0 \quad \forall i,j \in \{ \mathbf{X0s}^\mu \times \mathbf{X1s}^\mu \}$ por el teorema 8

Obtenemos $\mathbf{X1s}^\mu$ X $\mathbf{X0s}^\mu$ para toda $\mu \in \{1,2,3..p\}$
$\mathbf{X1s}^1 \times \mathbf{X0s}^1 = [2,4] \times [1,3] = \{(2,1),(2,3), (4,1), (4,3)\}$

$\mathbf{X1s}^2 \times \mathbf{X0s}^2 = [1,2] \times [3,4] = \{(1,3),(1,4),(2,3),(2,4)\}$
$\mathbf{X1s}^3 \times \mathbf{X0s}^3 = [4] \times [1,2,3] = \{(4,1),(4,2),(4,3)\}$

Introducimos las posicones en las que $m_{ij}=2$

$$\mathbf{M} = \begin{pmatrix} & & 2 & 2 \\ 2 & & 2 & 2 \\ & & & \\ 2 & 2 & 2 & \end{pmatrix}$$

Obtenemos $\mathbf{X0s}^{\mu} \times \mathbf{X1s}^{\mu}$ para toda $\mu \in \{1,2,3..p\}$

$\mathbf{X0s}^1 \times \mathbf{X1s}^1 = \{1,3\} \times \{2,4\} = \{(1,2),(1,4),(3,2),(3,4)\}$
$\mathbf{X0s}^2 \times \mathbf{X1s}^2 = \{3,4\} \times \{1,2\} = \{(3,1),(3,2),(4,1),(4,2)\}$
$\mathbf{X0s}^3 \times \mathbf{X1s}^3 = \{1,2,3\} \times \{4\} = \{(1,4),(2,4),(3,4)\}$

Introducimos las posiciones en las que $w_{ij}=0$

$$\mathbf{W} = \begin{pmatrix} & 0 & & 0 \\ & & & 0 \\ 0 & 0 & & 0 \\ 0 & 0 & & \end{pmatrix}$$

**Paso 3** Como paso final, después de haber realizado los pasos 1,2 $p$ veces, revisamos las memorias Max y Min para poner los valores adecuados en las posiciones que faltan.

$m_{ij} = 2 \leftrightarrow m_{ij} = 2$
$m_{ij} = 1 \leftrightarrow m_{ij} \neq 2$ y ($EX1s_i = 1$ or $EX0s_j = 1$) por teoremas 1-2
$m_{ij} = 0 \leftrightarrow m_{ij} \neq 2$ y ($EX1s_i \neq 1$ and $EX0s_j \neq 1$) por teorema 3

Para facilitar la comprensión pondremos los vecotores **EX0s** y **EX1s**, como en este caso E**X1s** afecta a los renglones pondremos la transpuesta de éste

$$\mathbf{EX0s} = [1 \quad 1 \quad 1 \quad 1]$$

$$(\mathbf{EX1s})^t = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

Por lo tanto la memoria **M** queda:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

Para facilitar la comprensión con la memoria **W**, tambien pondremos los vectores **EX0s** y **EX1s**, pero en este caso E**X0s** es el que afecta a los renglones, por lo tanto pondremos la transpuesta de éste

$w_{ij}= 0 \leftrightarrow w_{ij} =0$

$w_{ij}= 1 \leftrightarrow w_{ij} \neq 0$ and ($EX0s_i = 1$ or $EX1s_j = 1$) por teoremas 4-5

$w_{ij}= 2 \leftrightarrow w_{ij} \neq 0$ and ($EX0s_i \neq 1$ and $EX1s_j \neq 1$ ) por teorema 6

$$\mathbf{EX1s} = [1 \quad 1 \quad 0 \quad 1]$$

$$(\mathbf{EX0s})^t = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Por lo tanto la memoria **W** queda:

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

### Fase de Recuperacion Memoria Max

**Paso 1** De la memoria **M**, obtenemos los vectores **EM0s, M1s**$^i$, como se definió en 4.1, para ilustrarlo pondremos a la transpuesta de **EM0s** al lado de la memoria Max **M.**

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} \qquad (\mathbf{EM0s})^t = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{M1s}^1 = \{1,2\}$$

$$\mathbf{M1s}^2 = \{2\}$$

$$\mathbf{M1s}^3 = \{1,2,3,4\}$$

$$\mathbf{M1s}^4 = \{4\}$$

**Paso 2** Obtenemos los vectores **X0s, X1s** de los patrones de entrada como se definió en 4.1.

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{X0s}^1 = \{1,3\} \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{X0s}^2 = \{3,4\} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{X0s}^3 = \{1,2,3\}$$

**Paso 3** El valor del patron de salida **YM** en la posición $YM_i$, cuando el patrón de entrada **x** es operado sobre una memoria asociativa Max **M,** es:

$YM_i$=0 si $EM0s_i = 1$
$YM_i$=1 si $EM0s_i = 0$ y $\mathbf{M1s}^i = \emptyset$
$YM_i$=0 si $EM0s_i = 0$ y $\mathbf{M1s}^i \cap \mathbf{X0s} \neq \emptyset$
$YM_i$=1 si $EM0s_i = 0$ y $\mathbf{M1s}^i \cap \mathbf{X0s} = \emptyset$

Empezaremos tratando de recuperar el patron asociado a $\mathbf{x}^1$:

Para i=1 tenemos que $EM0s^1 = 0$ y $\mathbf{M1s}^1 \cap \mathbf{X0s}^1 = \{1,2\} \cap \{1,3\} = \{1\}$, entonces $YM_1$=0
Para i=2 tenemos que $EM0s^2 = 0$ y $\mathbf{M1s}^2 \cap \mathbf{X0s}^1 = \{2\} \cap \{1,3\} = \phi$, entonces $YM_2$=1
Para i=3 tenemos que $EM0s^3 = 0$ y $\mathbf{M1s}^3 \cap \mathbf{X0s}^1 = \{1,2,3,4\} \cap \{1,3\} = \{1,3\}$, entonces $YM_3$=0
Para i=4 tenemos que $EM0s^4 = 0$ y $\mathbf{M1s}^4 \cap \mathbf{X0s}^1 = \{4\} \cap \{1,3\} = \phi$ entonces $YM_4$=1

Acomodando los valores en el vector **YM,** tenemos**:**

$$\mathbf{YM}(\mathbf{x}^1) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Comparando el patrón de salida **YM** con $\mathbf{x}^1$ Podemos observar que son iguales, por lo cual hemos obtenido una recuperación correcta.

Ahora recuperaremos el patron asociado a $\mathbf{x}^2$:

Para i=1 tenemos que $EM0s^1 = 0$ y $\mathbf{M1s}^1 \cap \mathbf{X0s}^2 = \{1,2\} \cap \{3,4\} = \phi$, entonces $YM_1$=1
Para i=2 tenemos que $EM0s^2 = 0$ y $\mathbf{M1s}^2 \cap \mathbf{X0s}^2 = \{2\} \cap \{3,4\} = \phi$, entonces $YM_2$=1
Para i=3 tenemos que $EM0s^3 = 0$ y $\mathbf{M1s}^3 \cap \mathbf{X0s}^2 = \{1,2,3,4\} \cap \{3,4\} = \{3,4\}$, entonces $YM_3$=0
Para i=4 tenemos que $EM0s^4 = 0$ y $\mathbf{M1s}^4 \cap \mathbf{X0s}^2 = \{4\} \cap \{3,4\} = \{4\}$ entonces $YM_4$=0

Acomodando los valores en el vector **YM,** tenemos**:**

$$\mathbf{YM}(\mathbf{x}^2) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Comparando el patrón de salida **YM** con $\mathbf{x}^2$ Podemos observar que son iguales, por lo cual hemos obtenido una recuperación correcta.

Ahora recuperaremos el patron asociado a $\mathbf{x}^3$:

Para i=1 tenemos que $EM0s^1 = 0$ y $\mathbf{M1s}^1 \cap \mathbf{X0s}^3 = \{1,2\} \cap \{1,2,3\} = \{1,2\}$, entonces $YM_1$=0
Para i=2 tenemos que $EM0s^2 = 0$ y $\mathbf{M1s}^2 \cap \mathbf{X0s}^3 = \{2\} \cap \{1,2,3\} = \{2\}$, entonces $YM_2$=0
Para i=3 tenemos que $EM0s^3 = 0$ y $\mathbf{M1s}^3 \cap \mathbf{X0s}^3 = \{1,2,3,4\} \cap \{1,2,3\} = \{1,2,3\}$, entonces $YM_3$=0
Para i=4 tenemos que $EM0s^4 = 0$ y $\mathbf{M1s}^4 \cap \mathbf{X0s}^3 = \{4\} \cap \{1,2,3\} = \phi$ entonces $YM_4$=1

Acomodando los valores en el vector **YM,** tenemos**:**

$$\mathbf{YM} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Comparando el patrón de salida **YM** con $\mathbf{x}^2$ Podemos observar que son iguales, por lo cual hemos obtenido una recuperación correcta.

### Fase de Recuperacion Memoria Min

**Paso 1** De la memoria Min **W**, obtenemos **W1s**, **EW2s** como se definió en 4.1, para facilitar la comprension pondremos la transpuesta de **EW2s** al lado de la memoria **W**.

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad (\mathbf{EW2s})^t = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{aligned} \mathbf{W1s}^1 &= \{1,3\} \\ \mathbf{W1s}^2 &= \{1,2,3\} \\ \mathbf{W1s}^3 &= \{3\} \\ \mathbf{W1s}^4 &= \{3,4\} \end{aligned}$$

**Paso 2** Obtenemos el vector **X1s** como se definió en 4.1.

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{X1s}^1 = \{2,4\} \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{X1s}^2 = \{1,2\} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{X1s}^3 = \{4\}$$

**Paso 3** El valor del patron de salida **YW** en la posición $YM_i$, cuando el patrón de entrada xes operado sobre una memoria asociativa **W**, es:

$YW_i$ =1 si $EW2s_i$ = 1 por teorema 4
$YW_i$ =0 si $EW2s_i$ = 0 y $\mathbf{W1s}^i = \emptyset$ por teorema 5
$YW_i$ =1 si $EW2s_i$ = 0 y $\mathbf{W1s}^i \cap \mathbf{X1s} \neq \emptyset$ por teorema 6
$YW_i$ =0 si $EW2s_i$ = 0 y $\mathbf{W1s}^i \cap \mathbf{X1s} = \emptyset$ por teorema 6

Empezaremos tratando de recuperar el patron asociado a $\mathbf{x}^1$:

Para i=1 tenemos que $EW0s^1 = 0$ y $\mathbf{W1s}^1 \cap \mathbf{X1s}^1 = \{1,3\} \cap \{2,4\} = \phi$, entonces $YM_1$=0
Para i=2 tenemos que $EW0s^2 = 0$ y $\mathbf{W1s}^2 \cap \mathbf{X1s}^1 = \{1,2,3\} \cap \{2,4\} = \{2\}$, entonces $YM_2$=1
Para i=3 tenemos que $EW0s^3 = 0$ y $\mathbf{W1s}^3 \cap \mathbf{X1s}^1 = \{3\} \cap \{2,4\} = \phi$, entonces $YM_3$=0
Para i=4 tenemos que $EW0s^4 = 0$ y $\mathbf{W1s}^4 \cap \mathbf{X1s}^1 = \{3,4\} \cap \{2,4\} = \{4\}$ entonces $YM_4$=1

Acomodando los valores en el vector **YM,** tenemos**:**

$$\mathbf{YW} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Comparando el vector **YW** con el vector $\mathbf{x}^1$ se observa que tenemos una recuperación correcta.
Ahora trataremos de recuperar el patron asociado a $\mathbf{x}^2$:

Para i=1 tenemos que $EW0s^1 = 0$ y $\mathbf{W1s}^1 \cap \mathbf{X1s}^1 = \{1,3\} \cap \{1,2\} = \{1\}$, entonces $YM_1$=1
Para i=2 tenemos que $EW0s^2 = 0$ y $\mathbf{W1s}^2 \cap \mathbf{X1s}^1 = \{1,2,3\} \cap \{1,2\} = \{1,2\}$, entonces $YM_2$=1
Para i=3 tenemos que $EW0s^3 = 0$ y $\mathbf{W1s}^3 \cap \mathbf{X1s}^1 = \{3\} \cap \{1,2\} = \phi$, entonces $YM_3$=0
Para i=4 tenemos que $EW0s^4 = 0$ y $\mathbf{W1s}^4 \cap \mathbf{X1s}^1 = \{3,4\} \cap \{1,2\} = \phi$ entonces $YM_4$=0

Acomodando los valores en el vector **YM,** tenemos**:**

$$\mathbf{YW} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Comparando el vector **YW** con el vector $\mathbf{x}^2$ se observa que tenemos una recuperación correcta.

Finalmente trataremos de recuperar $x^3$

Para i=1 tenemos que $EW0s^1 = 0$ y $\mathbf{W1s}^1 \cap \mathbf{X1s}^3 = \{1,3\} \cap \{4\} = \phi$, entonces $YM_1$=0
Para i=2 tenemos que $EW0s^2 = 0$ y $\mathbf{W1s}^2 \cap \mathbf{X1s}^3 = \{1,2,3\} \cap \{4\} = \phi$, entonces $YM_2$=0
Para i=3 tenemos que $EW0s^3 = 0$ y $\mathbf{W1s}^3 \cap \mathbf{X1s}^3 = \{3\} \cap \{4\} = \phi$, entonces $YM_3$=0
Para i=4 tenemos que $EW0s^4 = 0$ y $\mathbf{W1s}^4 \cap \mathbf{X1s}^3 = \{3,4\} \cap \{4\} = \{4\}$ entonces $YM_4$=1

Acomodando los valores en el vector **YM,** tenemos**:**

$$\mathbf{YW} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Comparando el vector **YW** con el vector $\mathbf{x}^3$ se observa que tenemos una recuperación correcta.

## *4.4 Relaciones entre las memorias del tipo Max y las del tipo Min*

En las secciones anteriores se presentó una forma de crear algoritmos simplificados de aprendizaje y recuperación. Una aportación de esta tesis es la de incluir teoremas que dan la posibilidad de ahorrarnos el procedimiento de aprendizaje de una de las memorias y obtenerla a partir de la otra. A continuación se definirá un operador que ayudará en esta tarea.

### 4.4.1 El operador φ

El operador $\varphi$:B→B, con **B**={0,1,2}, se define de la siguiente manera:

$$\varphi(b) = 2 - b \ , b \in B$$

Desarrollando para todos los posibles valores en el conjunto B, tenemos:

$$\varphi(0) = 2 - 0 = 2$$
$$\varphi(1) = 2 - 1 = 1$$
$$\varphi(2) = 2 - 2 = 0$$

Otra forma de representarlo, es:

$$\varphi(b) = \begin{cases} 0 & si \quad b = 2 \\ 1 & si \quad b = 1 \\ 2 & si \quad b = 0 \end{cases}$$

**Fig 4.1** Operador $\varphi$

### 4.4.2 Conversión de una memoria autoasociativa Max a una memoria autoasociativa Min y viceversa

En esta sección se demuestra un teorema que permite obtener una memoria autoasociativa Min a partir de una memoria autoasociativa Max. Esto nos da la gran ventaja de que en la fase de aprendizaje sólo se entrena a una sola memoria y la otra se obtiene al final mediante los valores de la memoria entrenada. Obviamente, el tiempo necesario para obtener ambas memorias será aproximadamente la mitad del correspondiente a los algoritmos originales.

***Teorema 15*** Sea **M** una memoria autoasociativa Alfa-Beta del tipo Max y $m_{ij}$ su ij-ésima componente y **W** una memoria autoasociativa Alfa-Bata del tipo Min con $w_{ij}$ como su ij-ésima compoenente. Es posible obtener los valores de la memoria **M** a partir de la memoria **W** y viceversa de la siguiente forma.

$$\mathbf{W} = \varphi\left(\mathbf{M}^t\right) \qquad \mathbf{M} = \varphi\left(\mathbf{W}^t\right)$$

Desarrollando tenemos:

$$\mathbf{W} = \begin{pmatrix} \varphi(m_{11}) & \varphi(m_{21}) & \cdots & \varphi(m_{j1}) & \cdots & \varphi(m_{n1}) \\ \varphi(m_{12}) & \varphi(m_{22}) & \cdots & \varphi(m_{j2}) & \cdots & \varphi(m_{n2}) \\ \vdots & \vdots & & \vdots & & \vdots \\ \varphi(m_{1i}) & \varphi(m_{2i}) & \cdots & \varphi(m_{ji}) & \cdots & \varphi(m_{ni}) \\ \vdots & \vdots & & & & \vdots \\ \varphi(m_{1n}) & \varphi(m_{2n}) & \cdots & \varphi(m_{jn}) & \cdots & \varphi(m_{nn}) \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} \varphi(w_{11}) & \varphi(w_{21}) & \cdots & \varphi(w_{j1}) & \cdots & \varphi(w_{n1}) \\ \varphi(w_{12}) & \varphi(w_{22}) & \cdots & \varphi(w_{j2}) & \cdots & \varphi(w_{n2}) \\ \vdots & \vdots & & \vdots & & \vdots \\ \varphi(w_{1i}) & \varphi(w_{2i}) & \cdots & \varphi(w_{ji}) & \cdots & \varphi(w_{ni}) \\ \vdots & \vdots & & & & \vdots \\ \varphi(w_{1n}) & \varphi(w_{2n}) & \cdots & \varphi(w_{jn}) & \cdots & \varphi(w_{nn}) \end{pmatrix}$$

Recordando el funcionamiento del operador $\varphi$, tenemos que:

$$w_{ij} = \begin{cases} 0 \leftrightarrow m_{ji} = 2 \\ 1 \leftrightarrow m_{ji} = 1 \\ 2 \leftrightarrow m_{ji} = 0 \end{cases} \qquad m_{ij} = \begin{cases} 0 \leftrightarrow w_{ji} = 2 \\ 1 \leftrightarrow w_{ji} = 1 \\ 2 \leftrightarrow w_i i = 0 \end{cases}$$

En sí las 2 relaciones son las mismas, dado que los distintos casos se pueden ver como los extremos de una misma bicondicional, por ejemplo:

$$w_{ij} = 0 \leftrightarrow m_{ji} = 2$$

$$m_{ji} = 2 \leftrightarrow w_{ij} = 0$$

Invirtiendo los índices tnenemos:

$$m_{ij} = 2 \leftrightarrow w_{ji} = 0$$

De igual forma se cumple para los otros 2 casos.

**Demostración**

Para realizar la demostración de este teorema demostraremos los 3 casos posibles,

### 1er caso.

La unica forma en que una memoria autoasociativa Min **W** pueda tener el valor de 0 es que en la memoria autoasociativa Max **M** la posición transpuesta tenga el valor de 2 y viceversa

$$w_{ij} = 0 \leftrightarrow m_{ji} = 2$$

A) $w_{ij} = 0 \rightarrow m_{ji} = 2$

$$\begin{gathered} w_{ij} = 0 \\ \rightarrow \exists \mu \text{ tal que } \alpha\left(x_i^\mu, x_j^\mu\right) = 0 \\ \rightarrow \exists \mu \text{ tal que } \mathrm{x}_i^\mu = 0, x_j^\mu = 1 \\ \rightarrow \exists \mu \text{ tal que } \alpha\left(x_j^\mu, x_i^\mu\right) = 2 \\ \rightarrow m_{ji} = 2 \end{gathered}$$

B) $m_{ji} = 2 \rightarrow w_{ij} = 0$

$$\begin{gathered} m_{ji} = 2 \\ \rightarrow \exists \mu \text{ tal que } \alpha\left(x_j^\mu, x_i^\mu\right) = 2 \\ \rightarrow \exists \mu \text{ tal que } \mathrm{x}_j^\mu = 1, x_i^\mu = 0 \\ \rightarrow \exists \mu \text{ tal que } \alpha\left(x_i^\mu, x_j^\mu\right) = 0 \\ \rightarrow w_{ij} = 0 \end{gathered}$$

**2do Caso.**

La única forma en que una memoria Min **W** pueda tener el valor de 1 es que en la memoria Max **M** la posición transpuesta tenga el valor de 1 y viceversa

$$w_{ij} = 1 \leftrightarrow m_{ji} = 1$$

A) $w_{ij} = 1 \rightarrow m_{ji} = 1$

$$\begin{gathered} w_{ij} = 1 \\ \rightarrow \neg\exists\mu \text{ tal que } \alpha\left(x_i^\mu, x_j^\mu\right) = 0 \\ \rightarrow \neg\exists\mu \text{ tal que } \mathrm{x}_i^\mu = 0, x_j^\mu = 1 \\ \rightarrow \neg\exists\alpha\left(x_j^\mu, x_i^\mu\right) = 2 \end{gathered}$$

Además:

$$\begin{gathered} w_{ij} = 1 \\ \rightarrow \exists\mu \text{ tal que } \alpha\left(x_i^\mu, x_j^\mu\right) = 1 \\ \rightarrow (x_i^\mu = 1, x_j^\mu = 1) \text{ o } (x_i^\mu = 0, x_j^\mu = 0) \\ \rightarrow \alpha\left(x_i^\mu, x_j^\mu\right) = 1 \end{gathered}$$

Al no poder existir un 2 en la posición $m_{ij}$, entonces el maximo es 1.

B) $m_{ji} = 1 \rightarrow w_{ij} = 1$

$$\begin{gathered} m_{ji} = 1 \\ \rightarrow \neg\exists\mu \text{ tal que } \alpha\left(x_j^\mu, x_i^\mu\right) = 2 \\ \rightarrow \neg\exists\mu \text{ tal que } \mathrm{x}_j^\mu = 1, x_i^\mu = 0 \\ \rightarrow \neg\exists\alpha\left(x_i^\mu, x_j^\mu\right) = 0 \end{gathered}$$

Además:

$$\begin{gathered} m_{ji} = 1 \\ \rightarrow \exists\mu \text{ tal que } \alpha\left(x_j^\mu, x_i^\mu\right) = 1 \\ \rightarrow (x_j^\mu = 1, x_i^\mu = 1) \text{ o } (x_j^\mu = 0, x_i^\mu = 0) \\ \rightarrow \alpha\left(x_i^\mu, x_j^\mu\right) = 1 \end{gathered}$$

Al no poder existir un 0 en la posición $w_{ji}$, entonces el mínimo es 1.

**3er caso.**

La única forma en que una memoria Min **W** pueda tener el valor de 2 es que en la memoria Max **M** la posición transpuesta tenga el valor de 0 y viceversa

$$\rightarrow w_{ij} = 2 \leftrightarrow w_{ji} = 0$$

A) $w_{ij} = 2 \rightarrow m_{ji} = 0$

$$\begin{gathered} w_{ij} = 2 \\ \rightarrow \left(x_i^{\mu}, x_j^{\mu}\right) = 2 \ \forall \ \mu \\ \rightarrow x_i^{\mu} = 1, x_j^{\mu} = 0 \ \forall \ \mu \\ \rightarrow \left(x_i^{\mu}, x_j^{\mu}\right) = 0 \ \forall \ \mu \\ \rightarrow m_{ji} = 0 \end{gathered}$$

B) $m_{ji} = 0 \rightarrow w_{ij} = 2$

$$\begin{gathered} m_{ji} = 0 \\ \rightarrow \left(x_j^{\mu}, x_i^{\mu}\right) = 0 \ \forall \ \mu \\ \rightarrow x_j^{\mu} = 0, x_i^{\mu} = 1 \ \forall \ \mu \\ \rightarrow \left(x_i^{\mu}, x_j^{\mu}\right) = 2 \ \forall \ \mu \\ \rightarrow w_{ij} = 2 \end{gathered}$$

Mediante la demostración de estos 3 lemas, el teorema 15 quedo demostrado. Este teorema se puede aplicar directamente, asi solo entrenamos una de las memorias y obtenemos la restante a partir de los valores de esta, sin temer que el valor en alguna de las posiciones quede indefinido.

### 4.4.2.1 Algoritmo Para Obtener Una Memoria Autoasociativa Min A Partir De Una Memoria Autoasociativa Max

**Paso 1** Hacemos **W**=**M**$^t$
**Paso 2** Aplicar $w_{ij} = \varphi(w_{ij}) \forall i, j \mid i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,n\}$

### 4.4.2.2 Algoritmo Para Obtener Una Memoria AutoAsociativa Max Min A Partir De Una Memoria AutoAsociativa Min

**Paso 1** Hacemos $\mathbf{M}=\mathbf{W}^t$
**Paso 2** Aplicar $m_{ij} = \varphi(m_{ij}) \forall i, j \mid i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,n\}$

#### Ejemplo

Para un mayor entendimiento, se aplicará el teorema a las memorias obtenidas del ejemplo de los algoritmos originales (vease apéndice D). Entonces tenemos que:

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

Primero obtendremos la memoria **W** a partir de la memoria **M**

**Paso 1** Hacemos $\mathbf{W}=\mathbf{M}^t$

$$\mathbf{W} = \mathbf{M}^t = \begin{pmatrix} 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

**Paso 2** Aplicar $w_{ij} = \varphi(w_{ij}) \forall i, j \mid i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$

$$\mathbf{W} = \varphi\mathbf{W} = \varphi\begin{pmatrix} 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{pmatrix} = \begin{pmatrix} \varphi(1) & \varphi(2) & \varphi(1) & \varphi(2) \\ \varphi(1) & \varphi(1) & \varphi(1) & \varphi(2) \\ \varphi(2) & \varphi(2) & \varphi(1) & \varphi(2) \\ \varphi(2) & \varphi(2) & \varphi(2) & \varphi(1) \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Como se puede ver, la memoria **W** obtenida es exactamente la misma que si hubiéramos seguido todo el procedimiento original.

Ahora obtendremos la memoria **W** a partir de la memoria **M**

**Paso 1** Hacemos $\mathbf{M}=\mathbf{W}^t$

$$\mathbf{M} = \mathbf{W}^t = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**Paso 2** Aplicar $w_{ij} = \varphi(w_{ij}) \forall i, j \mid i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$

$$\mathbf{M} = \varphi \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \varphi(1) & \varphi(1) & \varphi(0) & \varphi(0) \\ \varphi(0) & \varphi(1) & \varphi(0) & \varphi(0) \\ \varphi(1) & \varphi(1) & \varphi(1) & \varphi(1) \\ \varphi(0) & \varphi(0) & \varphi(0) & \varphi(1) \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

Como se puede ver, la memoria **M** obtenida es exactamente la misma que si hubiéramos seguido todo el procedimiento original.

