INSTITUTO POLITECNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

REDES NEURONALES ALFA-BETA SIN PESOS: CONDICIONES SUFICIENTES PARA LA RECUPERACIÓN DE PATRONES

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN INGENIERÍA DE CÓMPUTO

PRESENTA:

ANTONIO ALARCÓN PAREDES

DIRECTORES DE TESIS:

DR. AMADEO JOSÉ ARGÜELLES CRUZ
DR. CORNELIO YÁÑEZ MÁRQUEZ

MÉXICO, D.F. JUNIO DE 2009

0262

SIP-14

INSTITUTO POLITECNICO NACIONAL

SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 5 del mes de Junio de 2009 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"REDES NEURONALES ALFA-BETA SIN PESOS: CONDICIONES SUFICIENTES PARA LA RECUPERACIÓN DE PATRONES"

ALARCÓN	**PAREDES**	**ANTONIO**
Apellido paterno	materno	nombre(s)

Con registro:

B	0	7	1	2	7	4

aspirante al grado de: ***MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Luis Pastor Sánchez Fernández

Secretario

Dr. Carlos Fernando Aguilar Ibáñez

Primer vocal
(Director de tesis)

Dr. Amadeo José Argüelles Cruz

Segundo vocal
(Director de tesis)

Dr. Cornelio Yáñez Márquez

Tercer vocal

Dr. Víctor Manuel Silva García

Suplente

M. en C. Pablo Manrique Ramírez

EL PRESIDENTE DEL COLEGIO

Dr. Jaime Álvarez Gallegos

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de *México, D.F.*, el día *11* del mes de *junio* del año *2009*, el (la) que suscribe *Antonio Alarcón Paredes,* alumno (a) del Programa de la *Maestría en Ciencias en Ingeniería de Cómputo con Opción en Sistemas Digitales,* con número de registro *B071274*, adscrito al *Centro de Investigación en Computación*, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección del *Dr. Amadeo José Argüelles Cruz* y el *Dr. Cornelio Yáñez Márquez*, y cede los derechos del trabajo intitulado *Redes neuronales Alfa-Beta sin pesos: condiciones suficientes para la recuperación de patrones*, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección *alpha2pi@hotmail.com*. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Antonio Alarcón Paredes

Nombre y firma

Resumen

En este trabajo de tesis se presenta una contribución tanto teórica como experimental para describir y caracterizar el funcionamiento de la red neuronal Alfa-Beta sin pesos, denominada CAINN.

Este modelo combina las características inherentes a las redes neuronales sin pesos con la teoría subyacente de las memorias asociativas $\alpha\beta$, logrando así un modelo competitivo.

Se introducen 9 definiciones, así como la demostración de 7 lemas y 10 teoremas que nos permiten construir la base para establecer las condiciones suficientes que exhiban y expliquen formalmente la recuperación del conjunto fundamental completo, así como para proponer un algoritmo alternativo cuyo número de operaciones y tiempo de procesamiento, es menor al necesitado para el funcionamiento del algoritmo original de CAINN.

Adicionalmente se realizan experimentos sobre recuperación y clasificación de patrones entre CAINN y ADAM utilizando bases de datos contenidas en el repositorio de la Universidad de California en Irvine (UCI Machine Learning Repository). Estos experimentos son comparados contra algunos otros modelos inmersos en la literatura.

Los resultados obtenidos permiten colocar a la red neuronal Alfa-Beta sin pesos como una buena alternativa para clasificación y reconocimiento de patrones.

Abstract

This work presents an experimental and theoretical contribution to describe and characterize the behavior of the Weightless Alpha-Beta Neural Network, denoted CAINN.

This model combines the inherent properties of the weightless neural networks with the underlying theory of the $\alpha\beta$ associative memories to obtain, thus, a competitive model.

Also, 9 definitions, and the proof of 7 lemmas and 10 theorems which let us build the sufficient conditions that formally exhibit and explain the correct recalling for the entire fundamental set are proposed. In addition, this formalism is used to propose an alternative algorithm that reduces the time and number of operations needed for the CAINN original algorithm.

Furthermore, there were made experiments based on pattern recognition and classification between CAINN and ADAM, where two datasets included in the UCI Machine Learning Repository were used. That latter experiments are also compared versus some other models in the literature.

The results here obtained, allow the Weightless Alpha-Beta Neural Network to be a good alternative for pattern recognition and classification.

AGRADECIMIENTOS

Debo agradecer de manera sincera a mis asesores: Dr. Cornelio Yáñez Márquez y Dr. Amadeo José Agüelles Cruz, por haber aceptado nuestro trabajo en conjunto para la realización de esta tesis; asimismo, agradezco su apoyo y entera confianza en mi trabajo. Porque con su ejemplo han creado las directrices no solo para el desarrollo de esta tesis, sino para mi desarrollo como investigador.

Agradezco especialmente a los integrantes del grupo Alfa-Beta y a mis demás compañeros y amigos del CIC, por la ayuda y consejos brindados para el presente trabajo de tesis, así como por los momentos de esparcimiento y diversión que han hecho más amena esta labor.

A la SIP y al CONACyT, ya que gracias al apoyo que brindan al desarrollo de la investigación, hemos podido llegar a la conclusión de este trabajo.

Al Instituto Politécnico Nacional y al Centro de Investigación en Computación, lugar en el cual pasé la mayoría de mi tiempo y que se ha convertido en mi segundo hogar.

El agradecimiento más profundo es para mi familia, ya que sin su apoyo incondicional y su inspiración habría sido imposible recorrer este camino. A mis padres, Leticia y Francisco, porque han sido un ejemplo de honestidad y lucha; a mi hermano, Diego, por su apoyo y por siempre compartir las páginas de nuestras vidas. A mis tías, primos, primas, sobrinos, sobrinas; ya que todos ellos han contribuido, en pequeña o gran medida, a que haya podido llegar a ser quien soy.

¡Por ellos y para ellos!

Índice general

1. Introducción **3**
- 1.1. Antecedentes . . . 3
- 1.2. Justificación . . . 5
- 1.3. Objetivo . . . 6
- 1.4. Resultados esperados . . . 6
- 1.5. Organización del documento . . . 6

2. Estado del Arte **7**
- 2.1. Memorias Asociativas . . . 7
  - 2.1.1. Conceptos Básicos . . . 7
  - 2.1.2. *Lernmatrix* de Steinbuch . . . 10
  - 2.1.3. *Correlograph* de Willshaw, Buneman y Longuet-Higgins . . . 11
  - 2.1.4. *Linear Associator* de Anderson-Kohonen . . . 11
  - 2.1.5. La memoria asociativa *Hopfield* . . . 12
  - 2.1.6. *Memoria Asociativa Bidireccional* (*BAM*) de Kosko . . . 14
  - 2.1.7. Memorias Asociativas *Morfológicas* . . . 15
  - 2.1.8. Memorias Asociativas *Alfa-Beta* . . . 17
- 2.2. Redes Neuronales sin Pesos . . . 17
  - 2.2.1. El nodo RAM . . . 17
  - 2.2.2. Red neuronal WISARD . . . 19
  - 2.2.3. ADAM . . . 20
  - 2.2.4. PLN . . . 22
  - 2.2.5. SDM . . . 24
  - 2.2.6. MPLN y pRAM . . . 25
  - 2.2.7. CAINN . . . 26

3. Materiales y Métodos **27**
- 3.1. Memorias Asociativas *Alfa-Beta* . . . 27
  - 3.1.1. Memorias heteroasociativas $\alpha\beta$ . . . 31
  - 3.1.2. Memorias heteroasociativas $\alpha\beta$ tipo $\bigvee$ . . . 31
  - 3.1.3. Memorias heteroasociativas $\alpha\beta$ tipo $\bigwedge$ . . . 33
  - 3.1.4. Memorias autoasociativas $\alpha\beta$ . . . 36
  - 3.1.5. Memorias autoasociativas $\alpha\beta$ tipo $\bigvee$ . . . 36
  - 3.1.6. Memorias autoasociativas $\alpha\beta$ tipo $\bigwedge$ . . . 39
- 3.2. CAINN . . . 42

3.2.1. Operación α_g 42
3.2.2. Operaciones σ_α y σ_β 42
3.2.3. Algoritmo de la red neuronal Alfa-Beta sin pesos 44
3.2.4. El funcionamiento de CAINN 46

4. Modelo propuesto **58**
4.1. Definiciones, Lemas y Teoremas 58
4.2. Optimización del algoritmo 70
4.3. Condiciones suficientes para la recuperación completa del conjunto fundamental 72
4.4. Ejemplo del fundamento de CAINN 73
4.5. Ejemplo del modelo CAINN Optimizado 85

5. Resultados y discusión **90**
5.1. Bases de datos y equipo utilizado 90
5.1.1. Iris Plant Data Set 90
5.1.2. Contraceptive Method Choice Data Set (CMC) 91
5.1.3. Equipo de cómputo utilizado 92
5.2. Recuperación de patrones 92
5.2.1. Recuperación del conjunto fundamental completo 92
5.2.2. Recuperación del conjunto fundamental por particiones 94
5.3. Clasificación de patrones 95
5.3.1. Clasificación del conjunto fundamental completo 95
5.3.2. Clasificación del conjunto fundamental por particiones 97

6. Conclusiones y trabajo a futuro **100**
6.1. Conclusiones 100
6.2. Trabajo a futuro 101

Capítulo 1

Introducción

El presente trabajo de tesis tiene como aportación principal la fundamentación teórica y formal para la recuperación de patrones del modelo de redes neuronales *Alfa-Beta* sin pesos denominado con el acrónimo *CAINN* (Computer Artificial Intelligence Neural Network), creado en 2007 en el CIC-IPN [1]. Este algoritmo combina la flexibilidad y adaptabilidad de las redes neuronales sin pesos, tomando provecho de la sencillez y robustez de los algoritmos de las memorias asociativas Alfa-Beta creadas también en el CIC-IPN en 2002 [2].