### 4.4.3 Generación de una BAM a partir de las memorias Max y Min

Cuando se trabaja con memorias bidireccionales se tienen que aprender las relaciones en un sentido y en el sentido contrario. En otras palabras, las memorias se entrenan presentando las relaciones (**x**,**y**) donde el patrón **x** es el patrón de entrada y **y** es el patrón de salida para obtener las memorias **M** y **W** y posteriormente se obtienen las memorias **M** y **W** de la asociaciones inversas (**y**,**x**), representadas por **MR** y **WR.**

En esta sección se demuestra un teorema que permite obtener las memorias Min y Max de las asociaciones inversas (**y**,**x**) a partir de las memorias Min y Max de las asociaciones normales (**x**,**y**). Esto nos da la gran ventaja de que en la fase de aprendizaje sólo entrenamos las memorias en una dirección y nos ahorramos el proceso de aprendizaje de las memorias en la dirección contraria. Esto se refleja en el ahorro de una gran cantidad de tiempo en la fase de aprendizaje.

***Teorema 16*** Sea M una memoria asociativa del tipo Max y W una memoria asociativa del tipo Min, es posible obtener los valores de las memorias Min y Max de la asociación inversa denotadas por **WR** y **MR** siendo $wr_{ij}$ y $mr_{ij}$ sus componentes ij-ésimas, de la siguiente forma:

$$\mathbf{WR} = \varphi\left(\mathbf{M}^t\right) \qquad \mathbf{MR} = \varphi\left(\mathbf{W}^t\right)$$

Desarrollando tenemos:

$$\mathbf{WR} = \begin{pmatrix} \varphi(m_{11}) & \varphi(m_{21}) & \cdots & \varphi(m_{j1}) & \cdots & \varphi(m_{n1}) \\ \varphi(m_{12}) & \varphi(m_{22}) & \cdots & \varphi(m_{j2}) & \cdots & \varphi(m_{n2}) \\ \vdots & \vdots & & \vdots & & \vdots \\ \varphi(m_{1i}) & \varphi(m_{2i}) & \cdots & \varphi(m_{ji}) & \cdots & \varphi(m_{ni}) \\ \vdots & \vdots & & & & \vdots \\ \varphi(m_{1n}) & \varphi(m_{2n}) & \cdots & \varphi(m_{jn}) & \cdots & \varphi(m_{nn}) \end{pmatrix}$$

$$\mathbf{MR} = \begin{pmatrix} \varphi(w_{11}) & \varphi(w_{21}) & \cdots & \varphi(w_{j1}) & \cdots & \varphi(w_{n1}) \\ \varphi(w_{12}) & \varphi(w_{22}) & \cdots & \varphi(w_{j2}) & \cdots & \varphi(w_{n2}) \\ \vdots & \vdots & & \vdots & & \vdots \\ \varphi(w_{1i}) & \varphi(w_{2i}) & \cdots & \varphi(w_{ji}) & \cdots & \varphi(w_{ni}) \\ \vdots & \vdots & & & & \vdots \\ \varphi(w_{1n}) & \varphi(w_{2n}) & \cdots & \varphi(w_{jn}) & \cdots & \varphi(w_{nn}) \end{pmatrix}$$

Desarrollando se tiene:

$$rw_{ij} = \begin{cases} 0 \leftrightarrow (m_{ij})^t = 2 \\ 1 \leftrightarrow (m_{ij})^t = 1 \\ 2 \leftrightarrow (m_{ij})^t = 0 \end{cases} \qquad rm_{ij} = \begin{cases} 0 \leftrightarrow (w_{ij})^t = 2 \\ 1 \leftrightarrow (w_{ij})^t = 1 \\ 2 \leftrightarrow (w_{ij})^t = 0 \end{cases}$$

La demostración de este teorema se sigue de la misma manera que la del teorema 15.

### 4.4.3.1 Algoritmo Para Obtener La Memoria Heteroasociativa Max De La Asociacion Inversa A Partir De Una Heroasociativa Min

**Paso 1** Hacemos **MR**=$\mathbf{W}^t$
**Paso 2** Aplicar $mr_{ij} = \varphi(mr_{ij}) \forall i, j \mid i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$

### 4.4.3.2 Algoritmo Para Obtener La Memoria Heteroasociativa Min De La Asociacion Inversa A Partir De Una Heroasociativa Max

**Paso 1** Hacemos **WR**=$\mathbf{M}^t$
**Paso 2** Aplicar $wr_{ij} = \varphi(wr_{ij}) \forall i, j \mid i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$

## Ejemplo

En la seccion apéndice D se da un ejemplo con el siguiente conjunto fudamental, donde se obtuvieron las memorias que corresponden a las asociaciones $\left(\mathbf{x}^{\mu},\mathbf{y}^{\mu}\right)$

$$\mathrm{x}^1=\begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix}\quad \mathbf{y}^1=\begin{pmatrix}1\\0\\1\\0\\1\\0\end{pmatrix}\qquad \mathbf{x}^2=\begin{pmatrix}0\\1\\1\\0\\0\end{pmatrix}\quad \mathbf{y}^2=\begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix}\qquad \mathbf{x}^3=\begin{pmatrix}0\\0\\1\\0\\1\end{pmatrix}\quad \mathbf{y}^3=\begin{pmatrix}1\\0\\1\\0\\0\\1\end{pmatrix}$$

$$\mathbf{W}=\begin{pmatrix}1&0&0&1&1\\0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\\1&1&0&1&0\\0&0&0&0&0\end{pmatrix}\quad \mathbf{M}=\begin{pmatrix}2&2&1&2&1\\1&1&0&1&1\\2&2&1&2&2\\2&1&1&2&2\\2&1&1&2&2\\2&2&1&2&2\end{pmatrix}$$

En el mismo apéndice D se utiliza el mismo conjunto fundamental y se obtuvieron las memorias correspondientes a las asociaciones inversas $\left(\mathbf{y}^{\mu},\mathbf{x}^{\mu}\right)$; las memorias **WR** y **MR** que nos dio como resultado son:

$$\mathbf{WR}=\begin{pmatrix}0&1&0&0&0&0\\0&1&0&1&1&0\\1&2&1&1&1&1\\0&1&0&0&0&0\\1&1&0&0&0&0\end{pmatrix}\quad \mathbf{MR}=\begin{pmatrix}1&2&1&2&1&2\\2&2&1&2&1&2\\2&2&1&2&2&2\\1&2&1&2&1&2\\1&2&1&2&2&2\end{pmatrix}$$

Utilizaremos el algoritmo aquí propuesto para obtener la memoria Min de la relación inversa **WR** a partir de la memoria **M**

**Paso 1** Hacemos $\mathbf{WR}=\mathbf{M}^t$

$$\mathbf{WR} = \mathbf{M}^t = \begin{pmatrix} 2 & 1 & 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 1 & 1 & 2 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 2 & 1 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 \end{pmatrix}$$

**Paso 2** Aplicar $wr_{ij} = \varphi(wr_{ij}) \forall i, j \mid i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$

$$\mathbf{WR} = \begin{pmatrix} \varphi(2) & \varphi(1) & \varphi(2) & \varphi(2) & \varphi(2) & \varphi(2) \\ \varphi(2) & \varphi(1) & \varphi(2) & \varphi(1) & \varphi(1) & \varphi(2) \\ \varphi(1) & \varphi(0) & \varphi(1) & \varphi(1) & \varphi(1) & \varphi(1) \\ \varphi(2) & \varphi(1) & \varphi(2) & \varphi(2) & \varphi(2) & \varphi(2) \\ \varphi(1) & \varphi(1) & \varphi(2) & \varphi(2) & \varphi(2) & \varphi(2) \end{pmatrix}$$

$$\mathbf{WR} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Es fácil verificar que la memoria **WR** obtenida usando el algoritmo aquí planteado es exactamente igual a la obtenida mediante el método original; la gran vantaja es que aquí se obtuvo en 2 sencillos pasos.

Ahora, utilizaremos el algoritmo aquí propuesto para obtener la memoria Max de la relación inversa **MR** a partir de la memoria **W.**

**Paso 1** Hacemos $\mathbf{MR}=\mathbf{W}^t$

$$\mathbf{MR} = \mathbf{W}^t = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

**Paso 2** Aplicar $mr_{ij} = \varphi(mr_{ij}) \forall i, j \mid i \in \{1,2,\cdots,m\}, j \in \{1,2,\cdots,n\}$

$$\mathbf{MR} = \mathbf{W}^T = \begin{pmatrix} \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) \\ \varphi(0) & \varphi(0) & \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) \\ \varphi(0) & \varphi(0) & \varphi(1) & \varphi(0) & \varphi(0) & \varphi(0) \\ \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) \\ \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) & \varphi(0) & \varphi(0) \end{pmatrix}$$

$$\mathbf{MR} = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 2 & 2 \end{pmatrix}$$

Es fácil verificar que la memoria **MR** obtenida usando el algoritmo aquí planteado es exactamente igual a la obtenida mediante el metodo original; la gran vantaja es que aquí se obtuvo en 2 sencillos pasos.

## *4.5 El operador $\psi$*

Gracias a los teoremas de la sección 4.4 sabemos que dentro de una memoria autoasociativa Max existe la información de la memoria autoasociativa Min y viceversa. En forma similar en las memorias heteroasociativas existe la información de las memorias del tipo contrario de las asociaciones inversas. Mientras que en la sección 4.4 aprovechamos esto para generar las memorias faltantes, en esta sección se propone un nuevo operador que nos permitirá operar la misma memoria para obtener los patrones de salida correspondientes sin necesidad de calcular la(s) faltante(s).

Este nuevo operador está basado en el funcionamiento del operador $\beta$. Se usa el operador $\beta$ para obtener el vector **y**; sin embargo, para obtener el patrón **x** en esta tesis se propone el operador $\psi$, el cual se muestra a continuación

| $b$ | $a$ | $\psi(b,a)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 0 | 0 |
| 2 | 1 | 0 |

**Tabla 4.1** Operación $\psi$: **B** x **A** $\rightarrow$ **A**

Este operador se construyó de la siguiente forma: primero, sabemos que los valores de $b$ son los valores en determinada memoria, y también sabemos por los teoremas 15 y 16 que $\varphi(\beta(a,b))$ son los valores que corresponden a los valores de la relación opuesta. De esta forma, podemos crear el operador $\psi$:

| $b$ | $a$ | $\beta(b,a)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 0 | 1 |
| 2 | 1 | 1 |

| $b$ | $\delta b$ | $a$ | $\psi(b,a) = \beta(\delta b, a)$ |
|---|---|---|---|
| 0 | 2 | 0 | 1 |
| 0 | 2 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 |

### 4.5.1 Algoritmo Para Operar Las Memorias Heteroasociativas Min Y Max Como BAM

Siguiendo con lo que indican los teoremas 15 y 16, los cuales tienen que utilizar la transpuesta de la memoria en cuestión, el operador $\psi$ se utilizará sobre los valores de las columnas. Gracias a esto podemos utilizar las memorias Min y Max en un sentido como si fuera una BAM; en otras palabras, introducir un patrón de salida **y** y recuperar el patrón de entrada **x** correspondiente. Las fórmulas para lograr esto se muestran a continuación:

$$XM_j = \bigvee_{i=1}^{m} \psi(w_{ij}, y_i) \qquad XW_j = \bigwedge_{i=1}^{m} \psi(m_{ij}, y_i)$$

#### Ejemplo

Para mostrar el uso de este operador usaremos las memorias obtenidas en el ejemplo de los algoritmos originales, y para ello tenemos el siguiente conjunto fundamental y las memorias resultantes después de introducir las asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ (el proceso de aprendizaje puede ser revisado en el apéndice D.):

$$x^1 = \begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix} \quad \mathbf{y}^1 = \begin{pmatrix}1\\0\\1\\0\\1\\0\end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix}0\\1\\1\\0\\0\end{pmatrix} \quad \mathbf{y}^2 = \begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix}0\\0\\1\\0\\1\end{pmatrix} \quad \mathbf{y}^3 = \begin{pmatrix}1\\0\\1\\0\\0\\1\end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix}2&2&1&2&1\\1&1&0&1&1\\2&2&1&2&2\\2&1&1&2&2\\2&1&1&2&2\\2&2&1&2&2\end{pmatrix} \quad \mathbf{W} = \begin{pmatrix}1&0&0&1&1\\0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\\1&1&0&1&0\\0&0&0&0&0\end{pmatrix}$$

En el apéndice D también se obtuvieron las memorias correspondientes a las asociaciones inversas $(\mathbf{y}^\mu, \mathbf{x}^\mu)$, y posteriormente se operaron para recuperar el patrón **x** correspondiente. A continuación se muestra la forma de hacerlo usando las fórmulas aquí propuestas.

Empezaremos con la memoria **M**

$$\mathbf{M} = \begin{pmatrix}2&2&1&2&1\\1&1&0&1&1\\2&2&1&2&2\\2&1&1&2&2\\2&1&1&2&2\\2&2&1&2&2\end{pmatrix} \quad \mathbf{y}^1 = \begin{pmatrix}1\\0\\1\\0\\1\\0\end{pmatrix} \qquad \mathbf{y}^2 = \begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix} \qquad \mathbf{y}^3 = \begin{pmatrix}1\\0\\1\\0\\0\\1\end{pmatrix}$$

Para $\mathbf{y}^1$ tenemos:

$$\mathbf{XM} = \begin{pmatrix}\psi(2,1)\vee\psi(1,0)\vee\psi(2,1)\vee\psi(2,0)\vee\psi(2,1)\vee\psi(2,0)\\ \psi(2,1)\vee\psi(1,0)\vee\psi(2,1)\vee\psi(1,0)\vee\psi(1,1)\vee\psi(2,0)\\ \psi(1,1)\vee\psi(0,0)\vee\psi(1,1)\vee\psi(1,0)\vee\psi(1,1)\vee\psi(1,0)\\ \psi(2,1)\vee\psi(1,0)\vee\psi(2,1)\vee\psi(2,0)\vee\psi(2,1)\vee\psi(2,0)\\ \psi(1,1)\vee\psi(1,0)\vee\psi(2,1)\vee\psi(2,0)\vee\psi(2,1)\vee\psi(2,0)\end{pmatrix}$$

$$\mathbf{XM} = \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 1 & \vee & 0 \\ 1 & \vee & 1 & \vee & 1 & \vee & 0 & \vee & 1 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Para $\mathbf{y}^2$ tenemos:

$$\mathbf{XM} = \begin{pmatrix} \psi(2,0) & \vee & \psi(1,0) & \vee & \psi(2,1) & \vee & \psi(2,1) & \vee & \psi(2,1) & \vee & \psi(2,1) \\ \psi(2,0) & \vee & \psi(1,0) & \vee & \psi(2,1) & \vee & \psi(1,1) & \vee & \psi(1,1) & \vee & \psi(2,1) \\ \psi(1,0) & \vee & \psi(0,0) & \vee & \psi(1,1) & \vee & \psi(1,1) & \vee & \psi(1,1) & \vee & \psi(1,1) \\ \psi(2,0) & \vee & \psi(1,0) & \vee & \psi(2,1) & \vee & \psi(2,1) & \vee & \psi(2,1) & \vee & \psi(2,1) \\ \psi(1,0) & \vee & \psi(1,0) & \vee & \psi(2,1) & \vee & \psi(2,1) & \vee & \psi(2,1) & \vee & \psi(2,1) \end{pmatrix}$$

$$\mathbf{XM} = \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 1 & \vee & 0 & \vee & 1 & \vee & 1 & \vee & 0 \\ 0 & \vee & 1 & \vee & 1 & \vee & 1 & \vee & 1 & \vee & 1 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Para $\mathbf{y}^3$ tenemos:

$$\mathbf{XM} = \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 1 & \vee & 1 & \vee & 0 & \vee & 0 & \vee & 1 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

# CAPÍTULO 5

# Resultados y Discusión

En este capítulo se muestra una comparación entre los algoritmos originales y los algoritmos de aprendizaje y recuperación simplificados propuestos en esta tesis.

Hay que recalcar que los principales aspectos de interés registrados en la literatura sobre los modelos de memorias autoasociativas, como son: condiciones suficientes para recuperación correcta del conjunto fundamental, tipos de ruido que soportan y tipos de ruido a que son sensibles [41], no cambian en absolutamente nada dado que las memorias generadas por los algoritmos de aprendizaje simplificados son exactamente las mismas que los originales. A su vez, la capacidad de recuperación de los algoritmos simplificados es exactamente la misma dado que están basados en las definiciones y teoremas de las secciones 4.1 y 4.2

Los experimentos se realizaron en una computada personal (PC) armada, con un procesador Intel Pentium IV a 2.80 GHz, 256 MByte de RAM y 74.5 GBytes en disco duro, con el sistema operativo Microsoft Windows XP.

## *5.1 Densidad aritmética*

La reducción en los tiempos de procesamiento, ya sea para aprender o recuperar, se logra mediante la reducción del número de operaciones necesarias para realizar dichos procesos, en esta sección se realizará una comparación entre el número de operaciones necesarias entre los métodos originales y los métodos simplificados propuestos en esta tesis.

Dado que los algoritmos simplificados se basan en el número de 0s y 1s existentes en cada patrón, es necesario mostrar cómo se distribuyen los patrones con respecto al número de 0s y 1s. El número de patrones posibles con determinada cantidad de 1s o 0s está dado por el número de combinaciones de dicha cantidad con respecto a la dimensión de los patrones, esto es:

$$Nps = C^{n}_{\#01s}$$

Donde
Nps= Número de patrones
n= Dimensión del patrón
#01s= Es el número de 0s o 1s que deben de ir en el patrón

El número de 0s y el número de 1s en un patrón dado son inversamente proporcionales, esto es, que mientras el número de 0s crece, el de 1s se reduce y viceversa. Además, la distribución de los patrones con determinada cantidad de 1s y 0s es gaussiana.

Lo anterior significa que cuando pedimos una cantidad muy pequeña o muy cercana a la dimensión del patrón ya sea de 0s o 1s, el número posible de patrones será muy pequeño. La mayor cantidad de patrones posibles estará justo a la mitad; es decir, cuando pedimos el mismo número de 0s y 1s en los patrones de dimensión par o los más cercanos a la mitad de la dimensión del patrón en los patrones de dimensión impar.

Además, el número de patrones posibles con una determinada cantidad de 0s, es igual al número de patrones posibles con una cantidad de 1s igual a la dimensión del patrón menos dicho número de 0s y viceversa.

Para ilustrar esto, se da el siguiente ejemplo en cual se muestra la distribución de los patrones de dimensión 10.

| Num 0s,1s | Num patrones posibles |
|---|---|
| 0 | 1 |
| 1 | 10 |
| 2 | 45 |
| 3 | 120 |
| 4 | 210 |
| 5 | 252 |
| 6 | 210 |
| 7 | 120 |
| 8 | 45 |
| 9 | 10 |
| 10 | 1 |
| Total patrones | 1024 |

**Tabla 5.1** Distribucion de los patrones con respecto a la cnatidad de ceros y unos

Como se muestra en el ejemplo, el número de patrones posibles con *x* cantidad de ceros es igual al número de patrones posibles con 10-*x* 1s. Por ejemplo, el número de patrones con 3 ceros es igual al número de patrones con 7 unos.

Para poder hacer una buena comparación entre los algoritmos originales (que tienen operaciones $\alpha$, Min y Max) y los simplificados (que sólo usan comparaciones y asignaciones), será necesario representar las operaciones alfa, beta así como los máximos y mínimos en términos de asignaciones y comparaciones, por lo que tenemos lo siguiente:

$1\ op\ \alpha = 2$ comparaciones y 1 asignacion

$1\ op\ \beta = 2$ comparaciones y 1 asignacion

$1\ op\ \min, \max = 1$ comparacion y 1 asignacion

En la siguientes tablas se muestran el número de operaciones necesarias tanto para los algoritmos originales como para los simplificados. Solo se muestra la comparativa entre las memorias del tipo Max dado que los algoritmos par obtener las del tipo Min son prácticamente iguales y se pueden esperar resultados similares. Para ver la descripción completa de los pasos por favor refiérase a las secciones correspondientes

| **Operaciones necesarias aprendizaje Memorias Heteroasociativas tipo Max Algoritmo Original** | |
|---|---|
| **Paso** | **Operaciones Necesarias** |
| P 1. Generación de la matriz $\left[\mathbf{y}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]_{mxn}$ | $pmn\ \alpha =$ <br> $2pmn$ comparaciones + $pmn$ asignaciones |
| P 2 Aplicación del operador binario $\vee$ | $pmn$ max = <br> $pmn$ comparaciones + $pmn$ asignaciones |

**Tabla 5.2** Número de operaciones necesarias para obtener la memoria Heteroasociativa tipo Max usando el algoritmo original.

La siguiente tabla muestra el número de operaciones necesarias para el algoritmo simplificado, el número de operaciones en el paso 2 estará determinado por el número de 1s en en el patrón **y** y el número de 0s en el patrón **x**, el mejor de los casos es cuando no existen unos en **y** o ceros en ***x*,** el peor de los casos es cuando el vector **y** en todas sus posiciones vale 1 y **x** en todas sus posiciones vale 0. Aunque esta combinación es única y muy poco probable que se presente dado que provocaría una saturación total de la memoria del tipo Max. Aun asi por ser el peor de los casos se toma en cuenta. Entonces el número de operaciones necesarias para aprende $p$ patrones esta entre 0 y $pmn$.

Por otro lado el mejor de los casos se presenta cuando en **y** no existe ningun 1 o cuando en **x** no existe ningun 0. Por esta razón también se muestra el caso promedio que es cuando el número de ceros o unos del patron es la mitad de la dimensión de este.

$$0 \leq \frac{pmn}{2} \leq pmn$$

| **Operaciones necesarias aprendizaje Memorias Heteroasociativas tipo Max Algoritmo Modificado** | |
|---|---|
| **Paso** | **Op Necesarias** |
| P1. Obtención vectores $\mathbf{X0s}^{\mu}$, $\mathbf{Y1s}^{\mu}$ | $2p(m+n)$ *comparaciones* $+2p(m+n)$ *asignaciones* |
| P2 Introducir los máximos <br> $\mathbf{M}_{ij}= 2\ \ \forall i,j \in \{ (\mathbf{Y1s}^{\mu}) \text{ X } (\mathbf{X0s}^{\mu}) \}$ | $0 \leq \frac{pmn}{2} \leq pmn$ Asignaciones |
| P3 Introducir los valores faltantes a la memoria | $3mn$ comparaciones + $mn$ asignaciones |

**Tabla 5.3** Número de operaciones necesarias para obtener la memoria Heteroasociativa tipo Max usando el algoritmo simplificado.

Viendo las operaciones necesarias tanto para el algoritmo original como para el simplificado, es claro que el simplificado requiere de un menor número de operaciones,

dado que aunque el simplificado tiene un paso extra, solo el paso 2 tiene a las 3 variables como factores de una multiplicación lo cual igualaría a uno de los pasos del original, pero recordemos que este es el peor de los casos y solo existe una combinación posible, además el paso 3 es completamente independiente del número de patrones.

Por otro lado recordemos que el peor de los casos para la memoria Max es el mejor de los casos para la memoria Min. Lo cual nos daría un número de operaciones de 0 en el paso 2 de la memoria Min.

Las siguientes tablas muestran el número de operaciones necesarias para las algoritmos originales como para los simplificados para obtener las memorias autoasociativas. Recordemos que para estas memorias $\mathbf{y}^{\mu} = \mathbf{x}^{\mu} \forall \mu$. Por lo tanto, las dimensiones de los patrones son las mismas esto es m=n. Solo se muestra la comparativa entre las memorias del tipo Max dado que los algoritmos para obtener las del tipo Min son prácticamente iguales y se pueden esperar resultados similares. Para ver la descripción completa de los pasos por favor refiérase a las secciones correspondientes

| **Operaciones necesarias aprendizajeMemorias AutoAsociativas tipo Max Algoritmo Original** | |
|---|---|
| **Paso** | **Operaciones Necesarias** |
| P 1. Generación de la matriz $\left[\mathbf{y}^{\mu} \otimes \left(\mathbf{x}^{\mu}\right)^{t}\right]_{mxn}$ | $pn^2$ $\alpha$ = <br> $2pn^2$ comparaciones+ $pn^2$ asignaciones |
| P 2 Aplicación del operador binario $\vee$ | $pn^2$ max = <br> $pn^2$ comparaciones + $pn^2$ asignaciones |

**Tabla 5.4** Número de operaciones necesarias para obtener la memoria autoasociativa tipo Max usando el algoritmo original.

Analizando los algoritmos simplificados tenemos que el número de operaciones en el paso 2 estará determinado por el número de unos y ceros en el patrón **x**, el mejor de los casos es cuando no existen unos o ceros en **x** , el peor de los casos es cuando los valores de ceros y unos están distribuidos por igual en el patrón **x**. Esto es, el número de ceros es la mitad en los patrones con dimensión par o tiende a la mitad con los patrones de dimensión impar. Por lo tanto el número de operaciones es variable

$$0 \leq \frac{pn^2}{4}$$

| Operaciones necesarias aprendizaje Memorias AutoAsociativas tipo Max Algoritmo Modificado | |
|---|---|
| **Paso** | **Op Necesarias** |
| P1. Obtención vectores $\mathbf{X0s}^{\mu}$, $\mathbf{X1s}^{\mu}$ | *pn comparaciones + pn asignaciones* |
| P2 Introducir los máximos $m_{ij}= 2 \ \forall i,j \in \{ (\mathbf{X1s}^{\mu}) \ X \ (\mathbf{X0s}^{\mu}) \}$ | $0 \leq \frac{pn^2}{4}$ Asignaciones |
| P3 Después de repetirlos pasos 1 y 2 introducir los valores faltantes a la memoria | $3n^2$ comparaciones + $n^2$ asignaciones |

**Tabla 5.5** Número de operaciones necesarias para obtener la memoria autoasociativa tipo Max usando el algoritmo simplificado.

Viendo las operaciones necesarias tanto para el algoritmo original como para el simplificado, es claro que el simplificado requiere de un menor número de operaciones, dado que aunque el simplificado tiene un paso extra, solo el paso 2 tiene el número de patrones $p$ multiplicado por las $n^2$ posiciones de la memoria, sin embargo este esta dividido entre 4 además el paso 3 es completamente independiente del número de patrones.