Adicionalmente, se presenta un algoritmo alternativo que permite al modelo CAINN reducir el tiempo requerido tanto para su fase de aprendizaje como para la de recuperación, así como disminuir el número de operaciones necesarias para estas fases.

1.1. Antecedentes

El cerebro humano continuamente recibe señales del entorno en que se desenvuelve cada individuo; dichas señales son procesadas para poder emitir una respuesta ante cada una de ellas. Nuestro cerebro posee millones de neuronas que se interconectan formando redes neuronales, que son las responsables de llevar a cabo dichos procesos. Las neuronas se comunican entre sí mediante un proceso llamado *sinapsis* en el cual una neurona envía un neurotransmisor a otra produciendo un efecto de excitación o de inhibición en esta última[3,4].

El desarrollo en la investigación de las Redes Neuronales Artificiales (ANN) ha sido motivado e inspirado para crear modelos matemáticos y computacionales que simulen el comportamiento de las redes neuronales biológicas [3]. La idea original del concepto de redes neuronales artificiales, surge como un intento para comprender y explicar el funcionamiento del cerebro humano.

La historia de las ANN comienza con la investigación realizada por los científicos Warren S. McCulloch y Walter Pitts, quienes en sus trabajos pioneros desarrollaron

el primer modelo de neurona artificial en 1943 [4]. Por su parte, el neuropsicólogo canadiense Donald Hebb propuso la primera idea para el aprendizaje artificial [5], cuya teoría afirmaba que dentro de nuestro cerebro, el aprendizaje está basado en la modificación de los pesos en las conexiones sinápticas de las neuronas.

Posteriormente, el psicólogo de la Universidad de Cornell, Frank Rosenblatt, desarrolló el *Perceptron* [6], que constituiría la primera máquina de aprendizaje automático, dando así lugar a que en 1960, Widrow y Hoff llegaran a la creación del modelo *ADALINE* (Adaptive Linear Element) que consistía en un Perceptron simple, aplicando la regla delta de aprendizaje [7]. Todo parecía ir viento en popa, pero fue en 1969 cuando los prestigiados científicos Marvin Minsky y Seymour Pappert del MIT publican su libro *Perceptrons* [8], en el cual evidencian de manera muy pesimista las limitaciones del Perceptron de Rosenblatt. Como consecuencia de este hecho, el desarrollo de las redes neuronales fue decayendo, a tal grado que muchos de los investigadores del área prefirieron enfocarse hacia otro campo.

A la par, investigadores como Steinbuch, Willshaw, Buneman & Longuett-Higgins, Anderson & Kohonen, Kaoru Nakano y Shun-Ichi Amari; realizaron grandes desarrollos en cuanto a las memorias asociativas, saliendo a palestra modelos tales como la *Lernmatrix* [9], el *Correlograph* [10], el *Linear Associator* [11,12], el *Associatron* [13] y las *Self-Organizing Nets of Threshold Elements* [14], convirtiéndose algunos de éstos en modelos clásicos; pero fue realmente el reconocido físico estadounidense John Hopfield quien, en 1982, publica un artículo en la prestigiada y respetada *National Academy of Sciences*, proponiendo un modelo capaz de funcionar como red neuronal y memoria asociativa a la vez [15], el cual marcaría un hito en el desarrollo de las memorias asociativas y terminaría con la época de oscurantismo de las redes neuronales, propiciando su resurgimiento. Dos años después publicaría otro artículo, en el cual presentó una extensión a su modelo original [16]. Así, Hopfield es el responsable de haber revivido el interés en las redes neuronales, después de un período de estancamiento de más de una década.

Las memorias asociativas tienen como propósito esencial recuperar patrones completos a partir de patrones de entrada, los cuales pueden presentar alteraciones. Por su naturaleza, los modelos de las memorias asociativas se dividen en dos fases: la fase de aprendizaje, que constituye la generación de la memoria a través de la asociación de patrones de entrada con un respectivo patrón de salida y la fase de recuperación, que es en donde la memoria asociativa opera para recuperar los patrones. Si en cada asociación en la fase de aprendizaje, ocurre que el patrón de entrada es igual al de salida, la memoria es *autoasociativa*; de otro modo, la memoria es *heteroasociativa* [2].

Entre los modelos de memorias asociativas, sólo existen dos de éstos que se desenvuelven dentro de una teoría distinta al modelo de neurona propuesto por McCulloch y Pitts. En primera instancia, podemos mencionar las *Memorias Asociativas Morfológicas* [17], cuyo funcionamiento es superior al de los modelos existentes en la fecha en que se crearon. Así mismo, se encuentran las *Memorias Asociativas Alfa-Beta* [2], desarrolladas en 2002, en el CIC-IPN. Este modelo se basa en los operadores Alfa y

Beta, creados ex profeso por el autor de ese modelo. Cabe mencionar que las memorias asociativas Alfa-Beta constituyen uno de los modelos con mejor rendimiento hasta nuestros días.

Paralelamente al desarrollo de las memorias asociativas, se desarrolló un nuevo paradigma dentro de las redes neuronales; las denominadas *Redes Neuronales sin Pesos* (WNN), cuya base se fundamenta en el uso de dispositivos de memoria RAM para llevar a cabo sus funciones. Este desarrollo surge con el trabajo de Bledsoe y Browning, quienes, a finales de la década de los 50, proponen un método binario para el reconocimiento de patrones, llamado *n-tupla*, que es concebido como una memoria distribuida que almacena información a través de la cual se derivan subpatrones que están relacionados con determinadas clases. [18]. Al realizar la clasificación de nuevos patrones, la clase de salida se define como aquella para la cual los patrones de aprendizaje son más comunes con el patrón presentado [19].

Este método propició el desarrollo de las WNN con investigadores como Wilkie, Stonham & Aleksander, Austin y Kanerva, quienes tuvieron a bien sacar a la luz algunos de los modelos deWNN, tales como la máquina de reconocimiento denominada *WISARD* (Wilkie, Stonham & Aleksander Recognition Device), método en el cual se implementaron las máquinas n-tupla utilizando dispositivos RAM convencionales [20]; el modelo *ADAM* (Advanced Distributed Associative Memory) [21], desarrollado con el fin de mejorar las redes asociativas de Willshaw [10]; la *SDM* (Sparse Distributed Memory) [21]; y recientemente el modelo CAINN (Computer Artificial Intelligence Neural Network) [1], del cual se deriva este trabajo de tesis.

Las redes neuronales sin pesos, como lo sugiere su nombre, no presentan pesos entre nodos. Las funciones de las neuronas son almacenadas en tablas de búsqueda, cuyo contenido debe ser actualizado para llevar a cabo el aprendizaje. Las WNN poseen algoritmos *one-shot* (no iterativos), a diferencia de las ANN cuyos algoritmos son iterativos, además de que los algoritmos de las WNNs son fácilmente adaptables como sistemas direccionables por contenido.

1.2. Justificación

La efectividad de los modelos de WNN aplicados a varios problemas reales ha sido evidenciada en un gran número de estudios experimentales [26-32]. Así mismo, se ha demostrado que el hecho de fusionar las WNN con la teoría formal de los algoritmos Alfa-Beta, ha aportado a la ciencia un modelo sencillo, flexible y eficaz para ser aplicado en la solución de una gran diversidad de problemas de reconocimiento de patrones presentes en la vida real. Sin embargo, el algoritmo del modelo CAINN de redes neuronales Alfa-Beta sin pesos aún no ha sido caracterizado. Es necesario realizar un estudio minucioso de este algoritmo para que, a partir de esto, pueda realizarse la formalización de las condiciones suficientes que expliquen la correcta recuperación de patrones.

1.3. Objetivo

En este trabajo de tesis se desarrollará una investigación teórica y experimental que nos permita encontrar las condiciones suficientes que muestren la recuperación correcta de patrones en el modelo CAINN; aunado a esto, se propondrá un algoritmo alternativo que reduzca la cantidad de tiempo y operaciones necesarias para que el algoritmo CAINN lleve a cabo sus funciones.

1.4. Resultados esperados

Con esta investigación, se espera la obtención de las condiciones suficientes que expliquen la correcta recuperación de patrones en el modelo CAINN; además, se pretende que de este modelo se desprenda un nuevo algoritmo, que sea capaz de aprender y recuperar patrones en una cantidad de tiempo menor sin perder ninguna de sus características.

1.5. Organización del documento

El documento está organizado como sigue:
En el Capítulo 1 se presenta una breve introducción a los antecedentes del presente trabajo de tesis, así como la justificación, los objetivos y los resultados que se pretenden obtener. El Capítulo 2 contiene algunos de los modelos más representativos de las redes neuronales sin pesos así como de las memorias asociativas.

Dentro del Capítulo 3, se incluye la explicación del modelo CAINN y de las memorias asociativas Alfa-Beta, cuya teoría es de vital importancia al ser éstos los principales pilares sobre los cuales se desprende el presente trabajo de tesis. Estos modelos nos dan la pauta a seguir para desarrollar el contenido del capítulo 4, en donde se incluyen varias definiciones, y de demuestran los lemas y teoremas necesarios para obtener los resultados esperados en esta investigación.

En el Capítulo 5 se incluye una serie de experimentos realizados para ejemplificar las demostraciones anteriores, realizando un análisis comparativo de recuperación y clasificación de patrones entre CAINN y el modelo ADAM, así como contra otros clasificadores dentro de la literatura. Seguido de esto, en el Capítulo 6 se muestran las conclusiones de la tesis y el trabajo a futuro. Finalmente, se presentan las referencias bibliográficas.

Capítulo 2

Estado del Arte

En este capítulo se hace una revisión de la literatura, ubicando los modelos más representativos en el ámbito de las memorias asociativas, así como también los principales modelos que existen dentro de las redes neuronales sin pesos.

2.1. Memorias Asociativas

En esta sección se presentarán algunos conceptos tomados de [2,52-55] y que son fundamentales para comprender el funcionamiento de las memorias asociativas. A su vez, se explicarán algunos modelos representativos de las memorias asociativas.

2.1.1. Conceptos Básicos

Por su naturaleza, el funcionamiento de una memoria asociativa se escinde en dos fases claramente distinguibles:

1. Fase de **aprendizaje** (generación de la memoria asociativa).
2. Fase de **recuperación** (operación de la memoria asociativa).

El propósito fundamental de una memoria asociativa es recuperar patrones completos a partir de patrones de entrada que pueden presentar alteraciones aditivas, sustractivas o mezcladas. De acuerdo con esta afirmación, una memoria asociativa **M** puede formularse como un sistema de entrada y salida, idea que se esquematiza a continuación:

(2.1)

Fase de aprendizaje Fase de recuperación

El patrón de entrada está representado por un vector columna denotado por $\mathbf{x}$ y el patrón de salida, por el vector columna denotado por $\mathbf{y}$. Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida. La notación para esta asociación es similar a la de una pareja ordenada; por ejemplo, los patrones $\mathbf{x}$ e $\mathbf{y}$ del esquema forman la asociación $(\mathbf{x}, \mathbf{y})$.

Para facilitar la manipulación algebraica de los patrones de entrada y de salida, los denotaremos con las mismas letras, $\mathbf{x}$ e $\mathbf{y}$, agregándoles números naturales como superíndices para efectos de discriminación simbólica. Por ejemplo, a un patrón de entrada $\mathbf{x}^1$ le corresponderá un 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 $\mathbf{M}$ se representa mediante una matriz generada a partir de un conjunto finito de asociaciones conocidas de antemano: éste es el *conjunto fundamental de asociaciones*, o simplemente *conjunto fundamental*. Se denota por p la cardinalidad del conjunto fundamental, donde $p \in \mathbb{Z}^+$.

Si μ es un índice, el conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \,|\mu = 1, 2, \ldots, p\} \tag{2.2}$$

A los patrones que conforman las asociaciones del conjunto fundamental, se les llama *patrones fundamentales*.

Es posible que los patrones fundamentales sean alterados con diferentes tipos de ruido. Para diferenciar un patrón alterado del correspondiente patrón fundamental, usaremos la tilde en la parte superior; así, el patrón $\widetilde{\mathbf{x}}^k$ es una versión alterada del patrón fundamental $\mathbf{x}^k$, y el tipo de alteración que representa $\widetilde{\mathbf{x}}^k$ se evidenciará en el contexto específico donde se use.

Con el fin de especificar las componentes de los patrones, se debe dejar en claro cuál será 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 $\mathbf{M}$ 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 con $n, m \in \mathbb{Z}^+$. Es decir:

$$\mathbf{x}^\mu \in A^n \text{ e } \mathbf{y}^\mu \in A^m, \qquad \forall \mu \in \{1, 2, \ldots, p\} \tag{2.3}$$

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

$$x_j^\mu \tag{2.4}$$

Definición 2.1 *Sea $\mathbf{x} \in A^n$ un patrón fundamental de entrada para una memoria asociativa $\alpha\beta$. El patrón $\mathbf{x}$ puede ser alterado, para dar lugar a un vector $\widetilde{\mathbf{x}}$, por tres tipos de alteraciones:*

1. Alteración *aditiva*, si $\mathbf{x} \leq \widetilde{\mathbf{x}}$. Esto significa que todos los posibles cambios en los valores de las coordenadas de $\mathbf{x}$ para obtener $\widetilde{\mathbf{x}}$ consisten en colocar un valor 1 donde había un valor 0.

2. Alteración *sustractiva*, si $\mathbf{x} \geq \widetilde{\mathbf{x}}$. Esto significa que todos los posibles cambios en los valores de las coordenadas de $\mathbf{x}$ para obtener $\widetilde{\mathbf{x}}$ consisten en colocar un valor 0 donde había un valor 1.

3. Alteración *combinada* o *mezclada*, si la alteración es una mezcla de aditiva con sustractiva. En este caso no es posible establecer un orden entre el patrón limpio y el alterado, dado que los valores podrán ser cambiados aleatoriamente, sin respetar necesariamente las reglas de los items 1 y 2.

La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas. Si se cumple que $\mathbf{x}^\mu = \mathbf{y}^\mu \ \forall \mu \in \{1, 2, \ldots, p\}$, se dice que la memoria es *autoasociativa*; por lo que uno de los requisitos que debe cumplir es que $n = m$. Una memoria es *heteroasociativa* si $\exists \mu = \{1, 2, \ldots, p\}$ para el que se cumple que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$. Nótese que pueden existir memorias heteroasociativas con $n = m$ [12].

Una vez habiendo descrito estos conceptos básicos y con la notación anterior, es posible expresar las dos fases de una memoria asociativa:

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

2. **Fase de Recuperación**: hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida $\mathbf{y}^\mu$, cuando se opera la memoria $\mathbf{M}$ 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 $\mathbf{M}$ exhibe recuperación correcta si al presentarle como entrada, en la fase de recuperación, un patrón $\mathbf{x}^\omega$ con $\omega \in \{1, 2, \ldots, p\}$, ésta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^\omega$. Teniendo todos estos conceptos claros, podemos continuar explicando algunos de los modelos de memorias asociativas más representativos.

2.1.2. *Lernmatrix* de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar* [2]. La importancia de la *Lernmatrix* [9,56] se evidencia en una afirmación que hace Kohonen [12] 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 $\mathbf{x}^\mu \in A^n$, $A = \{0, 1\}$ y produce como salida la clase $\mathbf{y}^\mu \in A^m$ que le corresponde (de entre p clases diferentes), codificada ésta con un método que en la literatura se le ha llamado *one-hot* [57], a saber: para representar la clase $k \in \{1, 2, \ldots, p\}$, se asignan a las componentes del vector de salida $\mathbf{y}^\mu$ los siguientes valores: $y_k^\mu = 1$, e $y_j^\mu = 0$ $\forall j = \{1, 2, \ldots, k-1, k+1, \ldots, p\}$.

Fase de aprendizaje

Se genera el esquema (crossbar) al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^p$. Cada uno de los componentes m_{ij} de $\mathbf{M}$, en la Lernmatrix de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } x_j^\mu = 1 = y_i^\mu \\ -\varepsilon & \text{si } x_j^\mu = 1 \text{ e } y_i^\mu = 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.5}$$

donde ε es una constante positiva elegida con antelación. Es usual que el valor de ε sea igual a 1.

Fase de recuperación

La fase de recuperación consiste en obtener las coordenadas del vector $\mathbf{y}^\omega \in A^p$ que le corresponde al patrón $\mathbf{x}^\omega$. La i-ésima coordenada y_i^ω del vector de clase $\mathbf{y}^\omega \in A^p$ se obtiene como lo indica la siguiente expresión, donde $\bigvee$ es el operador máximo:

$$y_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij} \cdot x_j^\omega = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} m_{ij} \cdot x_j^\omega \right] \\ 0 & \text{en otro caso} \end{cases} \tag{2.6}$$

2.1.3. *Correlograph* de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa. En palabras de los autores "el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental" [10,58].