Para las memorias autoasociativas el peor de los casos es el mismo para los 2 tipos de memoria mientras que el mejor de los casos es cuando el patron **x** no tiene 0s o 1s. Lo cual nos daría un número de operaciones de 0 en el paso 2 de la memoria Min.

A continuacion se muestra el número de operaciones para la fase de recuperacion de las memorias heteroasocitaivas, para la fase de recuperacion lo importante es el número de 1s que hay en la memoria y en el patron de entrada **x**, estos 2 valores son los que determinaran el número de operaciones a realizar. Debido a esto no importa si es autoasociativa o heteroasociativa, por lo cual solo haremos el analisis para las memorias heteroasociativas del tipo Max, ya que para las demas se puede esperar el mismo número de operaciones.

Analizando el algortimo de aprendizaje original, tenemos:

| Operaciones necesarias recuperacion Memorias Heteroasociativas tipo Max Algoritmo Original | |
|---|---|
| **Paso** | **Op Necesarias** |
| P1.Obtener $\beta(m_{ij}, x_j)$ para toda i,j | $2\,pmn$ *comparaciones* $+$ $pmn$ *asignaciones* |
| P2 Obtener los minimosde cada renglon | *pmn comparaciones + pn asignaciones* |

**Tabla 5.6** Número de operaciones necesarias para recuperar $p$ patrones de la memoria heteroasociativa tipo Max usando el algoritmo original.

Analizando el algortimo de aprendizaje original, tenemos que el número de operaciones depende si existen 0s o no en los renglones, del número de 1s en la memoria Max y del número de 1s en el patron de entrada x:

El mejor de los casos para la recuparacion es cuando la memoria esta completamente saturada, esto es que en cada posicion vale 2, puesto que en este caso no existen 0s ni 1s, y

como $\mathbf{M1s}^i$ es vacio para toda i no existe interseccion con los 1s del patron de entrada **x**. Esto provocaria que número de operaciones en el paso 3 fuera de 0.

El peor de los casos es cuando el número de unos en el patron de entrada **x** es la mitad de la dimensión de este y en la memoria solo se ha aprendido el complemento de dicho patron, esto provoca que la mitad de los valores en cada renglon tengan el valor 1 y que la interseccion entre $\mathbf{X1s}^\mu$ y $\mathbf{M1s}^i$ sea vacia para toda *i*, pero forzandonos a haccer las *pmn* comparaciones para obtener la intersección.

| **Operaciones necesarias recuperacion Memorias Heteroasociativas tipo Max Algoritmo Modificado** | |
|---|---|
| **Paso** | **Op Necesarias** |
| P1. Obtención vectores $\mathbf{X0s}^\mu$, $\mathbf{X1s}^\mu$ | $p(m+n)$ *comparaciones* + $p(m+n)$ *asignaciones* |
| P2 Revisar **EX0s** | $p$ comparaciones + $p$ asignaciones |
| P3 Obtener la interseccion entre $\mathbf{X1s}^\mu$ y $\mathbf{M1s}^i \forall$ i | $0 \leq pmn$ *comparaciones* + $pn$ *asignacion* |

**Tabla 5.7** Número de operaciones necesarias para recuperar *p* patrones de la memoria heteroasociativa tipo Max usando el algoritmo simplificado.

## *5.2 Experimentos*

En la sección anterior se presento análisis del número de operaciones necesarias para el aprendizaje y recuperación de las memorias tanto con los algoritmos originales como con los modificados. Sin embargo la forma en como se refleja esto en el tiempo se muestra en esta sección mediante gráficas.

El analizas del tiempo se logra variado los factores que intervienen en este, es decir, el número de patrones y la dimensión de los mismos.

Para los siguientes experimentos se va cambiando el número de patrones en el conjunto fundamental

### 5.2.1 Experimentos fase de aprendizaje

En esta sección se analiza el tiempo de recuperación necesario para aprender determinada cantidad de patrones con cierta dimensión. En algunos de los experimentos se varia la cantidad de patrones a aprender y en otros la dimensión de estos.

Se hace un analisis de los algoritmos de aprendizaje para las memorias autoasociativas y heteroasociativas. Tambien se hace una analisis de un caso particular de memorias heteroasociativas: clasificadores, en las cuales el patron de salida es una patron de one-hot, es decir que solo una de sus posiciones vale 1 y en todas las demas vale 0.

### 5.2.2.1 Experimentos Memorias Heteroasociativas

En las memorias heteroasociativas sólo se puede utilizar los algoritmos simplificados, ya que φ y ψ no aplican a este tipo de memoria.

Empezaremos realizando una comparación entre el tiempo necesario para aprender 1000 patrones de dimension 1000.

| Aprendizaje heteroasociativas | Original | Simplificados | Diferencia | Razón de tiempos |
|---|---|---|---|---|
| Min + Max | 91.579 | 10.078 | 81.509 | 9.08 |
| Min | 50.531 Seg | 5.093 Seg | 45.438 seg | 9.92 |
| Max | 50.875 seg | 5.000 Seg | 45.875 seg | 10.175 |

**Tabla 5.8** Comparación tiempos para las memorias Heteroasociativas con 1000 patrones de dimensión 1000

La variación del tiempo conforme se incrmenta el número de patrones se muestra a continuación:

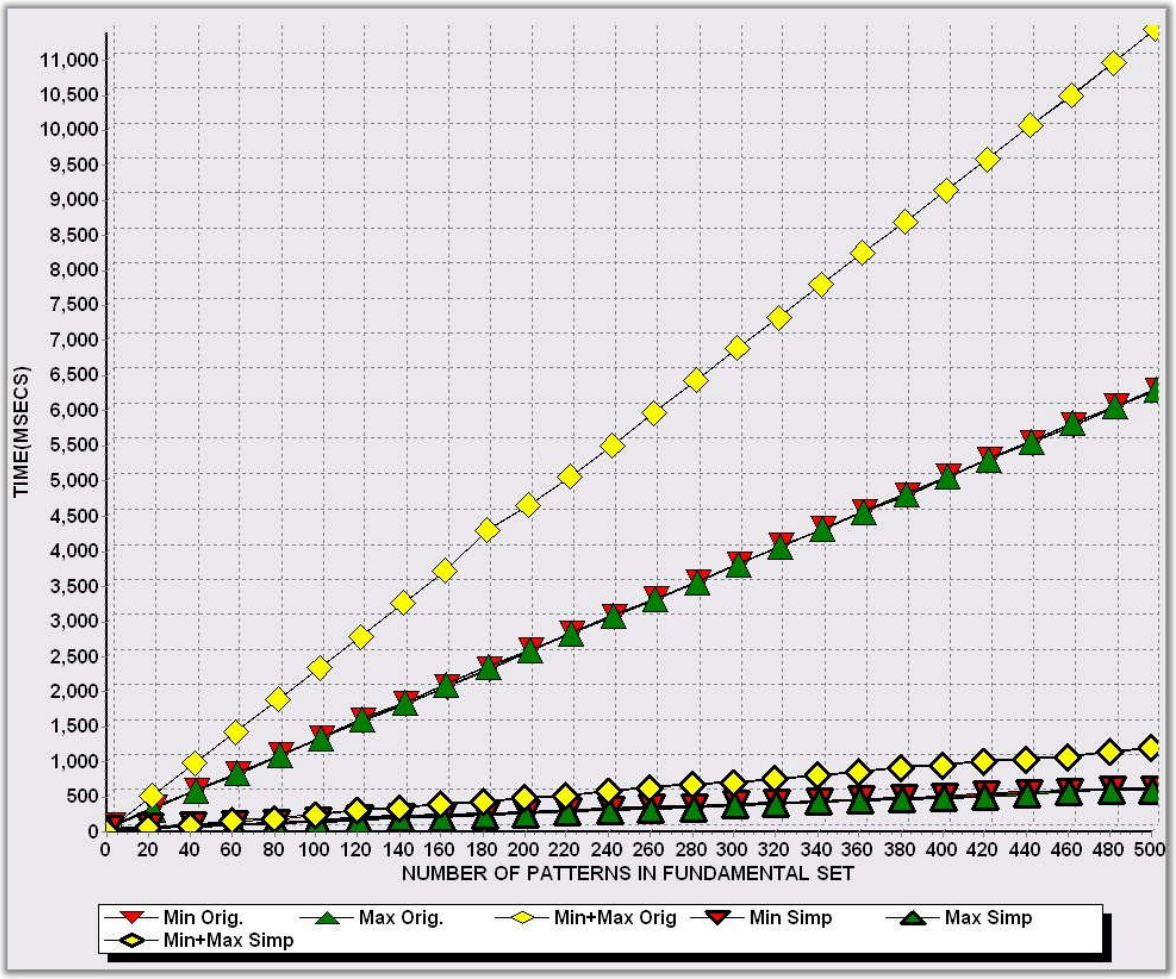



**Fig 5.1** Tiempo para aprender en una memoria heteroasociativa un número variable de patrones (0-500) con una dimensión de los patrones de entrada y salida de 500

En la gráfica anterior se muestra la comparación de tiempos variando el número de patrones, para lo cual se usaron patrones de entrada y salida con una dimensión de 500, y el número de patrones a recuperar varia de 0 a 500

Una interesante observación es que los incrementos de tiempo en cada una de los algoritmos esta dado por una línea recta, pero la recta del algoritmo propuesto tiene una menor pendiente, lo que da como resultado que la diferencia entre el tiempo necesario por cada uno de los algoritmos vaya creciendo a medida que se incrementa el número de patrones, siendo siempre mas rápido el algoritmo propuesto.

Ahora veamos que lo que pasa cuando variamos la dimensión de los patrones y el número de patrones a aprender lo dejamos estático.

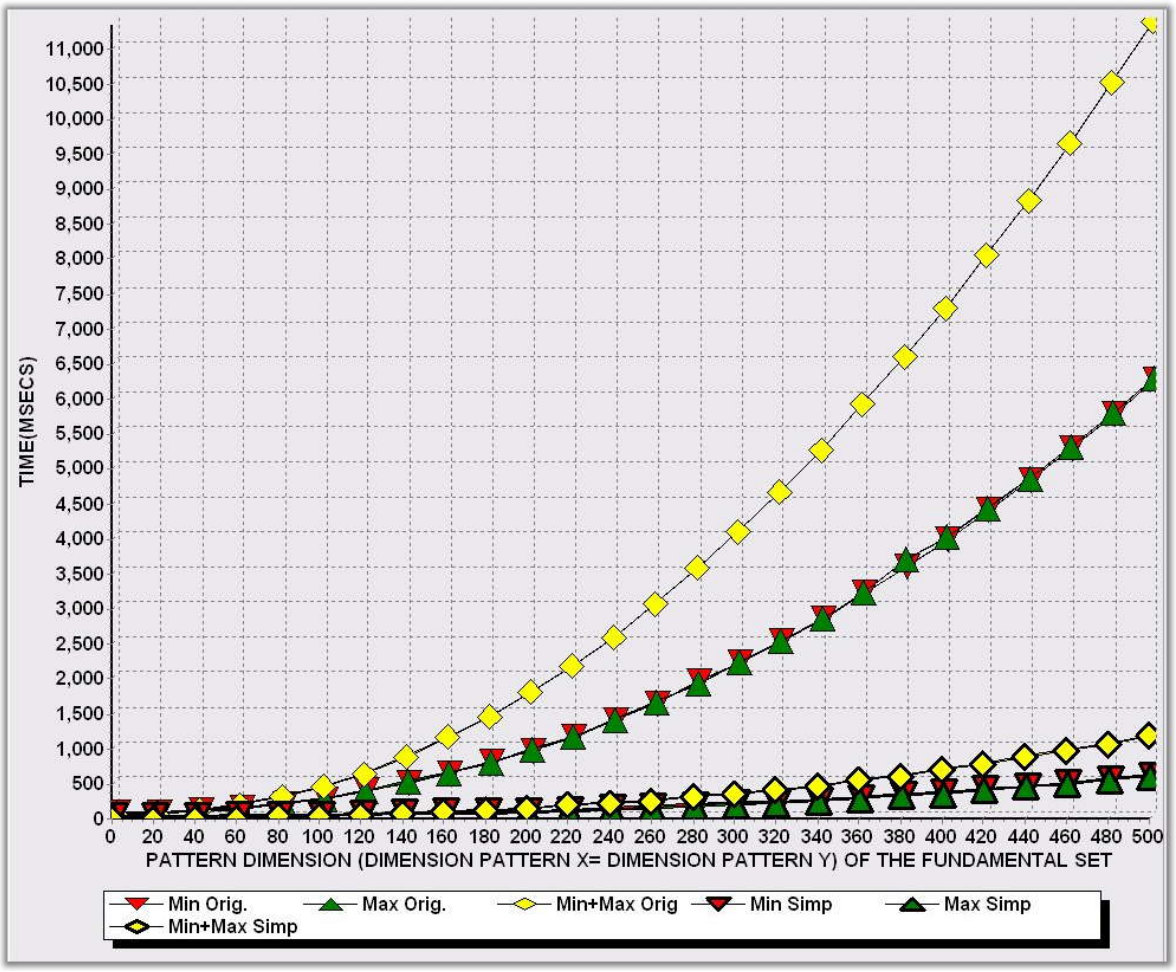



**Fig 5.2** Tiempo para aprender en memorias heteroasociativas patrones de dimension variable (0-500) con un número de patrones fijo (500)

Como se observa en la gráfica anterior el algoritmo original tiene un curva muy pronunciada que incrementa el tiempo, mientras que el algoritmo original esta curva apenas y se nota.

Al incrementar la dimensiòn de los patrones, el tiempo necesario sube. Pero aun asi, en esta gráfica notamos lo mismo que en la anterior, la curva del algoritmo original esta muy marcada, mientras que la del algoritmo propuesto apenas y se nota.

Debido a la diferencia en las curvas, podemos asegurar que el tiempo del algoritmo simplificado sera siempre menor y que la diferencia entre ambas se ira acrecentando a medida que se incremente la dimension de los patrones

### 5.2.2.2 Experimentos Memorias Heteroasociativas Con patrones de Salida One-Hot (Clasificadores)

Como ya se vio, en las memorias heteroasociativas solo se puede utilizar los algoritmos simplificados, ya que $\varphi$ y $\psi$ no aplican a este tipo de memoria. Sin embargo algo interesante ocurre con los memorias asocitivas cuando se utilizan como clasificador.

Cuando una memoria asociativa se usa como clasificador, los patrones de salida son One-Hot, esto es que unicamente tienen el valor de 1 en una sola posicion ( la clase a la que pertenecen).

Como se vio en la seccion anterior, el número de operaciones de los algoritmos simplificados se basan en el número de 0s y de 1s de los patrones de entrada y salida. Y como el patron de salida tiene una unica posición con valor de 1, entonces nos acercamos al mejor de los casos con las memorias del tipo Max y al peor de los casos en las memorias del tipo min

Empezaremos realizando una comparación entre el tiempo necesario para aprender 1000 patrones de dimension 1000.

| Aprendizaje Heteroasociativas One-Hot | Original | Simplificados | Diferencia | Razón de tiempos |
|---|---|---|---|---|
| Min + Max | 91.187 | 9.750 | 81.437 | 9.35 |
| Min | 50.250 seg | 9.703 seg | 40.547 seg | 5.178 |
| Max | 50.469 seg | .062 seg | 50.407 seg | 814.016 |

**Tabla 5.9** Comparación tiempos para las memorias Heteroasociativas con patroens de salida One Hot aprendiendo 1000 patrones de dimensión 1000

Aunque se logra reducir el tiempo hasta 814 veces para la memoria Max, para la memoria Min solo se reduce 5 veces lo cual provoca que cuando se necesitan ambas memorias el factor de reduccion quede en alrededor de 9.

Para darnos una idea de cómo se va reduciendo el tiempo conforme cambia el número de patrones o la dimension de estos.

Empezaremos mostrando la comparación variando el número de patrones, para la siguiente figura se tienen patrones de entrada y salida con una dimensión de 500, y el número de patrones a recuperar varia de 0 a 500

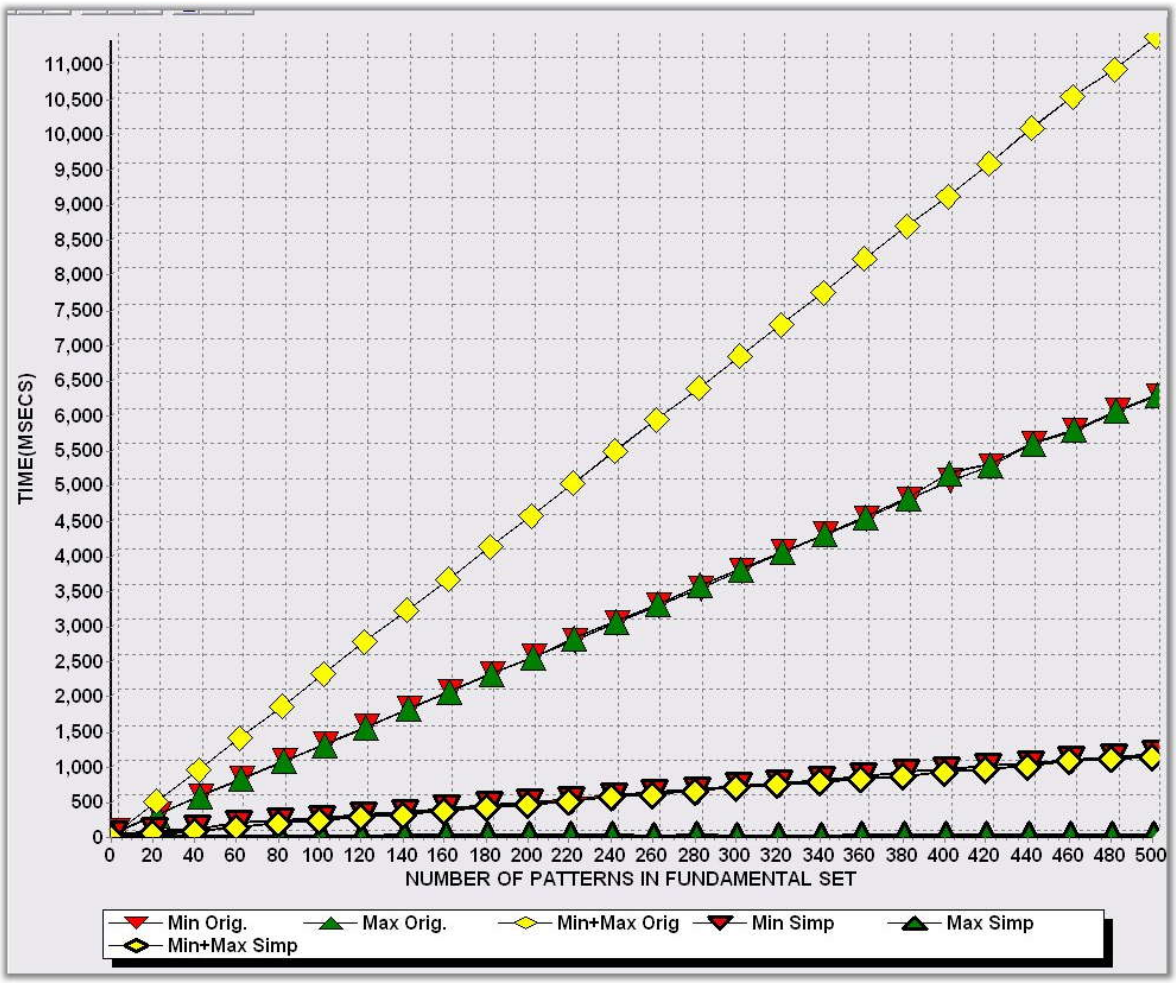



**Fig 5.3** Tiempo para aprender en una memoria heteroasociativa con patrones de salida One-Hot un número variable de patrones (0-500) con una dimensión de los patrones de entrada y salida de 500

Se observa que los incrementos de tiempo en cada una de los algoritmos esta dado por una línea recta, pero la recta de los algoritmos propuestos tiene una menor pendiente, lo que da como resultado que la diferencia entre el tiempo necesario por cada uno de los algoritmos vaya creciendo a medida que se incrementa el número de patrones, siendo siempre mas rápido el algoritmo propuesto.

Otra interesante observacion es que el tiempo de la memoria del tiempo Max se mantiene muy cercano a cero a pesar del incremento de los patrones, esto es debido a que en todas las asociaciones estamos cerca del mejor de los casos para este tipo de memoria. Sin embargo al mismo tiempo nos acercamos al peor de los casos para la memoria Min, lo cual incrementa su tiempo de cálculo. No obstante la razon de tiempos al comparar el tiempo que toma en total ambas memorias es similar a la de las memorias heteroasociativas.

Ahora veamos que lo que pasa cuando variamos la dimensión de los patrones y el número de patrones a aprender lo dejamos estático.

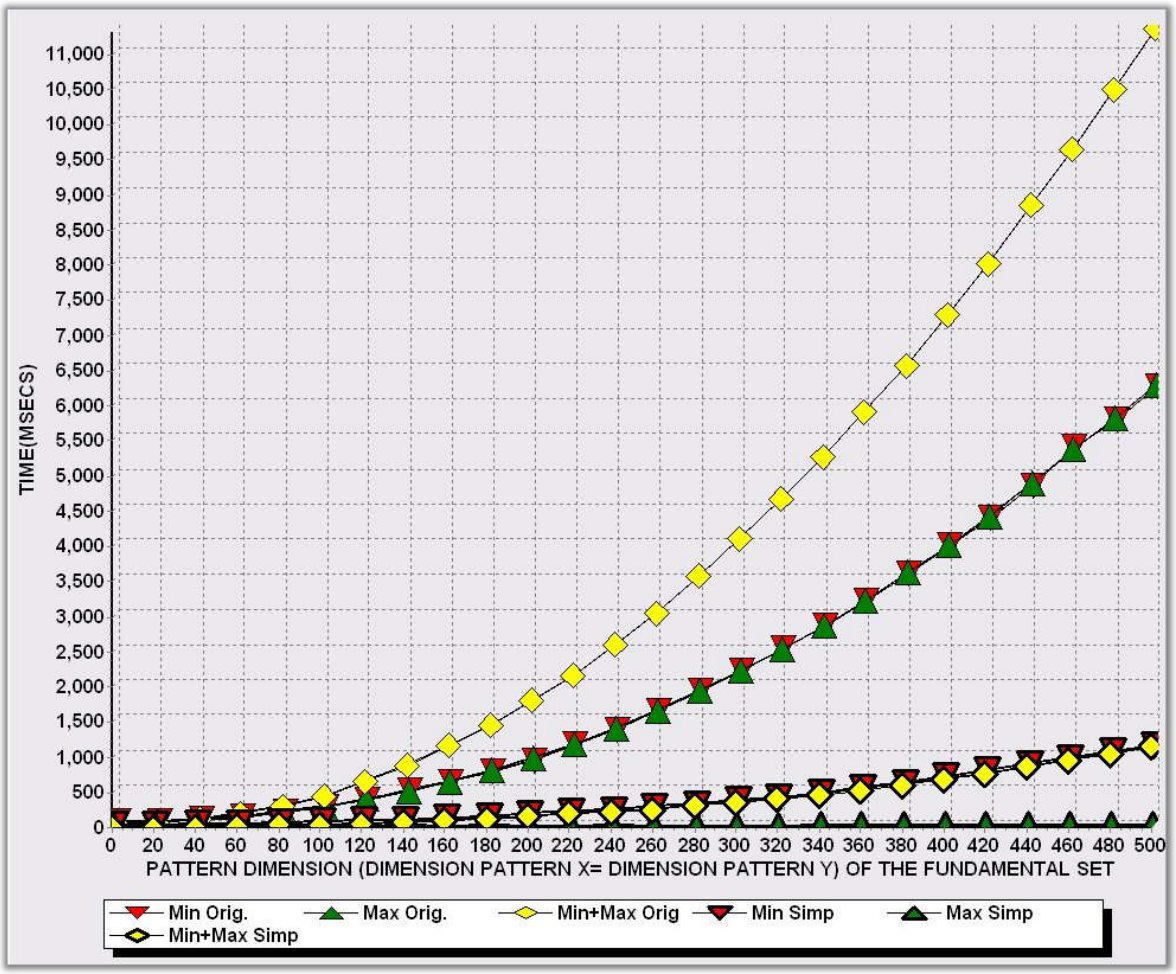



**Fig 5.4** Tiempo para aprender en memorias heteroasociativas con patrones de salida One-Hot patrones de dimension variable (0-500) con un número de patrones fijo (500)

Como se observa en la gráfica anterior el algoritmo original tiene un curva muy pronunciada que incrementa el tiempo, mientras que el algoritmo original esta curva apenas y se nota.

Al incrementar la dimensiòn de los patrones, el tiempo necesario sube. Pero aun asi, en esta gráfica notamos lo mismo que en la anterior, la curva dele algoritmo original esta muy marcada, mientras que la del algoritmo propuesto apenas y se nota.

Debido a la diferencia en las curvas, podemos asegurar que el tiempo del algoritmo simplificado será siempre menor y que la diferencia entre ambas se ira acrecentando a medida que se incremente la dimension de los patrones

### 5.2.2.3 Experimentos Memorias Autoasociativas

En las memorias autoasociativas además de los algoritmos simplificados también se pueden usar los operadores $\varphi$ y $\psi$. Lo cual nos permite reducir todavia más el tiempo de procesamiento, recordemos que el operador $\psi$ nos permite ahorrarnos completamente el cálculo de una de las memorias, para mostrar el operador $\psi$ calcualmos tambien el tiempo de la memoria Max.