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:

$$\Delta m_{ij} = \begin{cases} 1 & \text{si } y_i^\mu = 1 = x_j^\mu \\ 0 & \text{otro caso} \end{cases} \tag{2.7}$$

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:

$$y_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij} \cdot x_j^\omega > u \\ 0 & \text{en otro caso} \end{cases} \tag{2.8}$$

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 [10] de Willshaw et al. de 1969: $log2n$.

2.1.4. *Linear Associator* de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen [11,12,59].

Es curiosa la manera en cómo James A. Anderson y Teuvo Kohonen obtuvieron resultados asombrosamente similares a pesar de que trabajaron independientemente y sin tener noticia uno del otro, hasta tiempo después de que aparecieron los artículos; además, estos autores tienen formaciones profesionales totalmente diferentes: Anderson es neuropsicólogo y Kohonen es físico e ingeniero eléctrico.

Para presentar el Linear Associator consideremos de nuevo el conjunto fundamental: $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \,|\mu = 1, 2, \ldots, p\}$

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 $m \times n$.

2. Se suman las p matrices para obtener la memoria:

$$\mathbf{M} = \sum_{\mu=1}^{p} \mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t = [m_{ij}]_{m \times n} \tag{2.9}$$

de tal forma que la ij-ésima componente de la memoria $\mathbf{M}$ se expresa:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^\mu \; x_j^\mu \tag{2.10}$$

Fase de recuperación

Esta fase consiste en presentarle a la memoria un patrón de entrada $\mathbf{x}^\omega$, donde $\omega \in \{1, 2, \ldots, p\}$ y realizar la operación:

$$\mathbf{M} \cdot \mathbf{x}^\omega = \left[\sum_{\mu=1}^{p} \mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t \right] \cdot \mathbf{x}^\omega \tag{2.11}$$

2.1.5. La memoria asociativa *Hopfield*

El artículo de John J. Hopfield de 1982, publicado en los *proceedings* de la prestigiosa y respetada National Academy of Sciences, impactó positivamente y trajo a la palestra internacional su famoso modelo, capaz de funcionar como red neuronal y memoria asociativa a la vez [15].

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 $x_i = 0$ 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 xj entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona a sí misma, lo cual significa que $\forall i$, $m_{ii} = 0$. 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 $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \,|\mu = 1, 2, \ldots, p\}$ con $\mathbf{x}^\mu \in A^n$ y $A = \{-1, 1\}$.

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 $\mathbf{M}$:

$$m_{ij} = \begin{cases} \sum_{\mu=1}^{p} x_i^\mu \; x_j^\mu & \text{si } i \neq j \\ 0 & \text{si } i = j \end{cases} \tag{2.12}$$

Fase de recuperación

Si se le presenta un patrón de entrada $\widetilde{\mathbf{x}}$ 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 $\widetilde{\mathbf{x}}$, la fase de recuperación consta de tres pasos:

1. Para $t = 0$, se hace $x\,(t) = \widetilde{\mathbf{x}}$; es decir, $x_i\,(0) = \widetilde{x}_i$, $\forall i \in \{1, 2, \ldots, n\}$.
2. $\forall i \in \{1, 2, \ldots, n\}$ se calcula $x_i\,(t+1)$ de acuerdo con la condición siguiente:

$$x_i\,(t+1) = \begin{cases} 1 & \sum_{j=1}^{n} m_{ij} \; x_j\,(t) > 0 \\ x_i\,(t) & \sum_{j=1}^{n} m_{ij} \; x_j\,(t) = 0 \\ -1 & \sum_{j=1}^{n} m_{ij} \; x_j\,(t) < 0 \end{cases} \tag{2.13}$$

3 Se compara $x_i\,(t+1)$ con $x_i\,(t)$ $\forall i \in \{1, 2, \ldots, n\}$. Si $\mathbf{x}\,(t+1) = \mathbf{x}\,(t)$ el proceso termina y el vector recuperado es $\mathbf{x}\,(0) = \widetilde{\mathbf{x}}$. 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 $x_i\,(t+1) = x_i\,(\tau)$ $\forall i \in \{1, 2, \ldots, n\}$; 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 [60] se estableció formalmente que una cota superior para el número de vectores de estado arbitrarios estables en una memoria Hopfield es n.

2.1.6. *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 $(A_1, B_1), \ldots, (A_m, B_m)$, 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 [61], 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).

$$\begin{array}{rcl} A \rightarrow & \mathbf{M} & \rightarrow B \\ A\prime \leftarrow & \mathbf{M}^T & \leftarrow B \\ A\prime \rightarrow & \mathbf{M} & \rightarrow B\prime \\ A\prime\prime \leftarrow & \mathbf{M}^T & \leftarrow B\prime \\ & & \ldots \\ A_i \rightarrow & \mathbf{M} & \rightarrow B_i \\ A_i \leftarrow & \mathbf{M}^T & \leftarrow B_i \\ \ldots & & \end{array}$$

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:

$$\mathbf{M} = \sum_i A_i^T B_i \tag{2.14}$$

y la memoria dual $\mathbf{M}^T$ que está dada por:

$$\mathbf{M}^T = \sum_i \left(A_i^T B_i\right)^t = \sum_i B_i^T A_i \tag{2.15}$$

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:

$$A\mathbf{M}^j = \sum_i a_i \; m_{ij} \tag{2.16}$$

donde $\mathbf{M}^j$ es la j-ésima columna de $\mathbf{M}$. La suma de entrada para a_i es, de manera similar,

$$B\mathbf{M}_i^T = \sum_j b_j \; m_{ij} \tag{2.17}$$

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:

$$a_i = \begin{cases} 1 & \text{si } B\mathbf{M}_i^T > 0 \\ -1 & \text{si } B\mathbf{M}_i^T < 0 \end{cases}$$

$$b_i = \begin{cases} 1 & \text{si } A\mathbf{M}^j > 0 \\ -1 & \text{si } A\mathbf{M}^j < 0 \end{cases}$$

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).

2.1.7. 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 [17,62,63].

Hay dos tipos de memorias asociativas morfológicas: las memorias *max*, simbolizadas con $\mathbf{M}$, y las memorias *min*, cuyo símbolo es $\mathbf{W}$; 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 $\mathbf{D}$ y $\mathbf{H}$, denotado por $\mathbf{C} = \mathbf{D}\nabla\mathbf{H}$, es una matriz $[c_{ij}]_{m\times n}$ cuya ij-ésima componente c_{ij} es:

$$c_{ij} = \bigvee_{k=1}^{r} (d_{ik} + h_{kj}) \tag{2.18}$$

El *producto mínimo* entre $\mathbf{D}$ y $\mathbf{H}$, denotado por $\mathbf{C} = \mathbf{D} \Delta \mathbf{H}$, es una matriz $[c_{ij}]_{m \times n}$ cuya ij-ésima componente c_{ij} es:

$$c_{ij} = \bigwedge_{k=1}^{r} (d_{ik} + h_{kj}) \tag{2.19}$$

Las operaciones anteriores contienen a los operadores máximo $\bigvee$ y mínimo $\bigwedge$, los cuales están íntimamente ligados con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

Memorias heteroasociativas morfológicas

Algoritmo de las memorias *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 $\mathbf{y}^\mu \Delta \left(-\mathbf{x}^\mu\right)^t$ de dimensiones $m \times n$, donde el negado transpuesto del patrón de entrada $\mathbf{x}^\mu$ se define como $\left(-\mathbf{x}^\mu\right)^t = (-x_1^\mu, -x_2^\mu, \ldots, -x_n^\mu)$.

2. Se aplica el operador máximo $\bigvee$ a las p matrices para obtener la memoria $\mathbf{M}$.

$$\mathbf{M} = \bigvee_{\mu=1}^{p} \left[\mathbf{y}^\mu \Delta \left(-\mathbf{x}^\mu\right)^t\right] \tag{2.20}$$

Fase de recuperación Esta fase consiste en realizar el producto mínimo Δ de la memoria $\mathbf{M}$ con el patrón de entrada $\mathbf{x}^\omega$, donde $\omega \in \{1, 2, \ldots, p\}$, para obtener un vector columna $\mathbf{y}$ de dimensión m:

$$\mathbf{y} = \mathbf{M} \ \Delta \ \mathbf{x}^\omega \tag{2.21}$$

Las fases de aprendizaje y de recuperación de las memorias heteroasociativas *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:

$$\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \,|\mathbf{x}^\mu \in A_n, \text{ donde } \mu = 1, 2, \ldots, p\}$$

2.1.8. Memorias Asociativas *Alfa-Beta*

Las memorias asociativas Alfa-Beta [64] utilizan máximos y mínimos, y dos operaciones binarias inventadas exprofeso: α y β de las cuales heredan el nombre.

Debido a que los operadores α y β son base fundamental para el desarrollo del modelo CAINN, el fundamento teórico de las memorias asociativas *Alfa-Beta* se presenta en el siguiente capítulo de forma más completa.

2.2. Redes Neuronales sin Pesos

El desarrollo en la investigación de las Redes Neuronales Artificiales ha sido motivado, en primera instancia, por la intención de crear modelos matemáticos y computacionales que simulen con la mayor precisión posible el comportamiento de las redes neuronales biológicas [3].

La idea original del concepto de redes neuronales artificiales, surge como un intento para comprender y explicar el funcionamiento del cerebro humano. Dentro de las características propias de las redes neuronales artificiales (ANN), podemos mencionar la adaptabilidad, la habilidad de ser entrenadas mediante patrones de aprendizaje, y a partir de ellos, proporcionar una respuesta ante nuevos patrones de entrada, su velocidad y su robustez [22].

Las Redes Neuronales sin Pesos (WNN), como lo sugiere su nombre, no poseen pesos ajustables entre sus nodos. Estos modelos se caracterizan por presentar tanto entradas como salidas binarias; es decir, las neuronas sólo pueden tener dos estados: conectadas o no conectadas.

Las funciones que realiza cada neurona son almacenadas en tablas de búsqueda, que bien pueden ser implementadas al usar cualquier dispositivo de memoria RAM convencional. El aprendizaje en este tipo de redes neuronales se lleva a cabo al modificar los valores de las entradas en las tablas de búsqueda, con la posibilidad de ser implementadas en paralelo, lo que se traduce en algoritmos altamente flexibles y rápidos [23,24].

2.2.1. El nodo RAM

El nodo neuronal basado en RAM surge a partir del trabajo de Bledsoe & Browning, a finales de la década de los 50, cuya aportación radica en el desarrollo de un método para el reconocimiento de patrones denominado método de *n-tuplas* [18], que puede ser concebido como una memoria distribuida la cual almacena información, derivando subpatrones cuya función es relacionarse con determinadas clases.

Cuando se necesita clasificar patrones nuevos, la clase de salida es definida como aquella cuyos ejemplos de aprendizaje son más comunes con el nuevo patrón presentado [19].

Este método basa su funcionamiento en el principio de que el proceso de aprendizaje para reconocer un patrón puede verse como el conjunto de funciones lógicas que describan algún problema. Las funciones lógicas mencionadas, evaluarán como verdaderos a todos los patrones que pertenezcan a la clase que esté representada por dicha función, y como falso a todas las demás clases.

Para patrones desconocidos, el conjunto de funciones lógicas que tenga la mayoría de funciones evaluadas verdaderas, indicará la clase de dicho patrón [23].

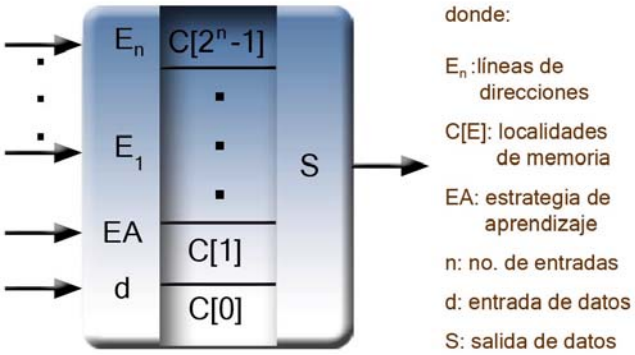

Figura 2.1: *El Nodo RAM*

De acuerdo con [23], en la *Fig. 2.1*, se describe un nodo neuronal basado en RAM con n entradas, el cual posee 2^n espacios de memoria, que serán direccionados por un vector de n bits denotado por $a = \{a^1, a^2, \ldots, a^n\}$, donde una señal binaria descrita por $E = \{E_1, E_2, \ldots, E_n\}$ en las líneas de entrada sólo tendrá acceso a una de las localidades del nodo. Es decir, una para la cual $a = E$. El bit $C[E]$, que fue almacenado en la fase de entrenamiento, activa la terminal EA y utiliza la salida S en la fase de recuperación.

No obstante que el nodo RAM *per se*, no es capaz de proporcionar generalización en la clasificación, sí existe la generalización dentro de las redes compuestas por nodos RAM; ésta puede ser introducida considerando la distancia de Hamming existente entre los patrones desconocidos y los patrones del conjunto de entrenamiento[25].

En las redes neuronales basadas en RAM, cada nodo RAM responde con un 1 solamente a aquellos patrones que se encuentren presentes dentro del conjunto de aprendizaje. De tal suerte que si se le presenta un patrón desconocido (que no está

contenido en la red), éste se clasificará en la misma clase del conjunto de aprendizaje en donde todas las salidas posibles de las RAM entreguen un 1 como resultado.

Esta arquitectura divide el conjunto de todos los posibles patrones en aquellos en los que puede darse la generalización y en los que no. Si se llegasen a requerir más de 2 clases, el camino a seguir será utilizar en conjunto varias redes basadas en RAM. Cada una de estas redes deberá ser entrenada para responder ante una clase de patrón [23].

2.2.2. Red neuronal WISARD

Para la década de los 70s, fue desarrollada la máquina denominada WISARD (Wilkie, Stonham & Aleksander Recognition Device) [20,33]. Este modelo implementó las máquinas n-tupla, y a su vez, utilizaba grupos de nodos RAM denominados discriminadores de clase, llamados así por el hecho de poder ser entrenados para elegir a la clase a la que pertenece cada una de las entradas. Estos discriminadores se conforman de una capa de k RAMs con n entradas cada una. Cada RAM almacena 2^n palabras de 1 bit. La estructura de WISARD está representada en la *Fig. 2.2.*

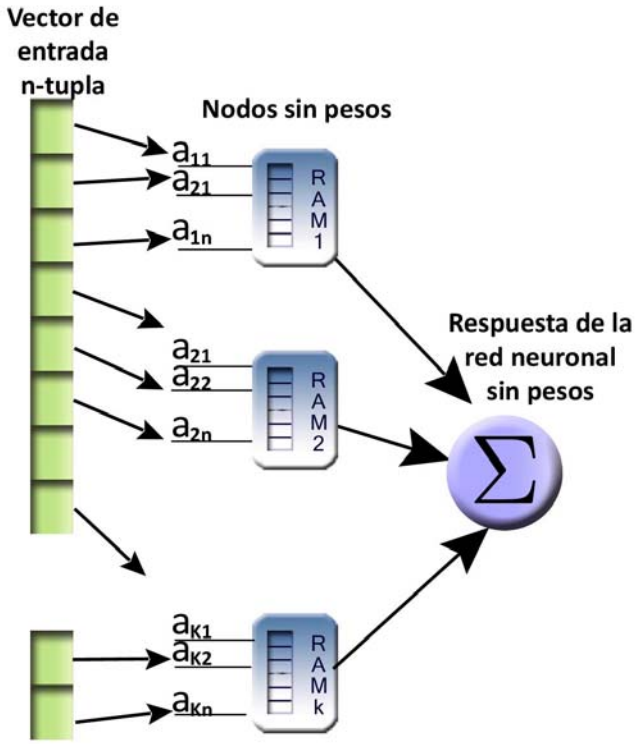

Figura 2.2: *Red neuronal WISARD*

Previo a la fase de aprendizaje, todas las $k2^n$ palabras deben igualarse a cero. Así, la fase de aprendizaje da inicio al presentarle al dispositivo n-tuplas seleccionadas de forma aleatoria desde un vector de valores binarios (vector de entrada).

Lo anterior se conoce como mapeo de entrada o patrón de conectividad; éste patrón representa un parámetro fijo de la red aún cuando se elije de forma aleatoria. Cada una de las salidas del discriminador está conectada a un dispositivo sumador, que toma las salidas de cada RAM y produce lo que se denomina la respuesta del discriminador.

El aprendizaje es llevado a cabo al presentarle al discriminador un patrón v con kn entradas, colocando en 1 todas las palabras a las que v tuvo acceso en los k nodos. Si el patrón v fuera presentado un tiempo después, tendrían que volverse a acceder todas las localidades que se escribieron previamente y cada uno de sus nodos responde con un 1.

Cuando se presenta una versión alterada u del patrón v, r actúa como una función del número de localidades previamente escritas a las que u accede, lo cual da una proporción de la similitud entre v y u. Así, el discriminador WISARD es capaz de clasificar patrones desconocidos de acuerdo con el conocimiento adquirido en esta fase.

El aprendizaje finaliza al presentar todos los vectores con las n-tuplas que definirán su clase representativa durante la fase de recuperación.

Al concluir el aprendizaje de forma individual para todas las clases, el reconocimiento de un patrón desconocido u se realiza mediante la presentación concurrente de u en todos los discriminadores y revisando cada respuesta r. El patrón se asigna a la clase contenida en el discriminador que presenta la respuesta r más alta.