Para darse una idea de cómo se reduce el tiempo se muestra el siguiente ejemplo con 1000 patrones de dimension 1000. Se muestra el tiempo para el cálculo de ambas memorias usando los distintos algoritmos mostrados en esta tesis, para comparar el operador psi se utilzara la memmoria Min para obtener la Max.

| Aprendizaje autooasociatvas | Tiempo (Seg.) | Diferencia T. Original (Seg.) | Razón de tiempos |
|---|---|---|---|
| Original | 91.172 | - | - |
| $\varphi$ | 50.656 | 40.516 | 1.79 |
| $\psi$ | 50.625 | 40.547 | 1.80 |
| Simplificados | 10.062 | 81.11 | 9.06 |
| Simplificados + $\varphi$ | 5.016 | 86.156 | 18.17 |
| Simplificados + $\psi$ | 5.000 | 86.172 | 18.23 |

**Tabla 5.10** Comparación tiempos para las memorias Autoasociativas aprendiendo 1000 patrones de dimensión 1000

Empezaremos mostrando la comparación variando el número de patrones, para la siguiente figura se tienen patrones de entrada y salida con una dimensión de 500, y el número de patrones a recuperar varia de 0 a 500

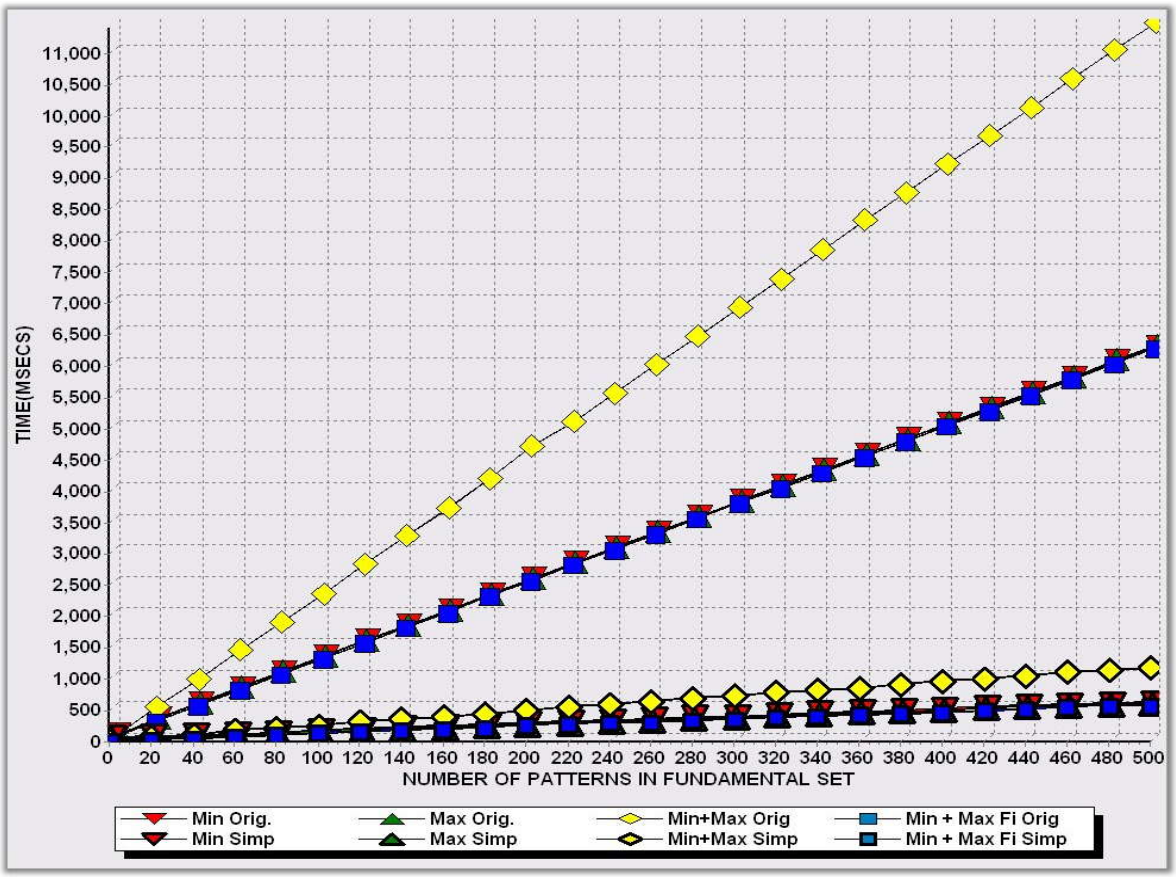



**Fig 5.5** Tiempo para aprender en una memoria autoasociativa un número variable de patrones (0-500) con una dimensión de los patrones de entrada y salida de 500

Al igual que en las memorias heteroasociativas, en las memorias autoasociativas los incrementos de tiempo en cada una de los algoritmos esta dado por una línea recta, pero la recta del algoritmo propuesto tiene una menor pendiente, lo que da como resultado que la diferencia entre el tiempo necesario por cada uno de los algoritmos vaya creciendo a medida que se incrementa el número de patrones, siendo siempre mas rápido el algoritmo propuesto.

Aunque las rectas de los algoritmos originales y simplificados sean practicamente las mismas comparadas con las de los heteroasociativas, la gran ventaja es que aquí si tenemos las posibilidad de usar los operadores $\varphi$ y $\psi$, lo cual como se puede observar reducen todavia a la mitad el tiempo de cálculo.

Ahora veamos que lo que pasa cuando variamos el la dimensión de los patrones y el número de patrones a aprender lo dejamos estático.

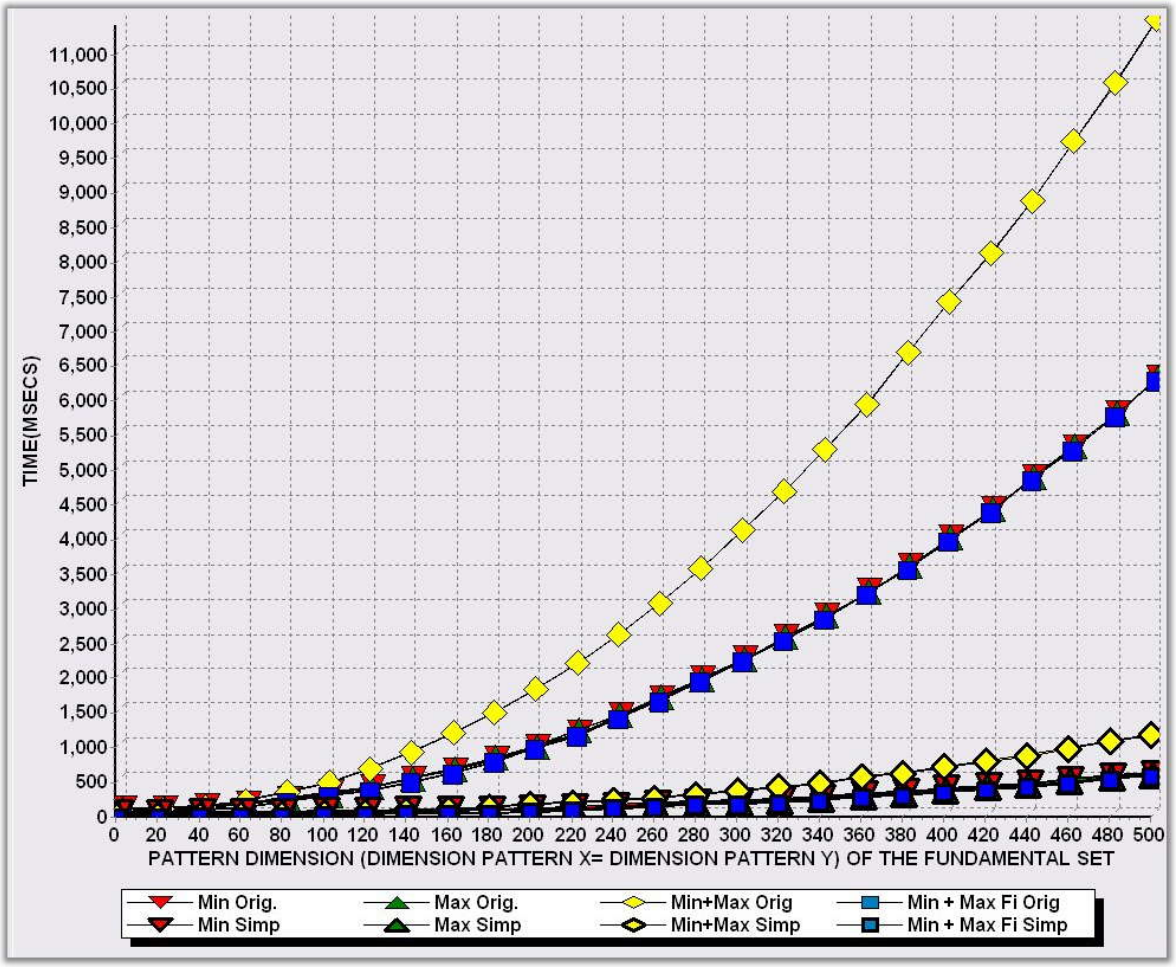



**Fig 5.6** Tiempo para aprender en una memoria autoasociativa patrones de dimension variable (0-500) con un número de patrones fijo (500)

Como se observa en la gráfica anterior el algoritmo original tiene un curva muy pronunciada que incrementa el tiempo, mientras que el algoritmo original esta curva apenas y se nota.

Debido a la diferencia en las curvas, podemos asegurar que el tiempo del algoritmo simplificado sera siempre menor y que la diferencia entre ambas se ira acrecentando a medida que se incremente la dimension de los patrones

### 5.2.2.4 Experimentos Memorias BAM Heteroasociativas

En las memorias BAM heteroasociativas sólo se puede utilizar los algoritmos simplificados, ya que φ y ψ no aplican a este tipo de memoria.

Para ilustrar la reduccion en los tiempos empezaremos mostrando una comparacion entre los diferente algoritmos vistos en esta tesis se muestra el siguiente ejemplo con 1000

patrones de dimension 1000. Se muestra el tiempo para el cálculo de ambas memorias usando los distintos algoritmos mostrados en esta tesis.

| Aprendizaje BAM | Tiempo (Seg.) | Diferencia T. Original (Seg.) | Razón de tiempos |
|---|---|---|---|
| Original | 182.234 | - | - |
| φ | 91.266 | 90.968 | 1.996 |
| ψ | 91.219 | 91.015 | 1.997 |
| Simplificados | 20.016 | 162.218 | 9.104 |
| Simplificados + φ | 10.047 | 172.187 | 18.138 |
| Simplificados + ψ | 10.016 | 172.218 | 18.194 |

**Tabla 5.11** Comparación tiempos para las memorias BAM aprendiendo 1000 patrones de dimensión 1000

Empezaremos mostrando la comparación variando el número de patrones, para la siguiente figura se tienen patrones de entrada y salida con una dimensión de 500, y el número de patrones a recuperar varia de 0 a 500

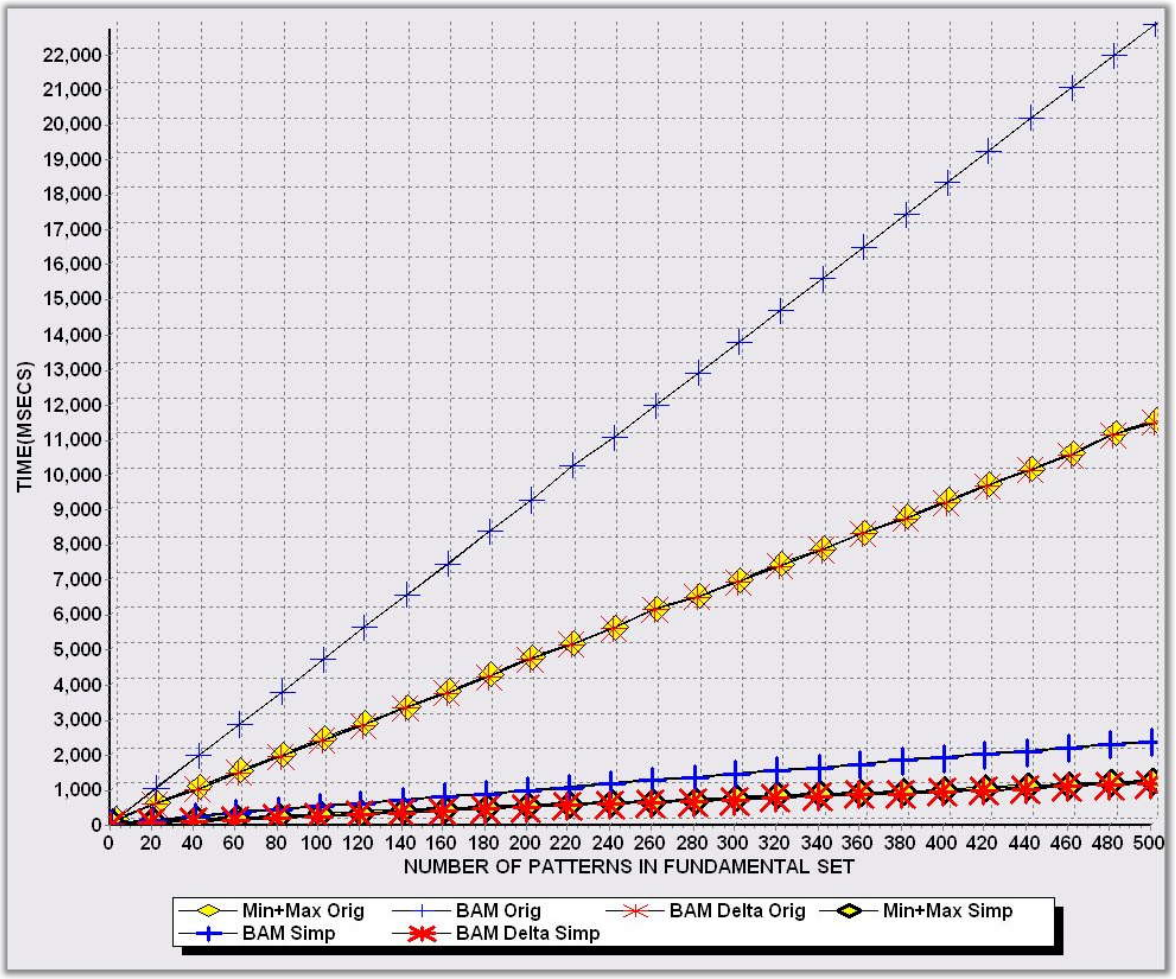



**Fig 5.7** Tiempo para aprender en una memoria BAM heteroasociativa un número variable de patrones (0-500) con una dimensión de los patrones de entrada y salida de 500

Una interesante observación es que los incrementos de tiempo en cada una de los algoritmos esta dado por una línea recta, pero la recta del algoritmo propuesto tiene una menor pendiente, lo que da como resultado que la diferencia entre el tiempo necesario por cada uno de los algoritmos vaya creciendo a medida que se incrementa el número de patrones, siendo siempre mas rápido el algoritmo propuesto.

Ahora veamos que lo que pasa cuando variamos la dimensión de los patrones y el número de patrones a aprender lo dejamos estático.

Como se observa en la gráfica anterior el algoritmo original tiene un curva muy pronunciada que incrementa el tiempo, mientras que el algoritmo original esta curva apenas y se nota.

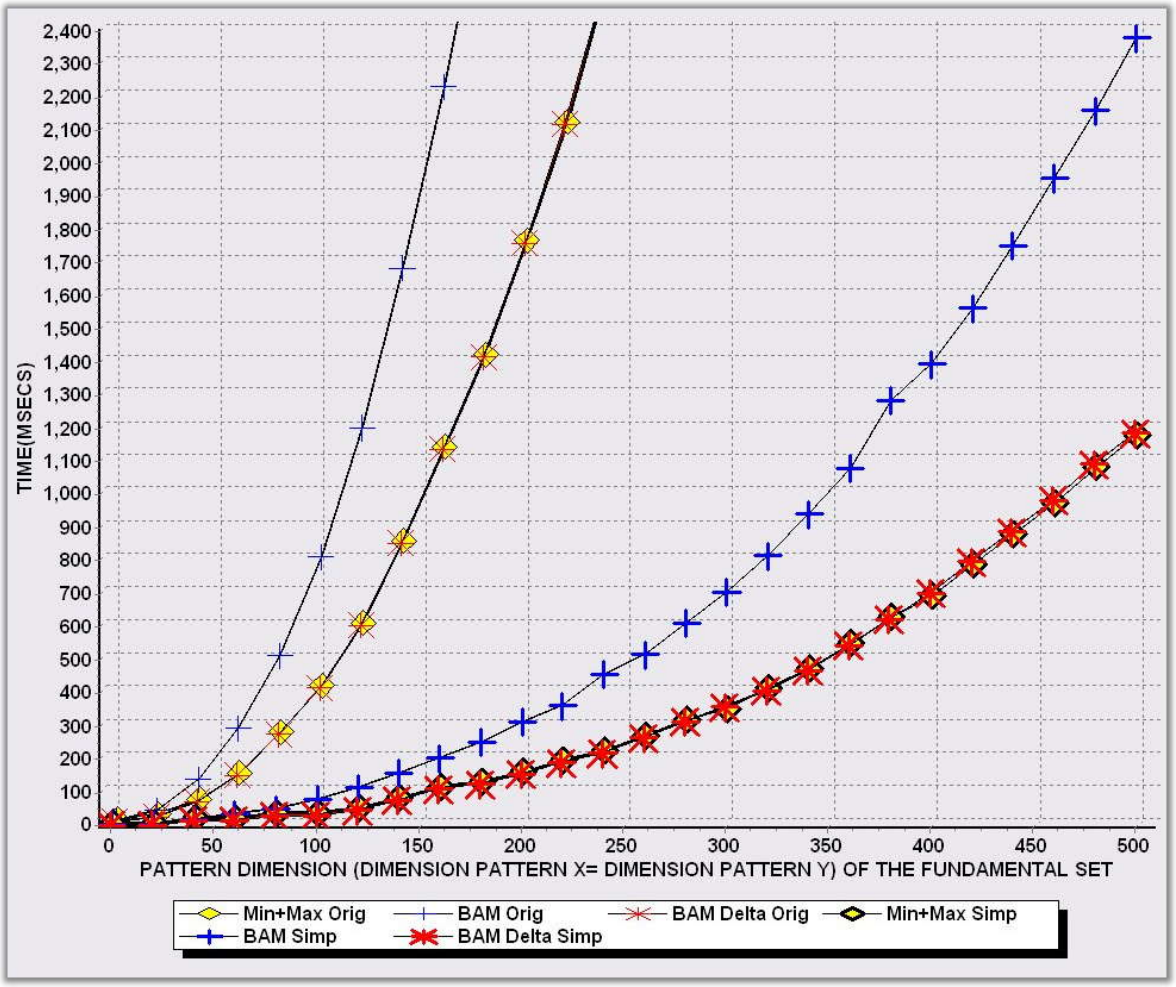



**Fig 5.8** Tiempo para aprender en memorias BAM patrones de dimension variable (0-500) con un número de patrones fijo (500)

Al incrementar la dimensiòn de los patrones, el tiempo necesario sube. Pero aun asi, en esta gráfica notamos lo mismo que en la anterior, la curva dele algoritmo original está muy marcada, mientras que la del algoritmo propuesto apenas y se nota.

Debido a la diferencia en las curvas, podemos asegurar que el tiempo del algoritmo simplificado sera siempre menor y que la diferencia entre ambas se ira acrecentando a medida que se incremente la dimension de los patrones

### 5.2.1 Experimentos fase de recuperación

En esta sección se realizan comparaciónes entre los algoritmos de recuperacion originales y simplificados

Empezamos con una gráfica que muestra el tiempo que toma recuperar una cantidad variable de patrones usando los algoritmos originales y los simplificados.

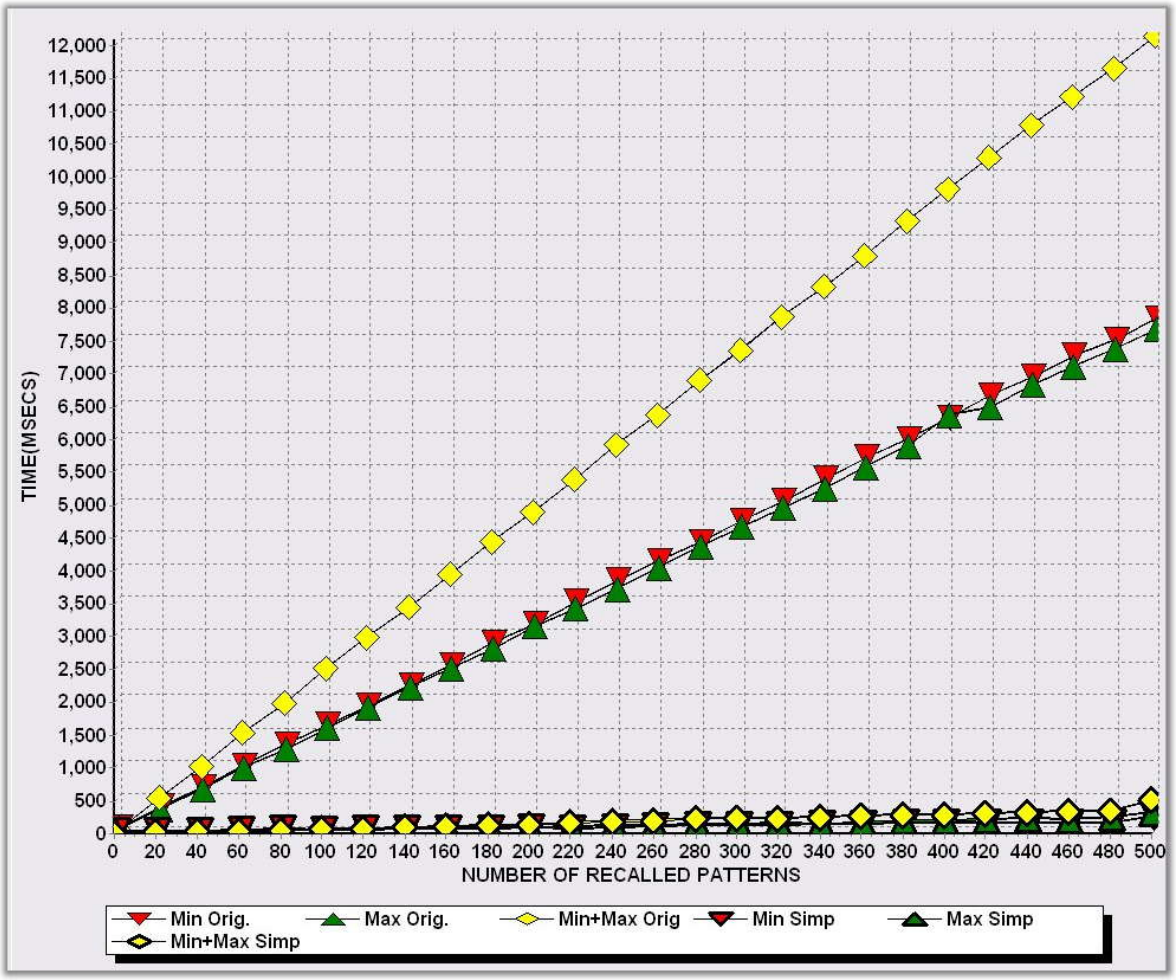



**Fig 5.9** Tiempo para recuperar un número variable de patrones (0-500) con una dimensión de 500.

Como se puede observar en la gráfica, los tiempos de recuperación estan dados por líneas rectas, la diferencia radica en que la pendiente de la recta que corresponde al algoritmo simplificado es mucho menor que la del algoritmo original, por lo tanto el tiempo

necesario para recuperar determinado número de patrones con el algoritmo simplificado sera menor que el necesario para el algoritmo original y la diferencia entre los tiempos necesarios se acrecentara mientras se incremente el número de patrones a recuperar.
Un ejemplo en el ahorro de tiempo se muestra a continuacion, para este proposito se trato de recuperar 1000 patrones de dimension 1000

| Aprendizaje heteroasociativas | Original (seg.) | Simplificados (seg.) | Diferencia (seg.) | Razón de tiempos |
|---|---|---|---|---|
| Min + Max | 11.969 | 0.391 | 11.570 | 30.61 |
| Min | 6.250 | 0.156 | 6.094 | 40.064 |
| Max | 6.187 | 0.235 | 5.952 | 26.32 |

**Tabla 5.12** Comparación tiempos para recuperar 1000 patrones de dimensión 1000 en una memoria heteroasociativa

Algo importante a hacer notar es que conforme la memoria se vaya saturando, el tiempo de recuperación va ir disminuyendo, esto se muestra en la siguiente gráfica, en la cual se recuperan patrones sobre las memorias saturadas.

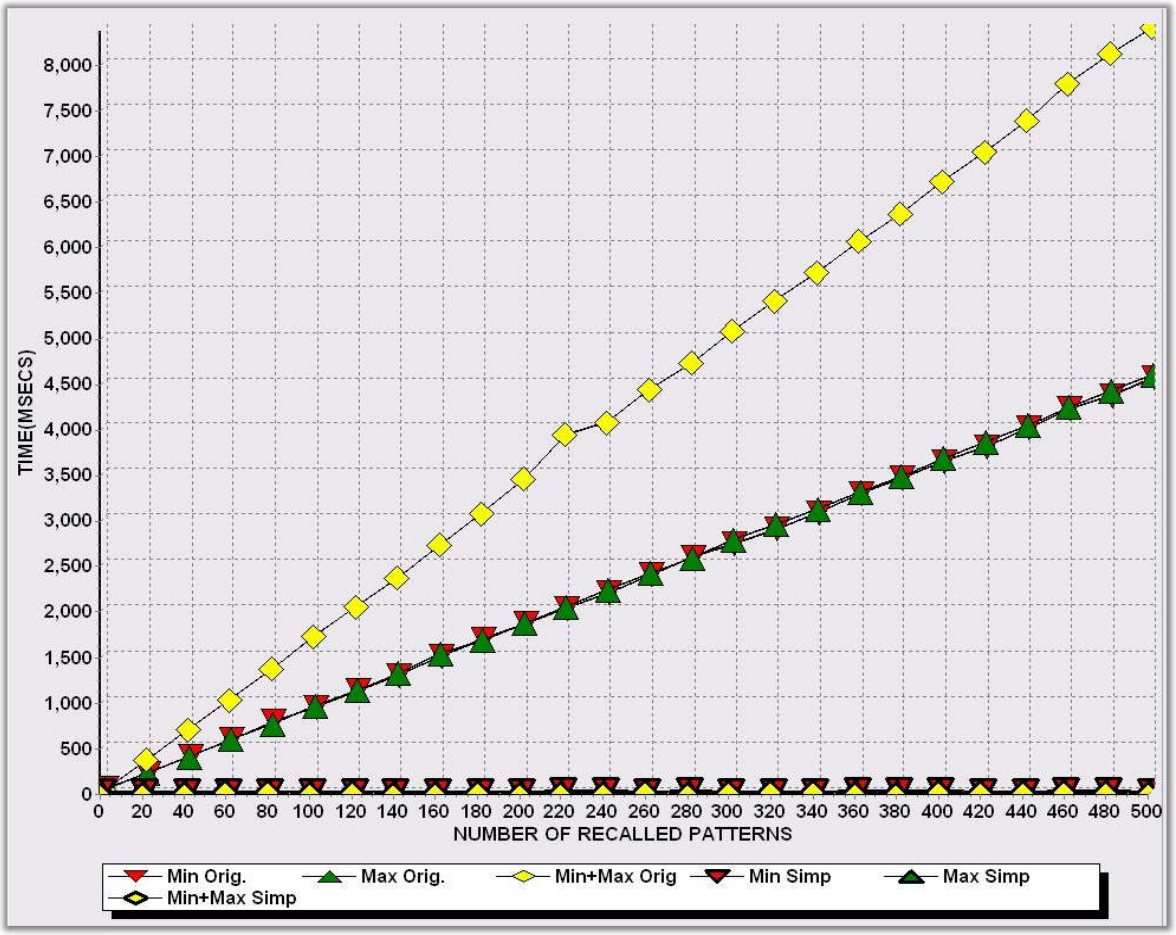



**Fig 5.10** Tiempo para recuperar un número variable de patrones (0-500) con una dimensión de 500 en una memorias heteroasociativa saturada.