La respuesta que n-tupla ofrece respecto del rendimiento depende principalmente de la elección del tamaño de n [34-37]. Resultados muestran que a mayor n la respuesta es mejor [35-38].

2.2.3. ADAM

Dentro de las redes basadas en RAM que utilizan métodos de n-tupla en memorias asociativas, podemos mencionar la arquitectura denominada ADAM (*Advanced Distributed Associative Memory*). Este modelo fue creado por Jim Austin en 1986, las consideraciones del diseño y un análisis detallado de su desempeño pueden ser encontrados en [42,43].

Con una estructura similar a la de la memoria asociativa no-holográfica de Willshaw [10], Austin desarrolló este modelo con el fin de mejorar las redes asociativas de Willshaw. En realidad, ADAM es un arreglo de dos redes asociativas, a las cuales les agregó un sistema de vectores denominado códigos de clase [44]. Es una red neuronal que se construye a partir de su componente principal denominada memoria de matriz de correlación (CMM), como se muestra en la *Fig. 2.3*.

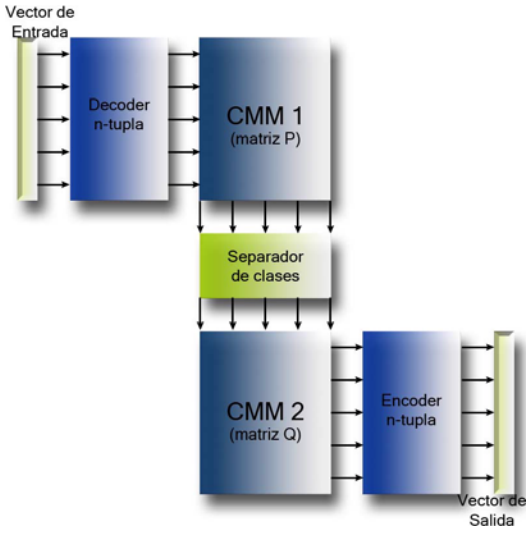

Figura 2.3: *Arquitectura de la red Neuronal ADAM.*

Para describir el modelo de manera analítica, considérese $A = \{0, 1\}$ y el conjunto fundamental $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \,|\mathbf{x}^\mu \in A^n, \mathbf{y}^\mu \in A^m$ donde $\mu = 1, 2, \ldots, p\}$

ADAM es un sistema de entrada y salida que opera teniendo como entrada un patrón binario $\mathbf{x}^\mu$ y produce como salida el patrón $\mathbf{y}^\mu$. En ambas fases, de aprendizaje y recuperación, para cada pareja de patrones de entrada y salida, se requiere un código de clase, el cual es un patrón $\mathbf{z}^\mu \in A^k$, donde $k \in \mathbb{Z}^+$; cada código de clase contiene r componentes con valor 1 y $k - r$ componentes con valor 0, donde $r \in \mathbb{Z}^+$.

Antes de llevar a cabo las fases de aprendizaje y recuperación, respectivamente, es preciso llevar a cabo dos pasos previos:

1. Elegir un valor $r < k$ para poder crear los códigos de clase.
2. Crear dos matrices nulas (llena de valores en 0): $\mathbf{P} = [p_{ij}]_{k\times n}$ y $\mathbf{Q} = [q_{ij}]_{m\times k}$

Fase de aprendizaje

Se introduce el patrón de entrada al dispositivo decodificador de funciones n-tupla; la salida de dicho decodificador se utiliza para entrenar a la primera CMM junto con el código de clase propuesto. Los componentes de la matriz $\mathbf{P}$ se actualizarán de acuerdo con la siguiente regla de aprendizaje:

$$p_{ij} = \begin{cases} 1 & \text{si } \; z_i = x_j \\ \text{valor anterior} & \text{otro caso} \end{cases} \tag{2.22}$$

El patrón a recordar se almacena en la segunda CMM y los componentes de la matriz $\mathbf{Q}$ serán actualizados de acuerdo con la siguiente regla:

$$q_{ij} = \begin{cases} 1 & \text{si } \; y_i = z_j \\ \text{valor anterior} & \text{otro caso} \end{cases} \tag{2.23}$$

Al final de la fase de entrenamiento del modelo ADAM, se obtienen las dos redes asociativas de Willshaw **P** y **Q** [45].

Fase de recuperación

Para realizar la fase de recuperación se considera un patrón de entrada $\widetilde{\mathbf{x}}^{\omega}$ donde $\omega \in \{1, 2, \ldots, p\}$ (si $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^{\omega}$ el patrón no ha sido alterado y pertenece al conjunto fundamental) y se lleva a cabo el siguiente procedimiento:

1. Se inicia sumando el número de asociaciones activas obtenidas en la fase previa, es decir:
$$(\mathbf{P} \cdot \widetilde{\mathbf{x}}^{\omega})_i = \sum_{j=1}^{n} p_{ij} \widetilde{x}_j^{\omega}, \qquad \forall i \in \{1, 2, \ldots, k\} \tag{2.24}$$

2 El valor de r será sustituido por el valor máximo de la suma de asociaciones activas de la matriz **P** en la primera CMM. Así recuperaremos, empleando el método de umbral Willshaw, el código de clase que será utilizado para recobrar el patrón contenido en la segunda red asociativa de Willshaw. De tal forma que:
$$z_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} p_{ij} \widetilde{x}_j^{\omega} = \bigvee_{h=1}^{k} \left[\sum_{j=1}^{n} p_{hj} \widetilde{x}_j^{\omega} \right] \\ 0 & \text{en otro caso} \end{cases} \tag{2.25}$$

3 Una vez conocido el código de clase $\mathbf{z}^{\omega}$, se procede a calcular el patrón de salida, así:
$$(\mathbf{Q} \cdot \mathbf{z}^{\omega})_i = \sum_{j=1}^{k} q_{ij} z_j^{\omega}, \qquad \forall i \in \{1, 2, \ldots, m\} \tag{2.26}$$

Se espera que el vector $\mathbf{Q} \cdot \mathbf{z}^{\omega}$ sea precisamente el patrón fundamental de salida $\mathbf{y}^{\omega}$.

2.2.4. PLN

El modelo de Neurona Lógica Probabilística (Probabilistic Logic Neuron, o PLN) fue desarrollado en 1987 por los investigadores Aleksander y Kan [39]. En este modelo, el nodo utiliza un número de dos bits alojado en una localidad de memoria. El contenido de las localidades de memoria se convierte en la probabilidad de disparo, es decir cuando se presenta un 1 a la salida del nodo [40]. En la *Fig.2. 4* se muestra la manera en la cual está constituida la red neuronal PLN.

Para n entradas binarias, cada dirección en el nodo PLN corresponde a una de las 2^n localidades. Los nodos simples de RAM entregan directamente a la salida el valor almacenado. Este valor almacenado en la localidad de memoria, es pasado a través de una función de salida, la cual será la encargada de convertir este contenido en una salida binaria del nodo. Dicho contenido puede ser 1, 0 o u, donde u se toma como un estado indefinido.

Cuando se accede a un estado u en la RAM, la salida de la RAM entregará un 0 o un 1 de manera equiprobable. Así cualquiera que sea el patrón de entrada, permitirá obtener un resultado a la salida. La respuesta dada por el nodo PLN se define a continuación:

$$r = \begin{cases} 0 & \text{si } C[I] = 0 \\ 1 & \text{si } C[I] = 1 \\ \text{random } (0,1) & \text{si } C[I] = u \end{cases} \tag{2.27}$$

Donde $C[I]$ indica el contenido de la dirección asociada con el patrón de entrada I y random$(0,1)$ es una función aleatoria que genera ceros y unos con la misma probabilidad.

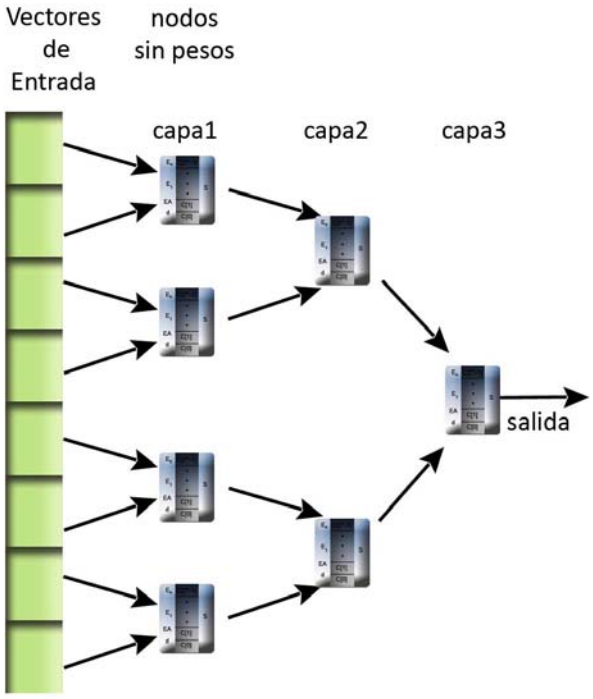

Figura 2.4: *Red neuronal PLN*

El hecho de usar el estado no definido para este tercer valor lógico u, hace posible el uso de estados desconocidos en la operación de redes neuronales sin pesos. Este valor es el estado que contienen todas las localidades de memoria, previo a realizarse la fase de aprendizaje, creando un estado de "*ignorancia*" en la red neuronal.