Ahora haremos la comparacion de tiempos recuperando 1000 patrones de dimension 1000 en una memoria saturada

| Aprendizaje heteroasociativas | Original (seg.) | Simplificados (seg.) | Diferencia (seg.) | Razón de tiempos |
|---|---|---|---|---|
| Min + Max | 65.469 | 0.047 | 65.422 | 1392.95 |
| Min | 35.468 | 0.031 | 35.437 | 1144.12 |
| Max | 36.063 | 0.031 | 36.032 | 1163.32 |

**Tabla 5.13** Comparación tiempos para recuperar 1000 patrones de dimensión 1000 en una memoria heteroasociativa saturada.

# CAPÍTULO 6

# Conclusiones y Trabajo Futuro

En este capitulo se enuncian las conclusiones a las que se llegó durante el desarrollo de este trabajo de tesis; además, como trabajo futuro se dan ideas que complementan, extienden o aplican el trabajo presentado, las cuales pueden ser desarrolladas por otros investigadores interesados en el tema.

## *6.1 Conclusiones*

1. Para poder desarrollar los nuevos algoritmos de aprendizaje y recuperación fue necesario agregar 12 nuevas definiciones y 14 nuevos teoremas.

2. Debido a que los algoritmos de aprendizaje y recuperación están basados en teoremas construidos sobre los operadores alfa y beta, ninguna de las capacidades tales como tolerancia al ruido o la capacidad de aprendizaje se ven afectados.

3. Los algoritmos originales de aprendizaje tienen un alto número de operaciones innecesarias: siempre tienen que recorrer toda la matriz haciendo la comparación para obtener los valores correctos, los máximos en la memorias asociativas Max y los mínimos en las memorias asociativas Min.

4. Los algoritmos propuestos de aprendizaje y recuperación son más rápidos que los originales.

5. En los experimentos los algoritmos simplificados muestran una reducción de tiempos de hasta 10 veces, para los distintos tipos de memoria.

6. En esta tesis se prueba experimentalmente que los algoritmos simplificados de aprendizaje son hasta 814 veces mas rápidos que los originales, para el caso de las memorias heteroasociativas del tipo Min.

7. En la fase de recuperación, los algoritmos originales tienen que recorrer todas las posiciones de la memoria realizando la operación $\beta$ con el patrón de entrada, para obtener el patrón de salida.

8. Para recuperar patrones en los algoritmos simplificados, el número de operaciones necesarias dependerá principalmente del número de unos y de la existencia de ceros en las memorias max y del número de unos de la existencia de doses en las memorias min.

9. En esta tesis se prueba experimentalmente que los algoritmos simplificados de recuperación son hasta 1314 veces más rápidos que los originales.

10. Es posible utilizar los algoritmos simplificados de recuperación sobre memorias ya entrenadas, sin que esto les afecte en ningún modo su eficiencia.

11. El operador $\varphi$ reduce el tiempo de aprendizaje para las memorias autoasociativas y BAM a aproximadamente la mitad, que combinado con los algoritmos simplificados, permite reducir el tiempo de aprendizaje hasta 18 veces en los experiementos de esta tesis. .

12. El operador $\psi$, además de reducir el tiempo a la mitad para las memorias autoasociativas y BAM, también reduce el espacio necesario en memoria a la mitad.

## *6.2 Trabajo futuro*

1. Probar los nuevos algoritmos con bases de datos muy grandes; por ejemplo, imágenes en color[41].

2. Sustituir los algoritmos originales en las aplicaciones previamente desarrolladas, para ahorrar tiempo de proceso[41][44].

3. Diseñar un método que permita obtener directamente, de la fase de aprendizaje, los datos necesarios para aplicar los algoritmos de esta tesis en la fase de recuperación.

4. Desarrollar algoritmos que realicen tanto la fase de aprendizaje como de recuperación en forma paralela, tanto para los algoritmos originales como para los modificados.

5. Encontrar un método para convertir una memoria heteroasociativa Min en una Max y viceversa,

# Apéndice A

# Simbología

| | |
|---|---|
| $\mathbf{M}$ | Memoria asociativa o memoria asociativa Max |
| $m_{ij}$ | ij – ésima posición de la memoria M |
| $\Delta m_{ij}$ | Incremento en $m_{ij}$ |
| $\mathbf{W}$ | Memoria asociativa Min |
| $w_{\mathbf{ij}}$ | ij – ésima posición de la memoria W |
| $\mathbf{MR}$ | Memoria asociativa del tipo Max que da como resultado de introducir las asociaciones inversas $(\mathbf{y}^{\mu}, \mathbf{x}^{\mu})$ |
| $mr_{\mathrm{ij}}$ | Ij-ésima poscion de la memoria **MR** |
| $\mathbf{WR}$ | Memoria asociativa del tipo Min que da como resultado de introducir las asociaciones inversas $(\mathbf{y}^{\mu}, \mathbf{x}^{\mu})$ |
| $wr_{ij}$ | ij-ésima posición de la memoria **WR** |
| $\mathbf{x}$ | vector columna que corresponde a un patrón de entrada |
| $x_{\mathbf{i}}$ | Iésima posición del patron X |
| $\mathbf{y}$ | vector columna que corresponde a un patrón de salida |
| $y_{\mathrm{i}}$ | Iésima posición del patron Y |
| $\mathbf{YM}$ | Vector de salida que da una memoria asocitiva Max |
| $YM_{\mathrm{i}}$ | i-ésima posición del vector **YM** |
| $\mathbf{YW}$ | Vector de salida que da una memoria asocitiva Min |
| $YW_{\mathrm{i}}$ | i-ésima posición del vector **YW** |
| $(\mathbf{x}, \mathbf{y})$ | asociación de un patrón de entrada con uno de salida |
| $(\mathrm{x}^k, \mathrm{y}^k)$ | asociación de la $k$-ésima pareja de patrones |
| $\{(\mathrm{x}^{\mu}, \mathrm{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$ | conjunto fundamental |
| $\mathbf{A}$ | Conjunto al que pertenecen los vectores **x** y **y.** $\mathbf{A} = \{0,1\}$ |
| $\mathbf{B}$ | Conjunto de 3 elementos $\mathbf{B} = \{0,1,2\}$ |
| $p$ | número de parejas del conjunto fundamental |
| $n$ | dimensión de los patrones de entrada |
| $m$ | dimensión de los patrones de salida |
| $\forall$ | cuantificador universal |
| $\in$ | pertenencia de un elemento a un conjunto |
| $\exists$ | cuantificador existencial |
| $\times$ | producto cruz (entre conjuntos) |
| $\cdot$ | producto usual entre vectores o matrices |
| $\vee$ | operador máximo |
| $(\mathbf{x}^{\mu})^t$ | Transpuesto del vector $\mathbf{x}^{\mu}$ |
| $\mathbf{I}$ | matriz identidad |
| $\tilde{\mathbf{x}}$ | versión alterada del patrón fundamental **x** |
| $\nabla$ | producto máximo |
| $\Delta$ | producto mínimo |
| $\wedge$ | operador mínimo |

| | |
|---|---|
| $\alpha$ | Operador Alfa, operador base de la fase de aprendizaje de las memorias asociativas alfa beta |
| $\beta$ | operador beta, operador base de la fase de aprendizaje de las memorias asociativas Alfa-Beta |
| $\alpha(x, y)$ | operación binaria $\alpha$ con argumentos $x$ y $y$ |
| $\beta(x, y)$ | operación binaria $\beta$ con argumentos $x$ y $y$ |
| **med** | operador media |
| $\nabla_\alpha$ | operador $\alpha$ *max* |
| $\nabla_\beta$ | operador $\beta$ *max* |
| $\Delta_\alpha$ | operador $\alpha$ m*in* |
| $\Delta_\beta$ | operador $\beta$ m*in* |
| $\otimes$ | símbolo que representa a las dos operaciones $\nabla_\alpha$ y $\Delta_\alpha$ |
| $\mathbf{V}$ | memorias asociativas Alfa-Beta tipo *Max* |
| $\mathbf{\Lambda}$ | memorias asociativas Alfa-Beta tipo *Min* |
| $\mathbf{Z}^+$ | conjunto de los números enteros positivos |
| **X0s** | Vector que contiene las posiciones de los ceros en el patron **x** |
| $X0s_i$ | Iésima posición del vector **X0s** |
| **X1s** | Vector que contiene las posiciones de los unos en el patron **x** |
| $X1s_i$ | Iésima posición del vector **X1s** |
| **EX0s** | Vector de existencia de 0s en el patron **x** |
| $EX0s_i$ | Iésima posición de **EX0s**, indica si para algún patrón del conjunto fundamental esa posición ha tenido el valor de 0. |
| **EX1s** | Vector de existencia de unos en el patron **x** |
| $EX1s_i$ | Iésima posición de **EX1s**, indica si para algún patrón **x** del conjunto fundamental esa posición ha tenido el valor de 0. |
| **Y0s** | Vector que contiene las posiciones de los 0s en el patron **y** |
| $Y0s_i$ | Iésima posición del vector **Y0s** |
| **Y1s** | Vector que contiene las posiciones de los unos en el patron **y** |
| $Y1s_i$ | Iésima posición del vector **Y1s** |
| **EY0s** | Vector de existencia de ceros en el patron **y** |
| $EY0s_i$ | Iésima posición de **EX0s**, indica si para algún patrón **y** del conjunto fundamental la posición $EY0s_i$ ha tenido el valor de 0. |
| **EY1s** | Vector de existencia de unos en el patron **y** |
| $EY1s_i$ | Iésima posición de **EX1s**, indica si para algún **y** patrón del conjunto fundamental esa posición ha tenido el valor de 0. |
| **EM0s** | Vector que indica la existencia de $m_{ij}=0$ en la memoria **M** |
| $EM0s_i$ | Iésima posición de le vector EM0s, en donde i es el renglón de la memoria **M** |
| **EW2s** | Vector que indica la existencia de $w_{ij}=2$ en la memoria **W** |
| $EW2s_i$ | Iésima posición de le vector EW2s, en donde i es el renglón de la memoria **W** |
| $\mathbf{M1s}^i$ | Vector de posiciónes de unos en el renglón i de la memoria **M** |
| $\mathrm{W1s}^i$ | Vector de posiciónes de unos en el renglón i de la memoria **W** |
| $\varphi$ | Operador fi |
| $\varphi$(a) | Operador fi con argumento a |
| $\psi$ | Operador psi |

| | |
|---|---|
| $\psi$ (a,b) | Operador psi con argumentos (a,b) |

# Apéndice B

# Conceptos Básicos

El propósito de este apéndice es el de introducir los conceptos básicos que se usan en las memorias asociativas, para facilitar la comprension, estos conceptos se introducen a través de un ejemplo

**Características**.- Es un elemento o parte especial, distintiva de un ser u objeto[50] y sirven para identificar a un ser de sus semejantes[49]. Por ejemplo: Algunas de las características que describen a un perro, es que es peludo, baboso, leal, tiene 4 patas y ladra. Las características que definen a una gallina son: tiene 2 patas, tiene alas, tiene plumas, pone huevos y cacarea. Las características que definen a un gato son: es que es peludo, tiene 4 patas, orejas puntiagudas,y maúlla.

Una vez especificado lo anterior, las características pueden ser usadas para 2 cosas:

*1.- Identificación o clasificación*.- Podemos usar un conjunto (o subconjunto) de características para identificar a qué se refieren dichas características. Siguiendo con el ejemplo de los animales, podemos hacer las siguientes preguntas:
¿Qué animal tiene patas? todos tienen patas
¿Qué animal tiene 4 patas? Perro y gato
¿Qué animal tiene 4 patas y ladra? Perro
¿Qué animal tiene 4 patas y no es baboso? Gato
¿Qué animal pone Huevos? Gallina
¿Qué animal es peludo y no maúlla? Perro

*2.- Detección de combinación de características incorrectas*.- Pueden llegar a existir combinaciones de características que son incorrectas, pero que gracias a otra característica ser capaz de decidir a qué se refiere y corregir el error en la combinación; por ejemplo:

¿Qué animal es peludo, ladra y cacarea?, el perro es el único que es peludo y ladra pero no cacarea.

¿Qué animal pone huevos, es peludo y cacarea?, la gallina pone huevos y cacarea pero no es peluda.

Al ser datos tomados del mundo real, una característica puede ser de cualquier tipo, es decir, puede ser numérica o no y en caso de serlo puede tener un valor real, entero, complejo, binario,o de cualquier otro tipo

Sin embargo cuando se trabaja con modelos computacionales, tales como las redes neuronales y las memorias asociativas, se debe escoger un conjunto de características mediante las cuales sea posible describir todos los objetos de interés(aquellos que deseamos que se aprendan) y que el modelo sea capaz de manejar.

La mayoría de los modelos son binarios, por lo que el tipo de característica que pueden aceptar son aquellas que tiene valor de falso o verdadero, representados por un 0 para falso y un 1 para verdadero.

Continuando con nuestro ejemplo de perros gallinas y gatos, las características podrían ser representadas de distintas formas; por ejemplo, podríamos tener la característica "número de patas", la cual podría tener los valores enteros de 2 y 4; o podríamos separarlo en dos características binarias: "anda en 2 patas" y "anda en 4 patas", las cuales tendrán el valor lógico de 1 para los casos en que se cumpla y el valor lógico de 0 para los que no. De forma similar también podríamos tener la característica "Sonido que emite", en vez de por separado las características binarias "ladra", "maúlla" y "cacarea".

**Patrón**.- Es un conjunto de características codificadas en cierto orden dentro de un vector y con un valor asignado. Se define como un vector de dimensión n, de la forma $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$ en donde $x_i$ representa una característica y $n$ es el número de características[49]; el valor que $x_i$ puede llegar a tener está restringido por el tipo de característica. Siguiendo con nuestro ejemplo, se pueden codificar las características de manera binaria de la siguiente forma:

| Pelo | Plumas | 2 Patas | 4 Patas | Ladra | Cacarea | Maulla | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | Perro |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | Gallina |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | Gato |

**Tabla B.1** Codificacion de los animales perro, gato y gallina como patrones binarios

Nótese que una vez que se describió un conjunto de características, todas las características deben de ser utilizadas para representar como patrón a cualquiera de los animales, además de que el orden debe ser respetado. Entonces la descripción de los animales es como sigue:

El perro está representado por el patrón "1001100", es decir, es el animal que es peludo, no tiene plumas, no anda en 2 patas, anda en 4 patas, ladra y no cacarea ni maúlla.

La gallina esta representada por el patrón "0110010", es decir es el animal que no es peludo, tiene plumas, anda en 2 patas, no anda en 4 patas, no ladra, cacarea y no maúlla.

El gato esta representado por el patrón "1001001", es decir, es el animal que es peludo, no tiene plumas, no anda en 2 patas, anda en 4 patas, no ladra, no cacarea y maúlla.

**Dimensión de un Patrón.-** Es el número de posiciones que tiene un vector **x**. Se representa por | **x** |. Por ejemplo:

| [-1,1,-1,1] |= 4 Patrón bipolar de dimensión 4
| [1,1,1,0,1,1,1] |=7 Patrón binario de dimensión 7
| [2,7,1]|=3 Patrón de enteros positivos de dimensión 3
| [0,1,0,0] |= 4 Patrón binario de dimensión 4

**Distancia de Hamming.-** La distancia de Hamming entre 2 patrones binarios, está dada por el número de posiciones en las cuales son diferentes [50], por ejemplo:
[1,0,1,1,1,0] y [0,1,1,1,1,0] tienen una distancia de Hamming de 2
[1,0,1,1,1,0] y [0,1,0,0,0,1] tienen una distancia de Hamming de 6

Por consiguiente la distancia de Hamming entre un patrón a sí mismo es 0 y a su complemento es la dimensión del mismo, nos sirve para saber qué tan similar es un patrón de otro. Así, entre mas cercana a 0 más similares son y entre más se acerque al valor de la dimensión más distante será.

**Ruido**.- Se conoce como ruido a las variaciones en el valor de las características de un patrón de referencia.

Si a un patrón se le cambia el valor de una característica mediante la adición de una valor, es conocido como ruido aditivo, si por el contrario es producido por una resta, entonces es conocido como ruido sustractivo. Cuando existe una combinación entre ruido sustractivo y aditivo, se le llama ruido mezclado.

En patrones binarios, el ruido aditivo se da cuando una característica que tenia una valor de falso se convierte en verdadera (Un 0 se convierte en 1), y el ruido sustractivo es cuando pasa lo contrario, es decir cuando una característica que tenia el valor de verdadero se convierte en falsa (un 1 se convierte en 0).

En patrones binarios la cantidad de ruido total es igual a la distancia de Hamming entre 2 patrones. Siguiendo con nuestro ejemplo:

| PATRONES RUIDOSOS | | | | | | |
|---|---|---|---|---|---|---|
| Pelo | Plumas | 2 Patas | 4 Patas | Ladra | Cacarea | Maulla |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Tabla B.2** Ejemplo de patrones alterados

Vamos a analizar el primer caso tenemos “1101110”, vamos a ver que tipo de ruido tiene comparado con los patrones que tenemos definidos

Comparado con perro “1001100”, tiene ruido aditivo en las posiciones 2 y 6. Lo cual nos da una distancia de Hammming de 2

Comparado con gallina ““0110010”, tiene ruido aditivo en las posiciones 1, 4 y 5, y ruido sustractivo en 3. Lo cual nos da una distancia de Hamming de 4.

Comparado con gato “1001001”, tiene ruido aditivo en las posiciones 2,5,6 y ruido sustractivo en 7. Lo cual nos da una distancia de Hamming de 4.

**Conjunto fundamental.-** En la vida real, cuando se nos da una conjunto de características, nuestra mente, con base en el conocimiento previo, automáticamente hace 2 cosas: asignar esas características a una clase o discernir si el conjunto de características es coherente, entendiendo por conjunto coherente, aquel en el que se garantiza que para cada patrón de entrada, existe uno y sólo uno de salida; es decir, que no hay ambigüedad.

Cuando se trabaja con modelos computacionales tales como redes neuronales o memorias asociativas, es necesaria una forma de representar ese conocimiento previo. Esto se hace mediante el establecimiento de relaciones de interés entre patrones; entonces, el conjunto fundamental es el conjunto que contiene todas estas relaciones.

En un conjunto fundamental se dice que hay patrones de entrada, que son los que se alimenta al modelo y patrones de salida que son los que da como resultado. Estas relaciones se pueden representar de la forma (**x**, **y**), en donde **x** es el patrón de entrada y **y** es el patrón de salida. En un conjunto fundamental existe un número finito de relaciones, las cuales vamos a denotar con el número entero no negativo $p$.

Dado que podemos tener $p$ asociaciones diferentes, la forma de diferenciarlas es mediante un superíndice, el cual nos indicará el número de asociación. Llamaremos a este superíndice $\mu$. Así, el conjunto fundamental queda representado como $(\mathbf{x}^{\mu},\mathbf{y}^{\mu})$, donde $\mu = 1,2,3\ldots p$.

Siguiendo con nuestro ejemplo de animales, los patrones de entrada serían los que hemos manejado hasta el momento; sin embargo, necesitamos definir los patrones de salida, y una forma sería codificar a perro, gallina y gato como cadenas binarias de la siguiente forma:

| PATRONES DE ENTRADA | | | | | | | PATRONES DE SALIDA | | |
|---|---|---|---|---|---|---|---|---|---|
| Pelo | Plumas | 2Patas | 4Patas | Ladra | Cacarea | Maúlla | Perro | Gallina | Gato |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

**Tabla B.3** Asociación de los patromes perro, gato y gallina con sus respectivas clases.

Por lo anterior, el conjunto fundamental queda así:

$(\mathbf{x}^1,\mathbf{y}^1) = ([1,0,0,1,1,0,0], [1,0,0])$
$(\mathbf{x}^2,\mathbf{y}^2) = ([0,1,1,0,0,1,0], [0,1,0])$
$(\mathbf{x}^3,\mathbf{y}^3) = ([1,0,0,1,0,0,1], [0,0,1])$

En este caso estamos asignando cada patrón de entrada a un patrón de salida que representa una clase, y decimos de esta forma que estamos clasificando los patrones de entrada. Para nuestro ejemplo sólo estamos asignando un patrón de entrada a cada clase pero en realidad podrían ser más.

Algo más que podemos agregar es que el patrón de salida puede tener tantas características como se desee o necesiten.

Por último, podríamos llegar a relacionar el vector de entrada consigo mismo y de esta forma tener una relación ($\mathbf{x}^{\mu}$, $\mathbf{x}^{\mu}$); a esto se le llama autoasociación y tiene características especiales cuando se cumple para todas las relaciones del conjunto fundamental, en cuyo caso decimos que el conjunto fundamental es auto asociativo. Su principal utilidad es la de discernir si un conjunto de características dado es coherente o no, y en caso de que no lo sea, identificar cuáles características son las incorrectas.

Si se cumple que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$ para algún $\mu$ decimos que el conjunto de entrenamiento es heteroasociativo, y por ende la memoria también es heteroasociativa.

Por lo anterior, podemos decir que una memoria auto asociativa pude considerarse como un caso particular de una memoria heteroasociativa [19].

**Selección de características**.- Una buena solución a un problema dado comienza desde la selección de las características que se utilizarán para resolverlo; de nada nos sirve tener características que no aporten información relevante y que sólo consuman tiempo de cómputo. El problema entonces es qué tan relevante es una característica y si se puede manejar un subconjunto de las características dadas para obtener resultados similares en la recuperación de los patrones, ya sean alterados o no.

El hecho de reducir características nos beneficia puesto que el número de operaciones necesarias disminuye tanto en el aprendizaje como en la recuperación. Si se puede o no reducir el número de características depende de las relaciones que tenemos en el conjunto fundamental. Es posible reducir el número de características necesarias sin que disminuya el porcentaje de efectividad, o inclusive incrementándolo.

Aunque esto parezca sencillo, en realidad es un asunto complicado. Si bien es verdad que existen características que son altamente relevantes (aquellas que son diferentes con respecto a los otros patrones), las cuales deben ser conservadas, y otras que prácticamente son irrelevantes (que son iguales para la mayoría de los patrones o que existe otra característica con valores similares) y que fácilmente podrían ser eliminadas, en la práctica la eliminación de una característica o conjunto de ellas podría llegar a incrementar la sensibilidad ante cierto tipo de alteración.

Básicamente, el problema de la selección de características es encontrar el conjunto mínimo de características con el cual la recuperación del conjunto fundamental es perfecta y la robustez de la memoria frente al ruido es alta.

Siguiendo con el ejemplo de animales, tenemos características que son completamente diferentes entre sí y que de por sí solas nos sirven para clasificar, como es el caso de “ladra”, “cacarea” y “maúlla”; incluso para este caso en particular estas características son ortogonales, lo cual significa que solas serían capaces de realizar la clasificación. Sin embargo, si sólo existieran esas, la mas mínima presencia de cualquier tipo de ruido

impediría una clasificación adecuada. Estas características deben de ser preservadas puesto que son las que aportan mayor información.

Por el otro lado, también observamos que los valores de las características “pelo” y “4 patas” son iguales para todos los patrones al igual que “Plumas” y “2 patas”, lo cual nos lleva a pensar que probablemente podríamos eliminar una de las iguales sin que haya mucha variación en la tolerancia al ruido.

Un método interesante que nos podría ayudar a comprender el comportamiento de un modelo determinado al quitar ciertas características es uno basado en el enmascaramiento de las características [47], ya que explora todas las posibles reducciones que se podrían llegar a hacer y determina cuál es la mejor.

# Apéndice C

# Estado del arte de las memorias asociativas

A continuación, en este apéndice haremos un breve recorrido por los modelos de memorias asociativas más representativos, los cuales sirvieron de base para la creación de modelos matemáticos que sustentan el diseño y operación de memorias asociativas más complejas. Para cada modelo se describe su fase de aprendizaje y su fase de recuperación.

Se incluyen cinco modelos clásicos basados en el anillo de los números racionales con las operaciones de multiplicación y adición: *Lernmatrix*, *Correlograph*, *Linear Associator* , Memoria Hopfield y su secuela, la BAM de Kosko; además, se presentan tres modelos basados en paradigmas diferentes a la suma de productos, a saber: memorias asociativas Morfológicas, memorias asociativas Alfa-Beta y memorias asociativas Mediana.

## *Lernmatrix* de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar* [20]. La importancia de la *Lernmatrix* [10, 29] se evidencia en una afirmación que hace Kohonen [13] en su artículo de 1972, donde apunta que las matrices de correlación, base fundamental de su innovador trabajo, vinieron a sustituir a la *Lernmatrix* de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^{\mu} \in A^n, \quad A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^{\mu} \in A^p$ que le corresponde (de entre $p$ clases diferentes), codificada ésta con un método que en la literatura se le ha llamado *one-hot* [30].

La codificación *one-hot* funciona así: para representar la clase $k \in \{1, 2, ..., p\}$, se asignan a las componentes del vector de salida $\mathbf{y}^{\mu}$ los siguientes valores: $y_k^{\mu} = 1$, y $y_j^{\mu} = 0$ para $j = 1, 2, ..., k\text{-}1, k\text{+}1, ..., p$.