Al llevarse a cabo el aprendizaje, las localidades de memoria tienen un valor de u, con lo que se asegura que el comportamiento de la red sea imparcial. El proceso de aprendizaje en las redes PLN reemplaza los estados indeterminados u por *ceros* y *unos*; así, y de manera regular, es como la red produce patrones de salida correctos en respuesta a los patrones de aprendizaje en la entrada [1].

Las redes neuronales de arquitectura multicapa (denominadas pirámides), que utilizan el nodo PLN, sugeridas por Aleksander [41], presentan un número fijo de neuronas por capa, con un conjunto de conexiones de entrada (*fan-in*) y una conexión de salida (*fan-out*). Al utilizar los mismos patrones de entrada, se observó que la funcionalidad presente en las arquitecturas de pirámide es menor que la presentada por un nodo RAM [39].

2.2.5. SDM

La SDM (*Sparse Distributed Memory*) fue desarrollada por el finlandés Pentti Kanerva en 1988 [21] como un modelo matemático de la memoria humana a largo plazo.

La estructura básica del SDM consta de un gran número de direcciones de ubicación a las cuales se les asigna una dirección binaria. Cada dirección de ubicación es elegida aleatoriamente de un conjunto de 2^n posibles patrones del espacio de direcciones binarias con longitud n, de tal forma que cualquier decodificador o cualquier localidad H, pueda ser accedida desde cualquier lugar del espacio con la distancia de Hamming dentro de r_p bits desde su dirección de base.

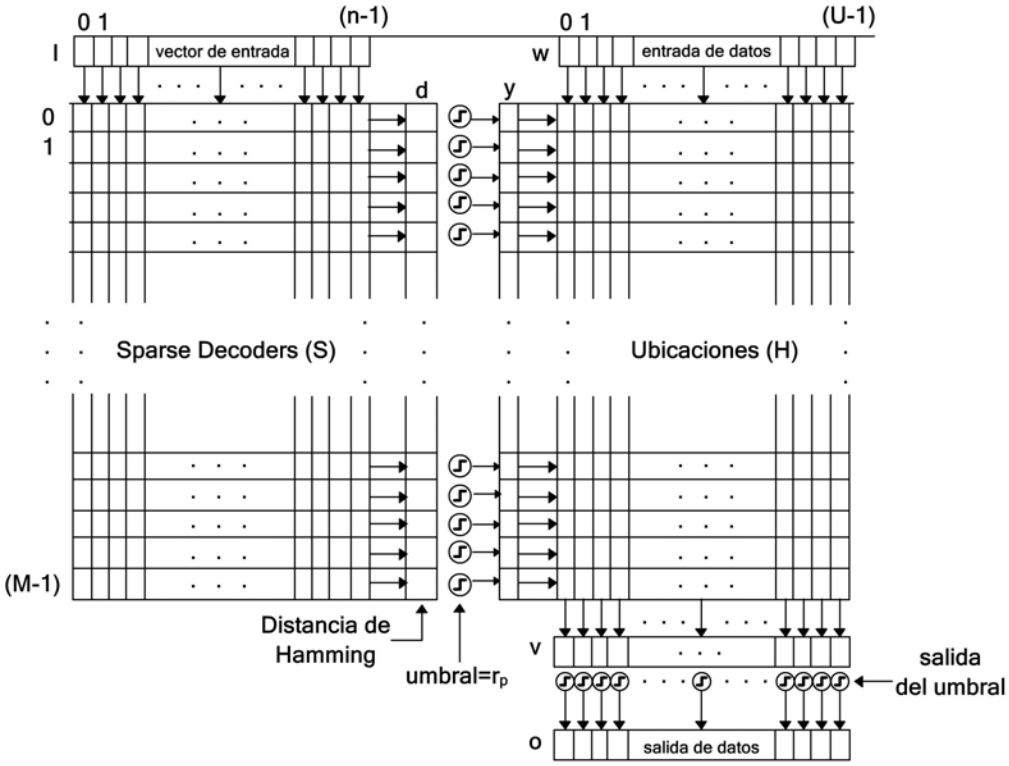

Figura 2.5: *Esquema de la SDM*

En la *Fig. 2.5* podemos observar un esquema del funcionamiento de la SDM, donde

el vector de entrada d es el portador de los resultados de las distancias calculadas entre el vector de entrada I y todas las direcciones base de los M decodificadores.

Un elemento de umbral entre los vectores d e y es el encargado de activar sus salidas si la distancia es de $0 \leq d \leq r_p$; esto se lleva a cabo asignándoles un 1 a las salidas. Así, solamente se hacen asociaciones entre las direcciones de ubicación accedidas por el vector I y los elementos activos del vector y. Estos elementos se almacenan en una matriz H cuya regla de actualización es $H = \sum y\,(\omega)^T$.

En la fase de recuperación, para recuperar un patrón almacenado dada una dirección del vector I, se lleva a cabo un proceso de umbralizado al vector de salida $v = yH^T$ [23].

2.2.6. MPLN y pRAM

La red PLN de m-estados o MPLN fue desarrollada por Myers & Aleksander en 1989 [46,47]. Este modelo, esquematizado en la *Fig. 2.6*, almacena un extenso rango de valores discretos en cada componente de memoria. La función de salida puede manejar funciones probabilísticas, lineales o sigmoidales. Cada una de las localidades del nodo puede almacenar las probabilidades de salida que están clasificadas.

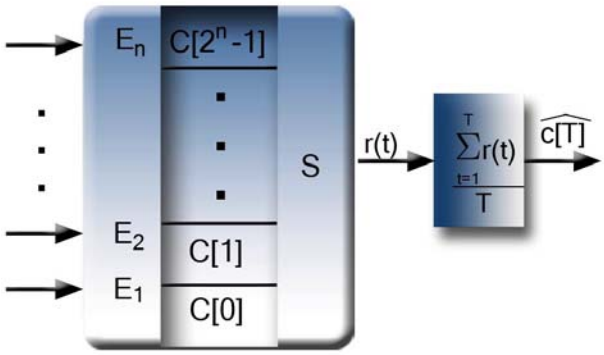

Figura 2.6: *Nodo pRAM con el proceso de cómputo a la salida.*

A su vez, Taylor propuso un modelo de neurona ruidosa [48] que incluía ecuaciones que incorporaban la formalización de muchas de las propiedades conocidas en las neuronas reales. La evolución de este modelo es equivalente a las redes RAM ruidosas o *pRAMs* (*probabilistic RAMs*).

El valor almacenado en $C[E]$ representa la probabilidad de que un pulso de corta duración y de amplitud 1 se produzca en la línea de salida S, dada la entrada E. El contenido de la localidad de memoria proporciona el valor promedio, es decir $y = C[E]$.

El modelo del nodo pRAM fue propuesto por Gorse & Taylor en 1989 [49], y al igual que las MPLN, las pRAMs también son una extensión del PLN, con la diferencia que en las pRAMs pueden almacenarse las probabilidades continuas, en el rango de valores de 0 y 1. Las pRAMs fueron extendidas en mayor medida, hasta ser capaces de mapear entradas continuas en salidas binarias. Como ejemplos de dicha extensión existen la denominada pRAM integradora (i-pRAM) [50] y otra unidad booleana denominada nodo cúbico [51].

2.2.7. CAINN

El objeto de estudio del presente trabajo de tesis es la formalización y análisis de este modelo, por lo que su funcionamiento se explicará a detalle en el siguiente capítulo.

Capítulo 3

Materiales y Métodos

3.1. Memorias Asociativas *Alfa-Beta*

Las memorias asociativas *Alfa-Beta* [2] utilizan *máximos* y *mínimos*, y dos operaciones binarias originales α y β de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β se deben especificar los conjuntos A y B, los cuales son:

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

La operación binaria $\alpha : A \times A \to B$ se define como:

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

Tabla 3.1 $\alpha : A \times A \to B$

La operación binaria $\beta : B \times A \to A$ 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

Tabla 3.2 $\beta : B \times A \to A$

Los conjuntos A y B, las operaciones binarias α y β, en conjunto con los operadores $\bigwedge$ (mínimo) y $\bigvee$ (máximo) usuales, conforman el sistema algebráico (A,B,α,β,$\bigwedge$,$\bigvee$) en el que se encuentran inmersas las memorias asociativas Alfa-Beta.