### Algoritmo de la *Lernmatrix*

#### Fase de Aprendizaje

Se genera el esquema (*crossbar*) al incorporar la pareja de patrones de entrenamiento $\left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}\right) \in A^n \times A^p$. Cada uno de los componentes $m_{ij}$ de $\mathbf{M}$, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } x_j^{\mu} = 1 = y_i^{\mu} \\ -\varepsilon & \text{si } x_j^{\mu} = 0 \text{ y } y_i^{\mu} = 1 \\ 0 & \text{en otro caso} \end{cases}$$

donde $\varepsilon$ una constante positiva escogida previamente: es usual que $\varepsilon$ es igual a 1.

### Fase de Recuperación

La $i$-ésima coordenada $y_i^\omega$ del vector de clase $\mathbf{y}^\omega \in A^p$ se obtiene como lo indica la siguiente expresión, donde $\vee$ es el operador *máximo*:

$$y_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^n m_{ij} \cdot x_j^\omega = \bigvee_{h=1}^p \left[ \sum_{j=1}^n m_{hj} \cdot x_j^\omega \right] \\ 0 & \text{en otro caso} \end{cases}$$

## *Correlograph* de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa [11, 31]. En palabras de los autores "el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental".

### Algoritmo del *Correlograph*

### Fase de Aprendizaje

La *red asociativa* se genera al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^m$. Cada uno de los componentes $m_{ij}$ de la *red asociativa* $\mathbf{M}$ tiene valor cero al inicio, y se actualiza de acuerdo con la regla:

$$m_{ij} = \begin{cases} 1 & \text{si } y_i^\mu = 1 = x_j^\mu \\ \textit{valor anterior} & \text{en otro caso} \end{cases}$$

### Fase de Recuperación

Se le presenta a la *red asociativa* $\mathbf{M}$ un vector de entrada $\mathbf{x}^\omega \in A^n$. Se realiza el producto de la matriz $\mathbf{M}$ por el vector $\mathbf{x}^\omega$ y se ejecuta una operación de umbralizado, de acuerdo con la siguiente expresión:

$$y_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^n m_{ij} \cdot x_j^\omega \geq u \\ 0 & \text{en otro caso} \end{cases}$$

donde $u$ es el valor de umbral. Una estimación aproximada del valor de umbral $u$ se puede lograr con la ayuda de un número indicador mencionado en el artículo [11] de Willshaw *et al.* de 1969: $log_2 n$.

## Linear Associator de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen [12, 13, 36].

Para presentar el *Linear Associator* consideremos de nuevo el conjunto fundamental:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\} \text{ con } A = \{0, 1\}, \quad \mathbf{x}^\mu \in A^n \text{ y } \mathbf{y}^\mu \in A^m$$

### Algoritmo del *Linear Associator*

#### Fase de Aprendizaje

1) Para cada una de las $p$ asociaciones $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se encuentra la matriz $\mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t$ de dimensiones $m$ x $n$

2) Se suman la $p$ matrices para obtener la memoria

$$\mathbf{M} = \sum_{\mu=1}^{p} \mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t = [m_{ij}]_{m \times n}$$

de manera que la $ij$-ésima componente de la memoria M se expresa así:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^\mu x_j^\mu$$

#### Fase de Recuperación

Esta fase consiste en presentarle a la memoria un patrón de entrada $\mathbf{x}^\omega$, donde $\omega \in \{1, 2, ..., p\}$ y realizar la operación

$$\mathbf{M} \cdot \mathbf{x}^\omega = \left[ \sum_{\mu=1}^{p} \mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t \right] \cdot \mathbf{x}^\omega$$

## La memoria asociativa Hopfield

El artículo de John J. Hopfield de 1982, publicado por la prestigiosa y respetada *National Academy of Sciences* (en sus *Proceedings*), impactó positivamente y trajo a la palestra internacional su famosa memoria asociativa [8].

En el modelo que originalmente propuso Hopfield, cada neurona $x_i$ tiene dos posibles estados, a la manera de las neuronas de McCulloch-Pitts: $x_i = 0$ y $x_i = 1$; sin embargo, Hopfield observa que, para un nivel dado de exactitud en la recuperación de patrones, la capacidad de almacenamiento de información de la memoria se puede incrementar por un

factor de 2, si se escogen como posibles estados de las neuronas los valores $x_i = -1$ y $x_i = 1$ en lugar de los valores originales $x_i = 0$ y $x_i = 1$.

Al utilizar el conjunto {-1, 1} y el valor de umbral cero, la fase de aprendizaje para la memoria Hopfield será similar, en cierta forma, a la fase de aprendizaje del *Linear Associator*. La intensidad de la fuerza de conexión de la neurona $x_i$ a la neurona $x_j$ se representa por el valor de $m_{ij}$, y se considera que hay simetría, es decir, $m_{ij} = m_{ji}$. Si $x_i$ no está conectada con $x_j$ entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona a sí misma, lo cual significa que $m_{ij} = 0$. El estado instantáneo del sistema está completamente especificado por el vector columna de dimensión $n$ cuyas coordenadas son los valores de las $n$ neuronas.

La memoria Hopfield es autoasociativa, simétrica, con ceros en la diagonal principal. En virtud de que la memoria es autoasociativa, el conjunto fundamental para la memoria Hopfield es $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, ..., p\}$ con $\mathbf{x}^\mu \in A^n$ y $A = \{-1, 1\}$

## Algoritmo Hopfield

### Fase de Aprendizaje

La fase de aprendizaje para la memoria Hopfield es similar a la fase de aprendizaje del *Linear Associator*, con una ligera diferencia relacionada con la diagonal principal en ceros, como se muestra en la siguiente regla para obtener la $ij$-ésima componente de la memoria Hopfield $\mathbf{M}$:

$$m_{ij} = \begin{cases} \sum_{\mu=1}^{p} x_i^\mu x_j^\mu & \text{si } i \neq j \\ 0 \text{ si } i = j \end{cases}$$

### Fase de Recuperación

Si se le presenta un patrón de entrada $\tilde{\mathbf{x}}$ a la memoria Hopfield, ésta cambiará su estado con el tiempo, de modo que cada neurona $x_i$ ajuste su valor de acuerdo con el resultado que arroje la comparación de la cantidad $\sum_{j=1}^{n} m_{ij} x_j$ con un valor de umbral, el cual normalmente se coloca en cero.

Se representa el estado de la memoria Hopfield en el tiempo $t$ por $\mathbf{x}(t)$; entonces $x_i(t)$ representa el valor de la neurona $x_i$ en el tiempo $t$ y $x_i(t+1)$ el valor de $x_i$ en el tiempo siguiente $(t+1)$.

Dado un vector columna de entrada $\tilde{\mathbf{x}}$, la fase de recuperación consta de tres pasos:

1) Para $t = 0$, se hace $\mathbf{x}(t) = \tilde{\mathbf{x}}$; es decir, $x_i(0) = \tilde{x}_i$, $\forall i \in \{1,2,3,...,n\}$

2) $\forall i \in \{1,2,3,...,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) > 0 \\ x_i(t) \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) = 0 \\ -1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) < 0 \end{cases}$$

3) Se compara $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1, 2, 3,...,n\}$ . Si $\mathbf{x}(t+1) = \mathbf{x}(t)$ el proceso termina y el vector recuperado es $\mathbf{x}(0) = \tilde{\mathbf{x}}$. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar a un valor $t = \tau$ para el cual $x_i(\tau+1) = x_i(\tau)$ $\forall i \in \{1, 2, 3,...,n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

En el artículo original de 1982, Hopfield había estimado empíricamente que su memoria tenía una capacidad de recuperar $0.15n$ patrones, y en el trabajo de Abu-Mostafa & St. Jacques [32] se estableció formalmente que una cota superior para el número de vectores de estado arbitrarios estables en una memoria Hopfield es $n$.

## Memoria Asociativa Bidireccional (BAM) de Kosko.

Bart Kosko, investigador de la *University of Southern California*, propuso en 1988 la *Bidireccional Associative Memory* (BAM) [14] para subsanar la clara desventaja de la autoasociatividad de la memoria Hopfield. La BAM maneja pares de vectores $(A_1,B_1)$, ... $(A_m,B_m)$, donde $A \in \{0, 1\}^n$ y $B \in \{0, 1\}^p$.

Al igual que Austin ensambló dos redes asociativas de Willshaw para diseñar su ADAM [33], Kosko ideó un arreglo de dos memorias Hopfield, y demostró que este diseño es capaz de asociar patrones de manera heteroasociativa.

La matriz $\mathbf{M}$ es una memoria Hopfield con la única diferencia que la diagonal principal es diferente de cero. $\mathbf{M}^T$ es la matriz transpuesta de $\mathbf{M}$ que, ahora como entrada, recibe a $B$ y la salida será $A$. El proceso bidireccional anteriormente ilustrado continúa hasta que $A$ y $B$ convergen a una pareja estable $(A_i, B_i)$.

$$
\begin{aligned}
&A \rightarrow \mathbf{M} \rightarrow B \\
&A' \leftarrow \mathbf{M}^T \leftarrow B \\
&A' \rightarrow \mathbf{M} \rightarrow B' \\
&A'' \leftarrow \mathbf{M}^T \leftarrow B' \\
&\qquad\qquad \cdots \\
&A_i \rightarrow \mathbf{M} \rightarrow B_i \\
&A_i \leftarrow \mathbf{M}^T \leftarrow B_i
\end{aligned}
$$

Kosko descubrió que su memoria funcionaba mejor con patrones bipolares que con patrones binarios (a la manera de Hopfield), por tanto: $A \in \{-1, 1\}^n$ y $B \in \{-1, 1\}^p$

Para la codificación de la BAM se superponen las $m$ asociaciones sumándolas para formar la matriz de correlación:

$$\mathbf{M} = \sum_i A_i^T B_i$$

y la memoria dual $\mathbf{M}^T$ que está dada por:

$$\mathbf{M}^T = \sum_i \left(A_i^T B_i\right)^t = \sum_i B_i^T A_i$$

En el proceso de decodificación, cada neurona $a_i$ que se encuentra en el campo $A$ y cada neurona $b_i$ localizada en el campo $B$, de forma independiente y asíncrona, examina la suma de entrada de las neuronas del otro campo, entonces puede o no cambiar su estado si la suma de entrada es mayor, igual o menor que un umbral dado. Si la suma de entrada es igual al umbral, entonces la neurona no cambia su estado. La suma de entrada para $b_j$ es el producto interno columna:

$$A\mathbf{M}^j = \sum_i a_i m_{ij}$$

donde $\mathbf{M}^j$ es la $j$-ésima columna de $\mathbf{M}$. La suma de entrada para $a_i$ es, de manera similar,

$$B\mathbf{M}_i^T = \sum_j b_j m_{ij}$$

donde $\mathbf{M}_i$ es la $i$-ésima fila de $\mathbf{M}$. Se toma el 0 como el umbral para todas las neuronas. Las funciones de umbral para $a_i$ y $b_j$ son:

$$a_i = \begin{cases} 1, & \text{si} \quad B\mathbf{M}_i^T > 0 \\ -1, & \text{si} \quad B\mathbf{M}_i^T < 0 \end{cases}$$

$$b_j = \begin{cases} 1, & \text{si} \quad A\mathbf{M}^j > 0 \\ -1, & \text{si} \quad A\mathbf{M}^j < 0 \end{cases}$$

Cuando se le presenta un patrón ($A$, $B$) a la BAM, las neuronas en los campos $A$ y $B$ se prenden o se apagan de acuerdo a la ocurrencia de 1's y 0's en los vectores de estado $A$ y $B$. Las neuronas continúan sus cambios de estado hasta que se alcance un estado estable bidireccional ($A_f$, $B_f$).

## Memorias Asociativas Morfológicas

La diferencia fundamental entre las memorias asociativas clásicas (*Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Asociativa Hopfield) y las memorias asociativas morfológicas radica en los fundamentos operacionales de éstas últimas, que son las operaciones morfológicas de *dilatación* y *erosión*; el nombre de las memorias asociativas morfológicas está inspirado precisamente en estas dos operaciones básicas.

Estas memorias rompieron con el esquema utilizado a través de los años en los modelos de memorias asociativas clásicas, que utilizan operaciones convencionales entre vectores y matrices para la fase de aprendizaje y suma de productos para la recuperación de patrones. Las memorias asociativas morfológicas cambian los productos por sumas y las sumas por máximos o mínimos en ambas fases, tanto de aprendizaje como de recuperación de patrones [19, 34, 35].

Hay dos tipos de memorias asociativas morfológicas: las memorias *Max*, simbolizadas con **M**, y las memorias *Min*, cuyo símbolo es **W**; en cada uno de los dos tipos, las memorias pueden funcionar en ambos modos: heteroasociativo y autoasociativo.

Se definen dos nuevos productos matriciales:

El *producto máximo* entre **D** y **H**, denotado por $\mathbf{C} = \mathbf{D} \nabla \mathbf{H}$, es una matriz $[c_{ij}]_{m \times n}$ cuya $ij$-ésima componente $c_{ij}$ es

$$c_{ij} = \bigvee_{k=1}^{r} \left( d_{ik} + h_{kj} \right)$$

El *producto mínimo* de **D** y **H** denotado por $\mathbf{C} = \mathbf{D} \Delta \mathbf{H}$, es una matriz $[c_{ij}]_{m \times n}$ cuya $ij$-ésima componente $c_{ij}$ es

$$c_{ij} = \bigwedge_{k=1}^{r} \left( d_{ik} + h_{kj} \right)$$

Los productos máximo y mínimo contienen a los operadores máximo y mínimo, los cuales están íntimamente ligados con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

### Algoritmo de las memorias Heteroasociativas morfológicas *Max*

### Fase de Aprendizaje

1. Para cada una de las $p$ asociaciones $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se usa el producto mínimo para crear la matriz $\mathbf{y}^\mu \Delta (-\mathbf{x}^\mu)^t$ de dimensiones $m$ x $n$, donde el negado transpuesto del patrón de entrada $\mathbf{x}^\mu$ se define como $(-\mathbf{x}^\mu)^t = \left(-x_1^\mu, -x_2^\mu, ..., x_n^\mu\right)$.
2. Se aplica el operador máximo $\vee$ a las $p$ matrices para obtener la memoria $\mathbf{M}$.

$$\mathbf{M} = \bigvee_{\mu=1}^{p} \left[\mathbf{y}^\mu \Delta (-\mathbf{x}^\mu)^t\right]$$

### Fase de Recuperación

Esta fase consiste en realizar el producto mínimo $\Delta$ de la memoria $\mathbf{M}$ con el patrón de entrada $\mathbf{x}^\omega$, donde $\omega \in \{1, 2, ..., p\}$, para obtener un vector columna $\mathbf{y}$ de dimensión $m$:

$$\mathbf{y} = \mathbf{M} \Delta \mathbf{x}^\omega$$

Las fases de aprendizaje y de recuperación de las **memorias morfológicas *Min*** se obtienen por dualidad.

### Memorias Autoasociativas Morfológicas

Para este tipo de memorias se utilizan los mismos algoritmos descritos anteriormente y que son aplicados a las memorias heteroasociativas; lo único que cambia es el conjunto fundamental. Para este caso, se considera el siguiente conjunto fundamental:

$$\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mathbf{x}^\mu \in A^n, \text{ donde } \mu = 1, 2, ..., p\}$$

## 2.2.7 Memorias Asociativas Alfa-Beta

Las memorias asociativas Alfa-Beta [20] utilizan máximos y mínimos, y dos operaciones binarias originales $\alpha$ y $\beta$ de las cuales heredan el nombre.
Para la definición de las operaciones binarias $\alpha$ y $\beta$ se deben especificar los conjuntos $A$ y $B$, los cuales son:

$$A = \{0, 1\} \quad \text{y} \quad B = \{0, 1, 2\}$$

La operación binaria $\alpha$: $A$ x $A \rightarrow B$ se define como:

| $x$ | $y$ | $\alpha(x, y)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 2 |
| 1 | 1 | 1 |

La operación binaria $\beta$: $B$ x $A \rightarrow A$ se define como:

| $x$ | $y$ | $\beta(x, y)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 0 | 1 |
| 2 | 1 | 1 |

El fundamento teórico de las memorias asociativas Alfa-Beta se presenta en el siguiente capítulo de forma más completa, debido a que estas memorias son la base fundamental para este trabajo de tesis.

## Memorias Asociativas Mediana

Las Memorias Asociativas Mediana [27] utilizan los operadores A y B, definidos de la siguiente forma:

$$A(x, y) = x - y$$
$$B(x, y) = x + y$$

Las operaciones utilizadas se describen a continuación.

Sean $P = [p_{ij}]_{mxr}$ y $Q=[q_{ij}]_{rxn}$ dos matrices.

Operación $\lozenge_A$: $P_{mxr} \lozenge_A Q_{rxn} = [f_{ij}^A]_{mxn}$ donde $f_{ij}^A = \underset{k=1}{\overset{r}{\mathbf{med}}} A(p_{ik}, q_{k,j})$

Operación $\lozenge_B$: $P_{mxr} \lozenge_B Q_{rxn} = [f_{ij}^B]_{mxn}$ donde $f_{ij}^B = \underset{k=1}{\overset{r}{\mathbf{med}}} B(p_{ik}, q_{k,j})$

### Algoritmo de las Memorias Mediana

#### Fase de Aprendizaje

**Paso 1.** Para cada $\xi = 1, 2, ..., p$, de cada pareja $(\mathbf{x}^\xi, \mathbf{y}^\xi)$ se construye la matriz:

$$[\mathbf{y}^\xi \lozenge_A (\mathbf{x}^\xi)^t]_{mxn}$$

**Paso 2.** Se aplica el operador media a las matrices obtenidas en le paso 1 para obtener la matriz **M**, como sigue:

$$\mathbf{M} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \left[ y^{\xi} \lozenge_{\mathrm{A}} \left( x^{\xi} \right)^{t} \right]$$

El *ij*-ésimo componente **M** está dado como sigue:

$$m_{ij} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \mathrm{A}\left( y_i^{\xi}, x_j^{\xi} \right)$$

### Fase de Recuperación

Se tienen dos casos:

**Caso 1.** Recuperación de un patrón fundamental. Un patrón $\mathbf{x}^w$, con $w \in \{1, 2, ..., p\}$ se le presenta a la memoria **M** y se realiza la siguiente operación:

$$\mathbf{M} \lozenge_{\mathrm{B}} \mathbf{x}^w$$

El resultado es un vector columna de dimensión *n*, con la *i*-ésima componente dada como:

$$\left( \mathbf{M} \lozenge_B x^w \right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left( m_{ij}, x_j^w \right)$$

**Caso 2.** Recuperación de un patrón alterado. Un patrón $\tilde{\mathbf{x}}$, que es una versión alterada de un patrón $\mathbf{x}^w$, se le presenta a la memoria **M** y se realiza la siguiente operación:

$$\mathbf{M} \lozenge_{\mathrm{B}} \tilde{\mathbf{x}}$$

De nuevo, es resultado es un vector de dimensión *n*, con la *i*-ésima componente dada como:

$$\left( \mathbf{M} \lozenge_B \tilde{\mathbf{x}} \right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left( m_{ij}, \tilde{x}_j \right)$$

# Apéndice D

# Ejemplos de los algoritmos originales de las memorias asociativas Alfa-Beta

## Ejemplo de las memorias autoasociativas

El propósito de esta sección es la de ilustrar mediante un ejemplo el funcionamiento de las memorias auto asociativas alfa beta de ambos tipos. Para este ejemplo utilizaremos patrones de entrada de dimensión 4, . Utilizaremos el siguiente conjunto fundamental:

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

### Fase Aprendizaje:

Para aprender tenemos que p=3, asi que obtendremos las matrices representantes de cada asociación, entonces tenemos:

Para $\mu = 1$, tenemos $\left(\mathbf{x}^1, \mathbf{x}^1\right)$ construimos la matriz que representa la asociación 1

$$\left[\mathbf{x}^1 \otimes \left(\mathbf{x}^1\right)^t\right] = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) \\ \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

$$\left[\mathbf{x}^2 \otimes \left(\mathbf{x}^2\right)^t\right] = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \alpha(1,1) & \alpha(1,1) & \alpha(1,0) & \alpha(1,0) \\ \alpha(1,1) & \alpha(1,1) & \alpha(1,0) & \alpha(1,0) \\ \alpha(0,1) & \alpha(0,1) & \alpha(0,0) & \alpha(0,0) \\ \alpha(0,1) & \alpha(0,1) & \alpha(0,0) & \alpha(0,0) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\left[\mathbf{x}^3 \otimes \left(\mathbf{x}^3\right)^t\right] = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha(0,0) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,0) & \alpha(1,0) & \alpha(1,0) & \alpha(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

Paso 2.- La memoria asociativa Max la obtendremos de aplicar el operador máximo entre las 3 matrices obtenidas:

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

Paso 3.- La memoria asociativa Min la obtenemos de aplicar el operador minimo entre las 3 matrices obtenidas:

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 \end{pmatrix} \wedge \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \wedge \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 2 & 2 & 1 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

### Fase de recuperación

Una vez que obtuvimos las memorias Min y Max trataremos de recuperar los patrones del conjunto fundamental. Empezaremos con la memoria Max e introduciremos los 3 patrones de entrada del conjunto fundamental para ver si recupera correctamente.

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} \quad \mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Para $\mathbf{x}^1$ tenemos:

$$\left(\mathbf{M} \Delta_\beta \mathbf{x}^1\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j^1\right)$$

$$= \begin{pmatrix} \beta(1,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(1,0) & \wedge & \beta(1,1) & \wedge & \beta(1,0) & \wedge & \beta(1,1) \\ \beta(2,0) & \wedge & \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 0 & \wedge & 1 & \wedge & 0 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Para $\mathbf{x}^2$ tenemos:

$$\left(\mathbf{M} \Delta_\beta \mathbf{x}^2\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j^2\right)$$

$$= \begin{pmatrix} \beta(1,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) \\ \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) \\ \beta(1,1) & \wedge & \beta(1,1) & \wedge & \beta(1,0) & \wedge & \beta(1,0) \\ \beta(2,1) & \wedge & \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,0) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 0 & \wedge & 1 & \wedge & 0 & \wedge & 0 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Para $\mathbf{x}^3$ tenemos:

$$\left(\mathbf{M} \Delta_\beta \mathbf{x}^3\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j^3\right)$$

$$= \begin{pmatrix} \beta(1,0) & \wedge & \beta(1,0) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(1,0) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(1,0) & \wedge & \beta(1,0) & \wedge & \beta(1,0) & \wedge & \beta(1,1) \\ \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \wedge & 0 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 0 & \wedge & 1 & \wedge & 1 \\ 0 & \wedge & 0 & \wedge & 0 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Como se puede verificar la memoria autoasociativa Max recupero bien los 3 patrones miembros del conjunto fundamental

Ahora recuperaremos los patrones usando la memoria Min

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \mathbf{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Para $\mathbf{x}^1$ tenemos:

$$\left(\mathbf{W} \nabla_B \mathbf{x}^1\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^1\right)$$

$$= \begin{pmatrix} \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(1,0) & \vee & \beta(1,1) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(0,0) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(0,0) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 1 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Para $\mathbf{x}^2$ tenemos:

$$\left(\mathbf{W} \nabla_B \mathbf{x}^2\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^2\right)$$

$$= \begin{pmatrix} \beta(1,1) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,0) \\ \beta(1,1) & \vee & \beta(1,1) & \vee & \beta(1,0) & \vee & \beta(0,0) \\ \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,0) \\ \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(1,0) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 1 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Para $\mathbf{x}^3$ tenemos:

$$\left(\mathbf{W} \nabla_B \mathbf{x}^3\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^3\right)$$

$$= \begin{pmatrix} \beta(1,0) & \vee & \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(1,0) & \vee & \beta(1,0) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(0,0) & \vee & \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(0,0) & \vee & \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Como se puede verificar la memoria autoasociativa Min tambien recupero bien los 3 patrones miembros del conjunto fundamental.

## Ejemplo de las memorias heteroasociativas

El propósito de esta sección es la de ilustrar mediante un ejemplo el funcionamiento de las memorias alfa beta de ambos tipos. Para este ejemplo utilizaremos patrones de entrada de

dimensión 5 y patrones de salida de dimensión 6. Utilizaremos el siguiente conjunto fundamental:

$$\mathrm{x}^1=\begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix}\quad \mathbf{y}^1=\begin{pmatrix}1\\0\\1\\0\\1\\0\end{pmatrix}\qquad \mathbf{x}^2=\begin{pmatrix}0\\1\\1\\0\\0\end{pmatrix}\quad \mathbf{y}^2=\begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix}\qquad \mathbf{x}^3=\begin{pmatrix}0\\0\\1\\0\\1\end{pmatrix}\quad \mathbf{y}^3=\begin{pmatrix}1\\0\\1\\0\\0\\1\end{pmatrix}$$

**Fase de Aprendizaje:**

Para aprender tenemos que p=3, asi que obtendremos las matrices representantes de cada asociación como se vio, entonces tenemos:

Para $\mu=1$, tenemos $\left(\mathbf{x}^1,\mathbf{y}^1\right)$ construimos la matriz que representa la asociación 1

$$\left[\mathrm{y}^1\otimes\left(\mathbf{x}^1\right)^t\right]=\begin{pmatrix}1\\0\\1\\0\\1\\0\end{pmatrix}\otimes\begin{pmatrix}1&1&1&1&1\end{pmatrix}=\begin{pmatrix}\alpha(1,1)&\alpha(1,1)&\alpha(1,1)&\alpha(1,1)&\alpha(1,1)\\ \alpha(0,1)&\alpha(0,1)&\alpha(0,1)&\alpha(0,1)&\alpha(0,1)\\ \alpha(1,1)&\alpha(1,1)&\alpha(1,1)&\alpha(1,1)&\alpha(1,1)\\ \alpha(0,1)&\alpha(0,1)&\alpha(0,1)&\alpha(0,1)&\alpha(0,1)\\ \alpha(1,1)&\alpha(1,1)&\alpha(1,1)&\alpha(1,1)&\alpha(1,1)\\ \alpha(0,1)&\alpha(0,1)&\alpha(0,1)&\alpha(0,1)&\alpha(0,1)\end{pmatrix}$$

$$=\begin{pmatrix}1&1&1&1&1\\0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\end{pmatrix}$$

Para $\mu=2$, tenemos $\left(\mathbf{x}^2,\mathbf{y}^2\right)$ construimos la matriz que representa la asociación 2:

$$\left[\mathbf{y}^2\otimes\left(\mathbf{x}^2\right)^t\right]=\begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix}\otimes\begin{pmatrix}0&1&1&0&0\end{pmatrix}=\begin{pmatrix}\alpha(0,0)&\alpha(0,1)&\alpha(0,1)&\alpha(0,0)&\alpha(0,0)\\ \alpha(0,0)&\alpha(0,1)&\alpha(0,1)&\alpha(0,0)&\alpha(0,0)\\ \alpha(1,0)&\alpha(1,1)&\alpha(1,1)&\alpha(1,0)&\alpha(1,0)\\ \alpha(1,0)&\alpha(1,1)&\alpha(1,1)&\alpha(1,0)&\alpha(1,0)\\ \alpha(1,0)&\alpha(1,1)&\alpha(1,1)&\alpha(1,0)&\alpha(1,0)\\ \alpha(1,0)&\alpha(1,1)&\alpha(1,1)&\alpha(1,0)&\alpha(1,0)\end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \end{pmatrix}$$

Para $\mu = 3$, tenemos $\left(\mathbf{x}^3, \mathbf{y}^3\right)$ construimos la matriz que representa la asociación 3

$$\left[\mathbf{y}^3 \otimes \left(\mathbf{x}^3\right)^t\right] = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha(1,0) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,0) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,0) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) \end{pmatrix}$$

$$= \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 2 & 1 \end{pmatrix}$$

Paso 2.- La memoria asociativa Max la obtendremos de aplicar el operador máximo entre las 3 matrices obtenidas:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \end{pmatrix} \vee \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 2 & 1 \end{pmatrix}$$

$$\mathbf{M}=\begin{pmatrix} 1\vee 1\vee 2 & 1\vee 0\vee 2 & 1\vee 0\vee 1 & 1\vee 1\vee 2 & 1\vee 1\vee 1 \\ 0\vee 1\vee 1 & 0\vee 0\vee 1 & 0\vee 0\vee 0 & 0\vee 1\vee 1 & 0\vee 1\vee 0 \\ 1\vee 2\vee 2 & 1\vee 1\vee 2 & 1\vee 1\vee 1 & 1\vee 2\vee 2 & 1\vee 2\vee 1 \\ 0\vee 2\vee 1 & 0\vee 1\vee 1 & 0\vee 1\vee 0 & 0\vee 2\vee 1 & 0\vee 2\vee 0 \\ 1\vee 2\vee 1 & 1\vee 1\vee 1 & 1\vee 1\vee 0 & 1\vee 2\vee 1 & 1\vee 2\vee 0 \\ 0\vee 2\vee 2 & 0\vee 1\vee 2 & 0\vee 1\vee 1 & 0\vee 2\vee 2 & 0\vee 2\vee 1 \end{pmatrix}$$

$$\mathbf{M}=\begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 \end{pmatrix}$$

Paso 3.- La memoria heteroasociativa Min la obtenemos de aplicar el operador Min a las 3 matrices obtenidas.

$$\mathbf{W}=\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \wedge \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \end{pmatrix} \wedge \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 1 & 2 & 1 \end{pmatrix}$$

$$\mathbf{W}=\begin{pmatrix} 1\wedge 1\wedge 2 & 1\wedge 0\wedge 2 & 1\wedge 0\wedge 1 & 1\wedge 1\wedge 2 & 1\wedge 1\wedge 1 \\ 0\wedge 1\wedge 1 & 0\wedge 0\wedge 1 & 0\wedge 0\wedge 0 & 0\wedge 1\wedge 1 & 0\wedge 1\wedge 0 \\ 1\wedge 2\wedge 2 & 1\wedge 1\wedge 2 & 1\wedge 1\wedge 1 & 1\wedge 2\wedge 2 & 1\wedge 2\wedge 1 \\ 0\wedge 2\wedge 1 & 0\wedge 1\wedge 1 & 0\wedge 1\wedge 0 & 0\wedge 2\wedge 1 & 0\wedge 2\wedge 0 \\ 1\wedge 2\wedge 1 & 1\wedge 1\wedge 1 & 1\wedge 1\wedge 0 & 1\wedge 2\wedge 1 & 1\wedge 2\wedge 0 \\ 0\wedge 2\wedge 2 & 0\wedge 1\wedge 2 & 0\wedge 1\wedge 1 & 0\wedge 2\wedge 2 & 0\wedge 2\wedge 1 \end{pmatrix}$$

$$\mathbf{W}=\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

### Fase de recuperación

Una vez que obtuvimos las memorias Min y Max trataremos de recuperar los patrones del conjunto fundamental. Empezaremos con la memoria Max e introduciremos los 3 patrones de entrada del conjunto fundamental para ver si recupera el patrón $\mathbf{y}^{\mu}$ correspondiente

$$\mathbf{M}=\begin{pmatrix}2&2&1&2&1\\1&1&0&1&1\\2&2&1&2&2\\2&1&1&2&2\\2&1&1&2&2\\2&2&1&2&2\end{pmatrix} \qquad \mathrm{x}^1=\begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix} \qquad \mathbf{x}^2=\begin{pmatrix}0\\1\\1\\0\\0\end{pmatrix} \qquad \mathbf{x}^3=\begin{pmatrix}0\\0\\1\\0\\1\end{pmatrix}$$

Para $\mathbf{x}^1$ tenemos:

$$\left(\mathbf{M}\,\Delta_{\beta}\mathbf{x}^1\right)_i=\bigwedge_{j=1}^{n}\beta\left(m_{ij},x_j^1\right)$$

$$=\begin{pmatrix}\beta(2,1)\wedge\beta(2,1)\wedge\beta(1,1)\wedge\beta(2,1)\wedge\beta(1,1)\\\beta(1,1)\wedge\beta(1,1)\wedge\beta(0,1)\wedge\beta(1,1)\wedge\beta(1,1)\\\beta(2,1)\wedge\beta(2,1)\wedge\beta(1,1)\wedge\beta(2,1)\wedge\beta(2,1)\\\beta(2,1)\wedge\beta(1,1)\wedge\beta(1,1)\wedge\beta(2,1)\wedge\beta(2,1)\\\beta(2,1)\wedge\beta(1,1)\wedge\beta(1,1)\wedge\beta(2,1)\wedge\beta(2,1)\\\beta(2,1)\wedge\beta(2,1)\wedge\beta(1,1)\wedge\beta(2,1)\wedge\beta(2,1)\end{pmatrix}$$

$$=\begin{pmatrix}1\wedge1\wedge1\wedge1\wedge1\\1\wedge1\wedge0\wedge0\wedge1\\1\wedge1\wedge1\wedge1\wedge1\\1\wedge1\wedge1\wedge1\wedge1\\1\wedge1\wedge1\wedge1\wedge1\\1\wedge1\wedge1\wedge1\wedge1\end{pmatrix}=\begin{pmatrix}1\\0\\1\\1\\1\\1\end{pmatrix}$$

Para $\mathbf{x}^2$ tenemos:

$$\left(\mathbf{M}\,\Delta_{\beta}\mathbf{x}^2\right)_i=\bigwedge_{j=1}^{n}\beta\left(m_{ij},x_j^2\right)$$

$$= \begin{pmatrix} \beta(2,0) & \wedge & \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,0) \\ \beta(1,0) & \wedge & \beta(1,1) & \wedge & \beta(0,1) & \wedge & \beta(1,0) & \wedge & \beta(1,0) \\ \beta(2,0) & \wedge & \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) \\ \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) \\ \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) \\ \beta(2,0) & \wedge & \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 0 \\ 0 & \wedge & 1 & \wedge & 0 & \wedge & 0 & \wedge & 0 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Para $\mathbf{x}^3$ tenemos:

$$\left(\mathbf{M} \Delta_\beta \mathbf{x}^3\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j^3\right)$$

$$= \begin{pmatrix} \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) \\ \beta(1,0) & \wedge & \beta(1,0) & \wedge & \beta(0,1) & \wedge & \beta(1,0) & \wedge & \beta(1,1) \\ \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(1,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(1,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 0 & \wedge & 0 & \wedge & 0 & \wedge & 0 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Como se puede verificar solo fallo al recuperar el patron correspondiente a $\mathrm{x}^1$, y con los otros 2 tuvo una recuperacion correcta.

Ahora recuperaremos los patrones usando la memoria Min

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathrm{x}^1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Para $\mathbf{x}^1$ tenemos:

$$\left(\mathbf{W} \nabla_B \mathbf{x}^1\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^1\right)$$

$$= \begin{pmatrix} \beta(1,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(1,1) & \vee & \beta(1,1) \\ \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) \\ \beta(1,1) & \vee & \beta(1,1) & \vee & \beta(1,1) & \vee & \beta(1,1) & \vee & \beta(1,1) \\ \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) \\ \beta(1,1) & \vee & \beta(1,1) & \vee & \beta(0,1) & \vee & \beta(1,1) & \vee & \beta(0,1) \\ \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \vee & 0 & \vee & 0 & \vee & 1 & \vee & 1 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 1 & \vee & 1 & \vee & 1 & \vee & 1 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 1 & \vee & 0 & \vee & 1 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Para $\mathbf{x}^2$ tenemos:

$$\left(\mathbf{W} \nabla_B \mathbf{x}^2\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^2\right)$$

$$= \begin{pmatrix} \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(1,0) \\ \beta(0,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,0) \\ \beta(1,0) & \vee & \beta(1,1) & \vee & \beta(1,1) & \vee & \beta(1,0) & \vee & \beta(1,0) \\ \beta(0,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,0) \\ \beta(1,0) & \vee & \beta(1,1) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,0) \\ \beta(0,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,0) \end{pmatrix}$$

$$= \begin{pmatrix} 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 1 \vee 1 \vee 0 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 1 \vee 0 \vee 0 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Para $\mathbf{x}^3$ tenemos:

$$\left(\mathbf{W} \nabla_B \mathbf{x}^3\right)_i = \bigvee_{j=1}^{n} \beta\left(w_{ij}, x_j^3\right)$$

$$= \begin{pmatrix} \beta(1,0) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(1,0) \vee \beta(1,1) \\ \beta(0,0) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(0,1) \\ \beta(1,0) \vee \beta(1,0) \vee \beta(1,1) \vee \beta(1,0) \vee \beta(1,1) \\ \beta(0,0) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(0,1) \\ \beta(1,0) \vee \beta(1,0) \vee \beta(0,1) \vee \beta(1,0) \vee \beta(0,1) \\ \beta(0,0) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(0,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 \vee 0 \vee 0 \vee 0 \vee 1 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 0 \vee 1 \vee 1 \vee 1 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Como se puede verificar solo recupero bien el patron correspondiente a $x^1$, y con los otros 2 aunque fallo la diferencia es pequeña.

## Ejemplo de memorias asociativas bidireccionales

Anteriormente, en este mismo apéndice, se dio un ejemplo en el cual a partir del siguiente conjunto fundamental se obtuvieron las memorias **M** y **W** mostradas**.** Tambien se mostro los patrones obtenidos al operar las memorias y si son correctos o no se indican en esa misma seccion.

$$\mathrm{x}^1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 \end{pmatrix}$$

Para obtener la BAM correspondiente a este conjunto fundamental, aprenderemos la memorias correspondientes a las asociaciones en el sentido inverso $(\mathbf{y}^\mu, \mathbf{x}^\mu)$. Dado que ahora nuestros patrones de entrada son los patrones de salida y viceversa, tenemos:

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{y}^1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{y}^2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{y}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

### Fase De Aprendizaje

Prrimero obtenemos las martrices representantes de la asociación:

$$\mathbf{y}^1 \otimes (\mathbf{x}^1)^t = \begin{pmatrix} \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) \\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) \\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) \\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) \\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) \end{pmatrix}$$

$$\mathbf{y}^1 \otimes (\mathbf{x}^1)^t = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \end{pmatrix}$$

$$\mathbf{y}^2 \otimes (\mathbf{x}^2)^t = \begin{pmatrix} \alpha(0,0) & \alpha(0,0) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) \\ \alpha(1,0) & \alpha(1,0) & \alpha(1,1) & \alpha(1,1) & \alpha(1,1) & \alpha(1,1) \\ \alpha(1,0) & \alpha(1,0) & \alpha(1,1) & \alpha(1,1) & \alpha(1,1) & \alpha(1,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) \\ \alpha(0,0) & \alpha(0,0) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) \end{pmatrix}$$

$$\mathbf{y}^2 \otimes (\mathbf{x}^2)^t = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{y}^3 \otimes (\mathbf{x}^3)^t = \begin{pmatrix} \alpha(0,1) & \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \alpha(0,1) & \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,0) & \alpha(1,1) \\ \alpha(0,1) & \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,0) & \alpha(1,1) \end{pmatrix}$$

$$\mathbf{y}^3 \otimes (\mathbf{x}^3)^t = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 1 \end{pmatrix}$$

La memoria **MR** está dada por:

$$\mathbf{MR} = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \end{pmatrix} \vee \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 1 \end{pmatrix}$$

$$\mathbf{MR} = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 2 & 2 \end{pmatrix}$$

La memoria **WR** está dada por:

$$\mathbf{WR} = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \end{pmatrix} \wedge \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \wedge \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 2 & 1 \end{pmatrix}$$

Entonces la memoria **WR** queda:

$$\mathbf{WR} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

### Fase De Recuperación

Ahora operaremos las memorias para recuperar los patrones asociados

$$\mathbf{MR} = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 2 & 2 \end{pmatrix} \quad \mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{x}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Ahora introducimos $\mathbf{x}^1$:

$$\left(\mathbf{MR} \, \Delta_\beta \mathrm{x}^1\right)_i = \bigwedge_{j=1}^{n} \beta\left(mr_{ij}, x_j^1\right)$$

$$= \begin{pmatrix} \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) \\ \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) \\ \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) & \wedge & \beta(2,0) \\ \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) \\ \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,1) & \wedge & \beta(2,0) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Introducimos $\mathbf{x}^2$:

$$\left(\mathbf{MR} \, \Delta_\beta \mathrm{x}^2\right)_i = \bigwedge_{j=1}^{n} \beta\left(mr_{ij}, x_j^2\right)$$

$$= \begin{pmatrix} \beta(1,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,1) \\ \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,1) & \wedge & \beta(2,1) & \wedge & \beta(2,1) \\ \beta(1,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,1) & \wedge & \beta(1,1) & \wedge & \beta(2,1) \\ \beta(1,0) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,1) & \wedge & \beta(2,1) & \wedge & \beta(2,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Introducimos $\mathbf{x}^3$:

$$\left(\mathbf{MR} \, \Delta_\beta \mathrm{x}^3\right)_i = \bigwedge_{j=1}^{n} \beta\left(mr_{ij}, x_j^3\right)$$

$$= \begin{pmatrix} \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,0) & \wedge & \beta(2,1) \\ \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,0) & \wedge & \beta(2,1) \\ \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \\ \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,0) & \wedge & \beta(2,1) \\ \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(2,0) & \wedge & \beta(2,1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \\ 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \\ 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \\ 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \\ 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Como se puede verificar, la memoria **MR** fue capaz de recuperar los patrones correspondientes de la asociación en los 3 casos.

Ahora recuperaremos con la memoria Min **WR**

$$\mathbf{WR} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{x}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Ahora introducimos $\mathbf{x}^1$:

$$\left(\mathbf{WR} \nabla_\beta \mathrm{x}^1\right)_i = \bigvee_{j=1}^{n} \beta\left(wr_{ij}, x_j^1\right)$$

$$= \begin{pmatrix} \beta(0,1) \vee \beta(1,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(0,0) \\ \beta(0,1) \vee \beta(1,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(1,1) \vee \beta(0,0) \\ \beta(1,1) \vee \beta(2,0) \vee \beta(1,1) \vee \beta(0,0) \vee \beta(1,1) \vee \beta(1,0) \\ \beta(0,1) \vee \beta(1,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(0,0) \\ \beta(1,1) \vee \beta(1,0) \vee \beta(0,1) \vee \beta(0,0) \vee \beta(0,1) \vee \beta(0,0) \end{pmatrix}$$

$$= \begin{pmatrix} 0 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 1 \vee 0 \\ 1 \vee 1 \vee 1 \vee 0 \vee 1 \vee 0 \\ 0 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 \\ 1 \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Introducimos $\mathbf{x}^2$:

$$\left(\mathbf{WR} \nabla_\beta \mathrm{x}^2\right)_i = \bigvee_{j=1}^{n} \beta\left(wr_{ij}, x_j^2\right)$$

$$= \begin{pmatrix} \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) \\ \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(1,1) & \vee & \beta(0,1) \\ \beta(1,0) & \vee & \beta(2,0) & \vee & \beta(1,1) & \vee & \beta(0,1) & \vee & \beta(1,1) & \vee & \beta(1,1) \\ \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) \\ \beta(1,0) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) & \vee & \beta(0,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 1 & \vee & 0 \\ 0 & \vee & 1 & \vee & 1 & \vee & 0 & \vee & 1 & \vee & 1 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Introducimos $\mathbf{x}^3$:

$$\left(\mathbf{WR} \nabla_{\beta} \mathrm{x}^3\right)_i = \bigvee_{j=1}^{n} \beta\left(wr_{ij}, x_j^3\right)$$

$$= \begin{pmatrix} \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,0) & \vee & \beta(0,1) \\ \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \beta(1,1) & \vee & \beta(2,0) & \vee & \beta(1,1) & \vee & \beta(0,0) & \vee & \beta(1,0) & \vee & \beta(1,1) \\ \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,0) & \vee & \beta(0,1) \\ \beta(1,1) & \vee & \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,0) & \vee & \beta(0,1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 1 & \vee & 1 & \vee & 0 & \vee & 0 & \vee & 1 \\ 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \\ 1 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 & \vee & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

La memoria **WR** fue capaz de recuperar correctamente los patrones asociados a $\mathbf{x}^2$ y $\mathbf{x}^3$

# Apéndice E

# Código en C++ de los Algortimos Simplificados

El propósito de este apéndice es dar a los lectores un ejemplo de la implementación de las memorias asociativas $\alpha\beta$ asi como de los algoritmos de aprendizaje y recuperación, tanto de los algoritmos originales como los simplificados de las memorias alfa beta.

Para este propósito se decidió crear una clase, debido principalmente a su portabilidad. La clase esta generada en C++ Builder version 6.0. Gracias a esta clase cualquier investigador que quiere usar las memorias $\alpha\beta$ en algún desarrollo ya no tendrá que preocuparse por implementar las memorias y asi podrá concentrarse en las metas que desea alcanzar.

Nota.- En el lenguaje C++ las doble diagonales "//" indican un comentario de linea, y "/*" nos sirve para empezar un bloque de comentario que se termina con "*/".

```
/*
  Programa: Clase memoria asociativa Alfa Beta
  Programador: Edgar Armando Catalán Salgado
  Escuela: CIC -IPN
  Pais: Mexico DF
  Descripcion: Este codigo es una clase  que nos permite gnerar instancias de las
               memorias asociativas alfa beta, contiene todas los metodos originales
               de aprendizaje y recuperación, asi como tambien los  algortimos
               simplificados.

*/

#pragma hdrstop

#include "ClaseMemoriaAlfaBeta.h"
#include <stdio.h>
#include <stdlib.h>

//---------------------------------------------------------------------------

#pragma package(smart_init)

//Handle definiton Error
#define MEMORY_OK     1;
#define MEMORY_ERROR -1;


const int DimMaxX=1000,DimMaxY=1000,NumMaxPatrones=1000;
```

```
class AlfaBeta//:Memoria
  {
   private:
     int Alfa(int DatoX,int DatoY);
     int Beta(int DatoX,int DatoY);
     int *MemoriaOcupada;
     int  StatusCode;
     int ContX,ContY;

   public:
  // Metodos
     AlfaBeta (int DimX, int DimY); // Constructor
     ~AlfaBeta();                   // Destructor

     void InicilizaMemoria();          //Inicializacion

    //Algoritmos originales aprendizaje
     void AprendeAsociacion(int VectorX[DimMaxX], int VectorY[DimMaxY]);
     void AprendeAsociacionMMin(int VectorX[DimMaxX], int VectorY[DimMaxY]);
     void AprendeAsociacionMMax(int VectorX[DimMaxX], int VectorY[DimMaxY]);

    //Algoritmos originales recuperacion
     void RecuperaPatron(int VectorX[DimMaxX]);
     void RecuperaPatronMMin(int VectorX[DimMaxX]);
     void RecuperaPatronMMax(int VectorX[DimMaxX]);

    // Aprendizaje Simplificado
     void SeparaPatrones(int VectorX[DimMaxX], int VectorY[DimMaxY]);
     void SeparaPatronX(int VectorX[DimMaxX]);
     void SeparaPatronY(int VectorY[DimMaxY]);

     void AprendeMaxAbsMMax(int VectorX0s[DimMaxX],int VectorY1s[DimMaxY], int
DimX0s,int DimY1s);
     void AprendeMinAbsMMin(int VectorX1s[DimMaxX], int VectorY0s[DimMaxY], int
DimX1s, int DimY0s);
     void AprendizajeRapido(int VectorX[DimMaxX], int VectorY[DimMaxY]);
     void AprendizajeRapidoMMax(int VectorX[DimMaxX], int VectorY[DimMaxY]);
     void AprendizajeRapidoMMin(int VectorX[DimMaxX], int VectorY[DimMaxY]);
     void AjustaValoresMemoria();

    //Conversion de memorias
     void ConvierteMMaxAMMin();
     void ConvierteMMinAMMax();

    //Recuperacion simplificada
     void SimplificaMemoria();
     void RecuperacionModificada(int VectorX[DimMaxX]);
     void RecuperacionModificadaMMin(int VectorX[DimMaxX]);
     void RecuperacionModificadaMMax(int VectorX[DimMaxX]);

   //Datos
```

```
      int DimX,DimY;
      int MMin[DimMaxX][DimMaxY];
      int MMax[DimMaxX][DimMaxY];
      int VectorSalidaMMax[DimMaxX],VectorSalidaMMin[DimMaxX];

      //aprendizaje
      int VectorX0s[DimMaxX],VectorX1s[DimMaxX];
      int VectorY0s[DimMaxY],VectorY1s[DimMaxY];
      int DimX0s, DimX1s, DimY0s,DimY1s;
      int
VectorEX0s[DimMaxX],VectorEX1s[DimMaxX],VectorEY0s[DimMaxY],VectorEY1s[DimMax
Y];

     //recuperacion
      int MemoriaMax1s[DimMaxX][DimMaxY];
      int MemoriaMin1s[DimMaxX][DimMaxY];
      int EMMax0s[DimMaxY],EMMin2s[DimMaxY];

     //Operaciones min,max
      int __fastcall min(int Valor1, int Valor2);
      int __fastcall max(int Valor1, int Valor2);
  };

//Constructor
AlfaBeta::AlfaBeta( int DimX, int DimY )
  {
   AlfaBeta::DimX = DimX;
   AlfaBeta::DimY = DimY;
   InicilizaMemoria();
   AlfaBeta::MemoriaOcupada = new int[ AlfaBeta::DimX * AlfaBeta::DimY ];
   if( AlfaBeta::MemoriaOcupada == NULL )
     {
      AlfaBeta::StatusCode = MEMORY_ERROR;
     }
   else
     {
      AlfaBeta::StatusCode = MEMORY_OK;
     }
  }

//Destructor
AlfaBeta::~AlfaBeta( )
  {
   delete [] AlfaBeta::MemoriaOcupada;
  }

void AlfaBeta::InicilizaMemoria()
  {
   for (ContY=0;ContY<DimY;ContY++)
    {
     for (ContX=0;ContX<DimX;ContX++)
```

```
      {
       MMin[ContX][ContY]=2;// se asigna el mayor por que hace el min
       MMax[ContX][ContY]=0;// se asigna el menor por que hace el max
      }
    }
  }

void AlfaBeta::AprendeAsociacionMMax(int VectorX[DimMaxX], int VectorY[DimMaxY])
  {
   for (ContY=0;ContY<DimY;ContY++)
    {
     for (ContX=0;ContX<DimX;ContX++)
      {
       MMax[ContX][ContY]=
max(Alfa(VectorY[ContY],VectorX[ContX]),MMax[ContX][ContY]);
      }
    }
  }

void AlfaBeta::AprendeAsociacionMMin(int VectorX[DimMaxX], int VectorY[DimMaxY])
  {
   for (ContY=0;ContY<DimY;ContY++)
    {
     for (ContX=0;ContX<DimX;ContX++)
      {
       MMin[ContX][ContY]= min(Alfa(VectorY[ContY],VectorX[ContX]),MMin[ContX][ContY]);
      }
    }
  }


void AlfaBeta::AprendeAsociacion(int VectorX[DimMaxX], int VectorY[DimMaxY])
  {
   for (ContY=0;ContY<DimY;ContY++)
    {
     for (ContX=0;ContX<DimX;ContX++)
      {
       MMin[ContX][ContY]= min(Alfa(VectorY[ContY],VectorX[ContX]),MMin[ContX][ContY]);
       MMax[ContX][ContY]=
max(Alfa(VectorY[ContY],VectorX[ContX]),MMax[ContX][ContY]);
      }
    }
  }

void AlfaBeta::RecuperaPatron(int VectorX[DimMaxX])
  {
   for (ContY=0;ContY<DimY;ContY++)
     {
      VectorSalidaMMax[ContY]=1;   //para la comparacion, necesciDimos el minimo, asi que lo
inicializamos en un maximo
      VectorSalidaMMin[ContY]=0;
```

```
      for (ContX=0;ContX<DimX;ContX++)
        {

VectorSalidaMMax[ContY]=min(Beta(MMax[ContX][ContY],VectorX[ContX]),VectorSalidaMM
ax[ContY]);

VectorSalidaMMin[ContY]=max(Beta(MMin[ContX][ContY],VectorX[ContX]),VectorSalidaMMi
n[ContY]);
        }
      }
   }

void AlfaBeta::RecuperaPatronMMin(int VectorX[DimMaxX])
   {
    for (ContY=0;ContY<DimY;ContY++)
      {
       VectorSalidaMMin[ContY]=0;
       for (ContX=0;ContX<DimX;ContX++)
        {

VectorSalidaMMin[ContY]=max(Beta(MMin[ContX][ContY],VectorX[ContX]),VectorSalidaMMi
n[ContY]);
        }
      }
   }

void AlfaBeta::RecuperaPatronMMax(int VectorX[DimMaxX])
   {
    for (ContY=0;ContY<DimY;ContY++)
      {
       VectorSalidaMMax[ContY]=1;   //para la comparacion, necesciDimos el minimo, asi que lo
inicializamos en un maximo
       for (ContX=0;ContX<DimX;ContX++)
        {

VectorSalidaMMax[ContY]=min(Beta(MMax[ContX][ContY],VectorX[ContX]),VectorSalidaMM
ax[ContY]);
        }
      }
   }

int AlfaBeta::Alfa(int DatoX, int DatoY)
   {
    int ResAlfa;
    if (DatoX==0)
      {
       if (DatoY==0) //(X=0, Y=0)
         ResAlfa=1;
       else         // (X=0,Y=1)
         ResAlfa=0;
      }
```

```
  else
    {
    if (DatoY==0) //(X=1, Y=0)
      ResAlfa=2;
    else         //(X=1,Y=1)
      ResAlfa=1;
    }
  return ResAlfa;
 }

int AlfaBeta::Beta(int DatoX, int DatoY)
  {
  int ResBeta;
  if (DatoX==0)
    ResBeta=0; //Beta(X=0,Y=0)=Beta(X=0,Y=1)=0
  else if (DatoX==1)
    ResBeta=DatoY; //Beta(X=1,Y=0)=0 Beta(X=1,Y=1)=1
  else
    ResBeta=1; //Beta(X=2,Y=0)=Beta(X=2,Y=1)=1
  return ResBeta;
  }

/*********************************************************************
  Algoritmos simplificados
*********************************************************************/

////////////Aprendizaje//////////////////////////////

void AlfaBeta::SeparaPatronX(int VectorX[DimMaxX])
  {
  DimX0s=0; DimX1s=0;
  //Vectores X
  for (ContX=0;ContX<DimX;ContX++)
   {
   if (VectorX[ContX]==0)
    {
    VectorX0s[DimX0s]=ContX;
    DimX0s++;
    VectorEX0s[ContX]=1;
    }
   else
    {
    VectorX1s[DimX1s]=ContX;
    DimX1s++;
    VectorEX1s[ContX]=1;
    }
   }
  }

void AlfaBeta::SeparaPatronY(int VectorY[DimMaxY])
  {
```

```
  DimY0s=0; DimY1s=0;
  // Vectores Y
  for (ContY=0;ContY<DimY;ContY++)
   {
    if (VectorY[ContY]==0)
     {
      VectorY0s[DimY0s]=ContY;
      DimY0s++;
      VectorEY0s[ContY]=1;
     }
    else
     {
      VectorY1s[DimY1s]=ContY;
      DimY1s++;
      VectorEY1s[ContY]=1;
     }
   }
 }

void AlfaBeta::SeparaPatrones(int VectorX[DimMaxX], int VectorY[DimMaxY])
 {
  DimX0s=0; DimX1s=0; DimY0s=0; DimY1s=0;
  //Vectores X
  for (ContX=0;ContX<DimX;ContX++)
   {
    if (VectorX[ContX]==0)
     {
      VectorX0s[DimX0s]=ContX;
      DimX0s++;
      VectorEX0s[ContX]=1;
     }
    else
     {
      VectorX1s[DimX1s]=ContX;
      DimX1s++;
      VectorEX1s[ContX]=1;
     }
   }

  // Vectores Y
  for (ContY=0;ContY<DimY;ContY++)
   {
    if (VectorY[ContY]==0)
     {
      VectorY0s[DimY0s]=ContY;
      DimY0s++;
      VectorEY0s[ContY]=1;
     }
    else
     {
      VectorY1s[DimY1s]=ContY;
```

```
      DimY1s++;
      VectorEY1s[ContY]=1;
     }
    }
  }

void AlfaBeta::AprendeMaxAbsMMax(int VectorX0s[DimMaxX],int VectorY1s[DimMaxY], int
DimX0s,int DimY1s)
  {
   //Memoria Max
   for (ContY=0;ContY<DimY1s;ContY++)
    {
     for (ContX=0;ContX<DimX0s;ContX++)
       {
        MMax [VectorX0s[ContX]][VectorY1s[ContY]]=2;
       }
     }
  }

void AlfaBeta::AprendeMinAbsMMin(int VectorX1s[DimMaxX], int VectorY0s[DimMaxY], int
DimX1s, int DimY0s)
  {
  // Mem Min
  for (ContY=0;ContY<DimY0s;ContY++)
    {
     for (ContX=0;ContX<DimX1s;ContX++)
       {
        MMin[VectorX1s[ContX]][VectorY0s[ContY]] =0;
       }
     }
 }

void AlfaBeta::AprendizajeRapido(int VectorX[DimMaxX], int VectorY[DimMaxY])
  {
   SeparaPatrones(VectorX,VectorY);
   AprendeMaxAbsMMax(VectorX0s, VectorY1s, DimX0s, DimY1s);
   AprendeMinAbsMMin(VectorX1s, VectorY0s,DimX1s, DimY0s);
   //   AprendeAbsMems(VectorX0s,VectorX1s, VectorY0s,VectorY1s, DimX0s, DimX1s,
DimY0s, DimY1s);
  }

void AlfaBeta::AprendizajeRapidoMMax(int VectorX[DimMaxX], int VectorY[DimMaxY])
  {
   SeparaPatrones(VectorX,VectorY);
   AprendeMaxAbsMMax(VectorX0s, VectorY1s, DimX0s, DimY1s);
  }

void AlfaBeta::AprendizajeRapidoMMin(int VectorX[DimMaxX], int VectorY[DimMaxY])
  {
   SeparaPatrones(VectorX,VectorY);
   AprendeMinAbsMMin(VectorX1s, VectorY0s,DimX1s, DimY0s);
```

```
 }

void AlfaBeta::AjustaValoresMemoria()
 {
  for (ContY=0;ContY<DimY;ContY++)
   {
    for (ContX=0;ContX<DimX;ContX++)
      {
       //Mem Min
       if ((VectorEY0s[ContY]==1 || VectorEX1s[ContX]==1) && MMin[ContX][ContY]!=0)
         {
          MMin[ContX][ContY]=1;
         }
       //Mem Max
       if ((VectorEY1s[ContY]==1 || VectorEX0s[ContX]==1) && MMax[ContX][ContY]!=2)
         {
          MMax[ContX][ContY]=1;
         }
      }
   }
 }

void AlfaBeta::ConvierteMMaxAMMin()
 {
  for (ContY=0;ContY<DimY;ContY++)
   {
    for (ContX=0;ContX<DimX;ContX++)
     {
      MMin[ContX][ContY]=2-MMax[ContY][ContX];
     }
   }
}

void AlfaBeta::ConvierteMMinAMMax()
 {
  for (ContY=0;ContY<DimY;ContY++)
   {
    for (ContX=0;ContX<DimX;ContX++)
     {
      MMax[ContY][ContX]=2-MMin[ContX][ContY];
     }
   }
}

//////////////////Recuperacion//////////////////////////

void AlfaBeta::SimplificaMemoria()
 {
  int ValorPos;
  for (ContY=0;ContY<DimY;ContY++)
   {
```

```
    EMMax0s[ContY]=0;// inicializamos los vectores de existencia
    EMMin2s[ContY]=0;//
    MemoriaMax1s[ContY][0]=0; // Inicializamos la dim del vector
    MemoriaMin1s[ContY][0]=0; // de numero de unos en 0
   }
  for (ContY=0;ContY<DimY;ContY++)
   {
    for (ContX=0;ContX<DimX;ContX++)
      {
       //Mem Max
       if (MMax[ContX][ContY]==0)
        {
          EMMax0s[ContY]=1;
        }
       else if (MMax[ContX][ContY]==1)
        {
         MemoriaMax1s[ContY][0]++;
         MemoriaMax1s[ContY][MemoriaMax1s[ContY][0]]=ContX;
        }
       //Mem Min
       if (MMin[ContX][ContY]==2)
        {
         EMMin2s[ContY]=1;
        }
       else if (MMin[ContX][ContY]==1)
        {
         MemoriaMin1s[ContY][0]++;
         MemoriaMin1s[ContY][MemoriaMin1s[ContY][0]]=ContX;
        }
      }
    }
  }

void AlfaBeta::RecuperacionModificada(int VectorX[DimMaxX])
  {
   int ContPosVectorX, Intersectan;
   int VectorX0s[DimMaxX],VectorX1s[DimMaxX];
   int DimX0s=0, DimX1s=0;
   //Vectores X
   for (ContX=0;ContX<DimX;ContX++)
    {
     if (VectorX[ContX]==0)
      {
       VectorX0s[DimX0s]=ContX;
       DimX0s++;
      }
     else
      {
       VectorX1s[DimX1s]=ContX;
       DimX1s++;
      }
```

```
  }

 //Recuperacion con Memoria Max
 for (ContY=0;ContY<DimY;ContY++)
  {
  if (EMMax0s[ContY]==1) //Si existe un 0
    {
     VectorSalidaMMax[ContY]=0;
    }
  else if (EMMax0s[ContY]==0 && MemoriaMax1s[ContY][0]==0)
    {  // si no existe 0s ni 1s
     VectorSalidaMMax[ContY]=1;
    }
  else
    {
     Intersectan=0;
     for (ContX=0;ContX<MemoriaMax1s[ContY][0];ContX++)
       {
        for (ContPosVectorX=0;ContPosVectorX < DimX0s;ContPosVectorX++)
          {
           if (MemoriaMax1s[ContY][ContX+1]==VectorX0s[ContPosVectorX])  //ContX+1 es por
que MemoriaMax1s[ContY][0] es el numero de unos
             {
              Intersectan=1;
              VectorSalidaMMax[ContY]=0;
              ContX=MemoriaMax1s[ContY][0]; // Para salirnos
              ContPosVectorX=DimX0s;    //para salirnos de este ciclo
             }
          } // Fin del for de las posiciones del vector de entrada
        }// fin del for de unos en el renglon ContY de la mem
      if (Intersectan==0)
        {
         VectorSalidaMMax[ContY]=1;
        }
    }// fin del else
 }//fin for  de posicion cont  y

 //Recuperacion con Memoria Min
 for (ContY=0;ContY<DimY;ContY++)
  {
  if (EMMin2s[ContY]==1)
    {
     VectorSalidaMMin[ContY]=1;
    }
  else if (EMMin2s[ContY]==0 && MemoriaMin1s[ContY][0]==0)
    { //no 2s ni unos --> todos son 0
     VectorSalidaMMin[ContY]=0;
    }
  else
    {
     Intersectan=0;
```

```
      for (ContX=0;ContX<MemoriaMin1s[ContY][0];ContX++)
        {
        for (ContPosVectorX=0;ContPosVectorX < DimX1s;ContPosVectorX++)
          {
          if (MemoriaMin1s[ContY][ContX+1]==VectorX1s[ContPosVectorX])
            {
            Intersectan=1;
            VectorSalidaMMin[ContY]=1;
            ContX=MemoriaMin1s[ContY][0]; // Para salirnos
            ContPosVectorX=DimX1s;            //para salirnos
            }
          } // Fin del for de las posiciones del vector de entrada
        }// fin del for de x en la mem
      if (Intersectan==0)
        {
        VectorSalidaMMin[ContY]=0;
        }
      }// fin del else
   }//fin for  de posicion conty
 }// fin recuperacionModificada

void AlfaBeta::RecuperacionModificadaMMin(int VectorX[DimMaxX])
 {
 int ContPosVectorX, Intersectan;
 int VectorX0s[DimMaxX],VectorX1s[DimMaxX];
 int DimX0s=0, DimX1s=0;
 //Vectores X
 for (ContX=0;ContX<DimX;ContX++)
  {
  if (VectorX[ContX]==0)
   {
   VectorX0s[DimX0s]=ContX;
   DimX0s++;
   }
  else
   {
   VectorX1s[DimX1s]=ContX;
   DimX1s++;
   }
  }

 //Recuperacion con Memoria Min
 for (ContY=0;ContY<DimY;ContY++)
  {
  if (EMMin2s[ContY]==1)
    {
    VectorSalidaMMin[ContY]=1;
    }
  else if (EMMin2s[ContY]==0 && MemoriaMin1s[ContY][0]==0)
    { //no 2s ni unos --> todos son 0
    VectorSalidaMMin[ContY]=0;
```

```
      }
     else
      {
       Intersectan=0;
       for (ContX=0;ContX<MemoriaMin1s[ContY][0];ContX++)
         {
          for (ContPosVectorX=0;ContPosVectorX < DimX1s;ContPosVectorX++)
            {
             if (MemoriaMin1s[ContY][ContX+1]==VectorX1s[ContPosVectorX])
               {
                Intersectan=1;
                VectorSalidaMMin[ContY]=1;
                ContX=MemoriaMin1s[ContY][0]; // Para salirnos
                ContPosVectorX=DimX1s;            //para salirnos
               }
            } // Fin del for de las posiciones del vector de entrada
         }// fin del for de x en la mem
       if (Intersectan==0)
         {
          VectorSalidaMMin[ContY]=0;
         }
      }// fin del else
    }//fin for  de posicion conty
  }// fin recuperacionModificada MMin

void AlfaBeta::RecuperacionModificadaMMax(int VectorX[DimMaxX])
  {
   int ContPosVectorX, Intersectan;
   int VectorX0s[DimMaxX],VectorX1s[DimMaxX];
   int DimX0s=0, DimX1s=0;
   //Vectores X
   for (ContX=0;ContX<DimX;ContX++)
    {
     if (VectorX[ContX]==0)
      {
       VectorX0s[DimX0s]=ContX;
       DimX0s++;
      }
     else
      {
       VectorX1s[DimX1s]=ContX;
       DimX1s++;
      }
    }

   //Recuperacion con Memoria Max
   for (ContY=0;ContY<DimY;ContY++)
    {
     if (EMMax0s[ContY]==1) //Si existe un 0
       {
        VectorSalidaMMax[ContY]=0;
```

```
      }
    else if (EMMax0s[ContY]==0 && MemoriaMax1s[ContY][0]==0)
      {  // si no existe 0s ni unos
       VectorSalidaMMax[ContY]=1;
      }
    else
      {
       Intersectan=0;
       for (ContX=0;ContX<MemoriaMax1s[ContY][0];ContX++)
         {
          for (ContPosVectorX=0;ContPosVectorX < DimX0s;ContPosVectorX++)
            {
             if (MemoriaMax1s[ContY][ContX+1]==VectorX0s[ContPosVectorX])
               {
                Intersectan=1;
                VectorSalidaMMax[ContY]=0;
                ContX=MemoriaMax1s[ContY][0]; // Para salirnos
                ContPosVectorX=DimX0s;    //para salirnos de este ciclo
               }
            } // Fin del for de las posiciones del vector de entrada
         }// fin del for de x en la mem
       if (Intersectan==0)
         {
          VectorSalidaMMax[ContY]=1;
         }
      }// fin del else
   }//fin for  de posicion cont  y
  }// fin recuperacionModificada MMax

int __fastcall AlfaBeta::min(int Valor1, int Valor2)
  {
   int Minimo;
   if (Valor1<=Valor2)
     Minimo=Valor1;
   else
     Minimo=Valor2;
   return Minimo;
  }

int __fastcall AlfaBeta::max(int Valor1, int Valor2)
  {
   int Maximo;
   if (Valor1>=Valor2)
     Maximo=Valor1;
   else
     Maximo=Valor2;
   return Maximo;
  }
```

El código anterior correspondía a la clase, ahora veremos como hacer una instancia de esa clase y como hacer uso de sus métodos.

```
MemAB=new AlfaBeta (DimX,DimY);  // Creación de una nueva memoria Alfa Beta
MemAB->DimX=Entero;   //Asignación de las dimensiones  de los patrones X eY
MemAB->DimY=Entero;  //  donde entero es un entero positivo
```

El siguiente código muestra la forma de aprender un conjunto fundamental, de una memoria asociativa Alfa Beta usando el algoritmo original, para eso usaremos el método aprendeasociación el cual recibe como parámetros 2 vectores que corresponden al patrón de entrada X y al patrón de salida Y

```
 for (Cont=0;Cont<TamConjFund;Cont++)
     MemAB->AprendeAsociacion(ListaPatronesEntrada[Cont],ListaPatronesSalida[Cont]);
```

El siguiente código nos permite aprender  un número (TamConjFund) determinado de asociaciones, usando el algoritmos simplificado, para hacer esto utilizaremos el método aprendizaje rápido, el cual recibe como parámetros 2 vectores que corresponden al patrón de entrada X y al patrón de salida Y. recordemos que al terminar debemos revisar todas las posiciones de las memorias para poner los valores que falta, esto lo hace la rutina de AjustaValoresMemoria

```
for (Cont=0;Cont<TamConjFund;Cont++)
   {
     MemAB->AprendizajeRapido(ListaPatronesEntrada[Cont],ListaPatronesSalida[Cont]);
    }
MemAB->AjustaValoresMemoria();
```

# Referencias


[1] Hassoun, M. H. (1993), *Associative Neural Memories*, Oxford University Press, New York.

[2] Kohonen, T. (1989), *Self-Organization and Associative Memory*, Springer-Verlag, Berlin.

[3] McCulloch, W. & Pitts, W. (1943), "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133.

[4] Ritter, G. X. & Sussner, P. (1996), "An Introduction to Morphological Neural Networks" in *Proceedings of the 13th International Conference on Pattern Recognition*, vol. IV, Track D, pp. 709-717.

[5] Rosenblatt, F. (1958), "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, vol. 65, no. 6, pp. 386-408.

[6] Widrow, B. & Lehr M. A. (1990), "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415-1441.

[7] Werbos, P. J. (1990), "Backpropagation Through Time: What It Does and How to Do It", *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560.

[8] Hopfield, J.J. (1982), "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554-2558.

[9] Minsky, M. & Papert, S. (1969), *Perceptrons*, MIT Press ,Cambridge.

[10] Steinbuch, K. (1961), "Die Lernmatrix", *Kybernetik*, vol. 1, no. 1, pp. 36-45.

[11] Willshaw, D., Buneman, O. & Longuet-Higgins, H. (1969), "Non-holographic associative memory", *Nature*, no. 222, pp. 960-962.

[12] Anderson, J. A. (1972), "A simple neural network generating an interactive memory", *Mathematical Biosciences*, vol. 14, pp. 197-220.

[13] Kohonen, T. (1972), "Correlation matrix memories", *IEEE Transactions on Computers*, *C-21*, vol. 4, pp. 353-359.

[14] Kosko, B. (1988), "Bidirectional associative memories", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49-60.

[15] Haines, K. & Hecht-Nielsen, R., (1988), "A BAM With Increased Information Storage Capacity", *IEEE International Conference on Neural Networks*, vol. 1, pp. 181-190

[16] Leung, C.S. & Cheung, K.F., (1991), "Householder encoding for discrete bidirectional associative memory**",** *IEEE International Joint Conference on Neural Networks*, Vol. 1, pp. 237-241.

[17] Wang, Y.-F., Cruz J.B., Jr. & Mulligan, Jr., (1991, "Guaranteed recall of all training pairs for bidirectional associative memory", *IEEE Transactions on Neural Networks*, Vol. 1, Issue 6, pp. 559-567

[18] Shen, D. & Cruz, J.B., Jr., 2005, "Encoding strategy for maximum noise tolerance bidirectional associative memory", *IEEE Transactions on Neural Networks*, Vol. 16, Issue 2, pp. 293-300

[19] Ritter, G. X., Sussner, P. & Diaz-de-Leon, J. L. (1998), "Morphological associative memories", *IEEE Transactions on Neural Networks*, vol. 9, pp. 281-293.

[20] Yáñez-Márquez, C. (2002), *Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios*, Tesis de Doctorado en Ciencias de la Computación, Centro de Investigación en Computación, México.

[21] Acevedo-Mosqueda, M.E. (2006), *Memorias Asociativas Bidireccionales Alfa-Beta*, Tesis de Doctorado en Ciencias de la Computación, Centro de Investigación en Computación, México.

[22] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C.**,** & López-Yáñez, I. (2006). *A New Model of BAM: Alpha-Beta Bidirectional Associative Memories*, Lecture Notes in Computer Science, LNCS 4263, Springer-Verlag Berlin Heidelberg pp. 286-295. ISSN: 0302-9743.

[23] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C. & López-Yáñez, I. (2006). *Alpha-Beta Bidirectional Associative Memories Based Translator*, IJCSNS International Journal of Computer Science and Network Security, Vol.6, No.5A, pp. 190-194. ISSN: 1738-7906.

[24] Yáñez-Márquez, C., Sánchez-Fernández, L. P. & López-Yáñez, I. (2006). *Alpha-Beta Associative Memories for Gray Level Patterns*, Lecture Notes in Computer Science, LNCS 3971, Springer-Verlag Berlin Heidelberg, pp. 818-823. ISSN: 0302-9743.

[25] Yáñez Márquez, C.**,** Díaz de León Santiago, J.L. & Salgado Ramírez, J.C. (2005). *Applying the New V-Alpha-Beta Associative Memories to Gray Level Images*, IT-209, Serie Azul, ISBN 970-36-0282-7, CIC-IPN, México.

[26] Yáñez Márquez, C.**,** Díaz de León Santiago, J.L. & Salgado Ramírez, J.C. (2005). *New V-Alpha-Beta Associative Memories able to Learn and Recall Patterns with Integer Components*, IT-210, Serie Azul, ISBN 970-36-0283-5, CIC-IPN, México.

[27] Sossa, H., Barrón, R. & Vázquez, R. (2004), "New Associative Memories to Recall Real-Valued Patterns", *CIARP*, LNCS 3287, pp. 195-202.
[28] Simpson, P. K. (1990), *Artificial Neural Systems*, Pergamon Press , New York.

[29] Steinbuch, K. & Frank, H. (1961), "Nichtdigitale Lernmatrizen als Perzeptoren", *Kybernetik*, vol. 1, no.3, pp. 117-124.

[30] Papadomanolakis, K., Kakarountas, A., Sklavos, N. & Goutis C. E., (2002), A Fast Johnson-Mobius Encoding Scheme for Fault Secure Binary Counters, *Proceedings of Design, Autamtions and Test in Europe*, (DATE '02), pp. 1-7.

[31] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001). *Lernmatrix de Steinbuch*, IT-48, Serie Verde, ISBN 970-18-6688-6, CIC-IPN, México

[32] Abu-Mostafa, Y. & St. Jacques, J. (1985), "Information capacity of the Hopfield model", *IEEE Transactions on Information Theory*, IT-31, no. 4, pp. 461-464.

[33] Austin, J. (1987), "ADAM: A Distributed Associative Memory for Scene Analysis", In *Proceedings of First International Conference on Neural Networks*, pp. 285-295.

[34] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001). *Memorias Morfológicas Autoasociativas*, IT-58, Serie Verde, ISBN 970-18-6698-3, CIC-IPN, México.

[35] Díaz de León Santiago, J.L. & Yáñez Márquez, C. (2001). *Memorias Morfológicas Heteroasociativas*, IT-57, Serie Verde, ISBN 970-18-6697-5, CIC-IPN, México.

[36] Anderson, J. A. & Rosenfeld, E. (1990). *Neurocomputing: Fundations of Research*, MIT Press ,Cambridge.

[37] Rosen, K., (1999), *Discrete Mathematics and Its Applications*, McGraw-Hill, Estados Unidos.

[38] Pajares G., & M. de la Cruz, J. (2001). *Visión por Compuador: Imágenes digitales y aplicaciones*, Alfaomega, España.

[39] González, R. & Woods, R. (1996). *Tratamiento digital de imágenes*, Addison-Wesley/Díaz de Santos, Wilmington.

[40] Pitas, I. (2000). *Digital Image Processing Algorithms and Applications*, J. Wiley Sons, Inc.

[41] Cruz Meza, M.E., Yáñez Márquez, C.**,** & Sánchez Garfias F.A. (2006). *Un nuevo modelo de memorias asociativas Alfa-Beta para imágenes en color*, IT-(en trámite), Serie Azul, ISBN (en trámite), CIC-IPN, México.

[42] Guzmán, Enrique, Pogrebnyak, Oleksiy & Yáñez-Márquez, Cornelio, (2005). *Compresión de Imágenes utilizando Memorias Asociativas Morfológicas*, en Proc. de la

Décimosexta Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición Industrial "La Robótica en la Era Digital", ROC&C'2005, IEEE Sección México, Acapulco, Guerrero, México, 29 de noviembre al 4 de diciembre de 2005.

[43] Guzmán, E., Pogrebnyak, O. & Yáñez-Márquez, C. (2006). *Image Compression Algorithm Based on Morphological Associative Memories*, Lecture Notes in Computer Science, LNCS 4225, Springer-Verlag Berlin Heidelberg pp. ISSN: 0302-9743.

[44] Yáñez-Márquez, C., Felipe-Riverón, E.M., López-Yáñez, I. & Flores-Carapia, R. (2006). *A Novel Approach to Automatic Color Matching*, Lecture Notes in Computer Science, LNCS 4225, Springer-Verlag Berlin Heidelberg, pp. 529-538. ISSN: 0302-9743.

[45] Montiel-Hernández, N.M. (2006), *Posicionamiento basado en redes inalámbricas para la determinación de funcionalidades de un GIS*, Tesis de Maestría en Ciencias de la Computación, Centro de Investigación en Computación, México.

[46] Godhavari, T., Alainelu, N. R. & Soundararajan, R. (2005). Cryptography Using Neural Network, IEEE Indicon 2005 Conference, Chennai, India, pp. 258-261

[47] Aldape-Perez, C. Yanez-Marquez, L.O. Lopez Leyva, "Feature Selection Using a Hybrid Associative Classifier with Masking Techniques," *micai* , pp. 151-160, 2006.

[48] Catalán-Salgado, C. Yanez-Marquez "Non Iterative Hopfield Model", CERMA 2006, 2006

[49] Theodoridis, Sergios Koutroumbas, Konstantinos "Pattern Recognition" Elsevier Academic Press 4th Edition 2003.

[50] Lippmann R.P. "An introduction to computing with neural networks," IEEE ASSP Magazine, Vol. 4(2), pp. 4-22, 1987.