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

- Operación α máx: $P_{m\times r} \bigtriangledown_\alpha Q_{r\times n} = \left[f_{ij}^\alpha\right]_{m\times n}$, donde $f_{ij}^\alpha = \bigvee_{k=1}^{r} \alpha\left(p_{ik},q_{kj}\right)$
- Operación β máx: $P_{m\times r} \bigtriangledown_\beta Q_{r\times n} = \left[f_{ij}^\beta\right]_{m\times n}$, donde $f_{ij}^\beta = \bigvee_{k=1}^{r} \beta\left(p_{ik},q_{kj}\right)$
- Operación α mín: $P_{m\times r} \bigtriangleup_\alpha Q_{r\times n} = \left[h_{ij}^\alpha\right]_{m\times n}$, donde $h_{ij}^\alpha = \bigwedge_{k=1}^{r} \alpha\left(p_{ik},q_{kj}\right)$
- Operación β mín: $P_{m\times r} \bigtriangleup_\beta Q_{r\times n} = \left[h_{ij}^\beta\right]_{m\times n}$, donde $h_{ij}^\beta = \bigwedge_{k=1}^{r} \beta\left(p_{ik},q_{kj}\right)$

Nota *k es un entero positivo que puede tomar valores entre 1 y r inclusive.*

Restricciones:

- Ninguna de las cuatro operaciones está definida si $\exists j,k$ tales que $q_{kj} = 2$.
- Las operaciones $\bigtriangledown_\alpha$ y $\bigtriangleup_a$ no están definidas si $\exists i,j,k$ tales que $p_{ik} = 2$ o $q_{kj} = 2$.

Estas restricciones aparentan ser causa de potenciales problemas que podrían aparecer al usar las operaciones anteriores; sin embargo, las memorias asociativas están diseñadas de modo que nunca ocurra algún caso prohibido.

Lema 3.3 (Lema 3.9 de acuerdo con [2]). *Sea $\mathbf{x} \in A^n$ e $\mathbf{y} \in A^m$; entonces $\mathbf{y} \bigtriangledown_\alpha \mathbf{x}^t$ es una matriz de dimensiones $m \times n$, y además se cumple que:*

$$\mathbf{y} \bigtriangledown_\alpha \mathbf{x}^t = \mathbf{y} \bigtriangleup_\alpha \mathbf{x}^t$$

Demostración

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

$$= \begin{pmatrix} \bigvee_{k=1}^{1} \alpha(y_1, x_1) & \bigvee_{k=1}^{1} \alpha(y_1, x_2) \ldots & \bigvee_{k=1}^{1} \alpha(y_1, x_n) \\ \bigvee_{k=1}^{1} \alpha(y_2, x_1) & \bigvee_{k=1}^{1} \alpha(y_2, x_2) \ldots & \bigvee_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots \quad \ddots & \vdots \\ \bigvee_{k=1}^{1} \alpha(y_m, x_1) & \bigvee_{k=1}^{1} \alpha(y_m, x_2) \ldots & \bigvee_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{m \times n}$$

$$= \begin{pmatrix} \alpha(y_1, x_1) & \alpha(y_1, x_2) \ldots \alpha(y_1, x_n) \\ \alpha(y_2, x_1) & \alpha(y_2, x_2) \ldots \alpha(y_2, x_n) \\ \vdots & \vdots \quad \ddots \quad \vdots \\ \alpha(y_m, x_1) & \alpha(y_m, x_2) \ldots \alpha(y_m, x_n) \end{pmatrix}_{m \times n}$$

$$= \begin{pmatrix} \bigwedge_{k=1}^{1} \alpha(y_1, x_1) & \bigwedge_{k=1}^{1} \alpha(y_1, x_2) \ldots & \bigwedge_{k=1}^{1} \alpha(y_1, x_n) \\ \bigwedge_{k=1}^{1} \alpha(y_2, x_1) & \bigwedge_{k=1}^{1} \alpha(y_2, x_2) \ldots & \bigwedge_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots \quad \ddots & \vdots \\ \bigwedge_{k=1}^{1} \alpha(y_m, x_1) & \bigwedge_{k=1}^{1} \alpha(y_m, x_2) \ldots & \bigwedge_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{m \times n}$$

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

■

En efecto, resulta que $\mathbf{y} \bigtriangledown_\alpha \mathbf{x}^t$ es una matriz de dimensiones $m \times n$, y que $\mathbf{y} \bigtriangledown_\alpha \mathbf{x}^t = \mathbf{y} \triangle_\alpha \mathbf{x}^t$

Dado el resultado del lema anterior, es conveniente escoger un símbolo único, por ejemplo $\boxtimes$, que represente a las dos operaciones $\bigtriangledown_\alpha$ y $\triangle_\alpha$ cuando se opera un vector columna de dimensión m con un vector fila de dimensión n, de tal forma:

$$\mathbf{y} \bigtriangledown_\alpha \mathbf{x}^t = \mathbf{y} \boxtimes \mathbf{x}^t = \mathbf{y} \triangle_\alpha \mathbf{x}^t \tag{3.1}$$

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

$$\left[\mathbf{y} \boxtimes \mathbf{x}^t\right]_{ij} = \alpha(y_i, x_j) \tag{3.2}$$

Dado un índice de asociación μ, la expresión anterior indica que la $ij-$ésima componente de la matriz $\mathbf{y} \boxtimes \mathbf{x}^t$, se espresa de la siguiente manera:

$$\left[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t\right]_{ij} = \alpha\left(y_i^\mu, x_j^\mu\right) \tag{3.3}$$

Ahora se analizará el caso en el que se opera una matriz de dimensiones $m \times n$ con un vector columna de dimensión n utilizando las operaciones $\bigtriangledown_\alpha$ y $\bigtriangleup_\alpha$. En los lemas *3.11* y *3.12* de [2], se obtiene la forma que exhibirán las $i-$ésimas componentes de los vectores columna resultantes de dimensión m, a partir de ambas operaciones $\bigtriangledown_\alpha$ y $\bigtriangleup_\alpha$.

Lema 3.4 (Lema 3.11 de acuerdo con [2]). *Sean* $\mathbf{x} \in A^n$ y $\mathbf{P}$ una matriz de dimensiones $m \times n$. La operación $\mathbf{P}_{m\times n} \bigtriangledown_\beta \mathbf{x}$ da como resultado un vector columna de dimensión m, cuya $i-$ésima componente tiene la siguiente forma:

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

Demostración

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

$$= \begin{pmatrix} \beta\left(p_{11}, x_1\right) \bigvee \beta\left(p_{12}, x_2\right) \bigvee \ldots \bigvee \beta\left(p_{1n}, x_n\right) \\ \beta\left(p_{21}, x_1\right) \bigvee \beta\left(p_{22}, x_2\right) \bigvee \ldots \bigvee \beta\left(p_{2n}, x_n\right) \\ \vdots \qquad\qquad \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ \beta\left(p_{m1}, x_1\right) \bigvee \beta\left(p_{m2}, x_2\right) \bigvee \ldots \bigvee \beta\left(p_{mn}, x_n\right) \end{pmatrix} = \begin{pmatrix} \bigvee_{j=1}^{n} \beta\left(p_{1j}, x_j\right) \\ \bigvee_{j=1}^{n} \beta\left(p_{2j}, x_j\right) \\ \vdots \\ \bigvee_{j=1}^{n} \beta\left(p_{mj}, x_j\right) \end{pmatrix}$$

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

$$(\mathbf{P}_{m\times n} \bigtriangledown_\beta \mathbf{x})_i = \bigvee_{j=1}^{n} \beta\left(p_{ij}, x_j\right) \tag{3.4}$$

■

Lema 3.5 (Lema 3.12 de acuerdo con [2]). *Sean* $\mathbf{x} \in A^n$ y $\mathbf{P}$ una matriz de dimensiones $m \times n$. La operación $\mathbf{P}_{m\times n} \bigtriangleup_\beta \mathbf{x}$ da como resultado un vector columna de dimensión m, cuya $i-$ésima componente tiene la siguiente forma:

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

Demostración

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

$$= \begin{pmatrix} \beta(p_{11}, x_1) \bigwedge \beta(p_{12}, x_2) \bigwedge \cdots \bigwedge \beta(p_{1n}, x_n) \\ \beta(p_{21}, x_1) \bigwedge \beta(p_{22}, x_2) \bigwedge \cdots \bigwedge \beta(p_{2n}, x_n) \\ \vdots \qquad\qquad \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ \beta(p_{m1}, x_1) \bigwedge \beta(p_{m2}, x_2) \bigwedge \cdots \bigwedge \beta(p_{mn}, x_n) \end{pmatrix} = \begin{pmatrix} \bigwedge_{j=1}^{n} \beta(p_{1j}, x_j) \\ \bigwedge_{j=1}^{n} \beta(p_{2j}, x_j) \\ \vdots \\ \bigwedge_{j=1}^{n} \beta(p_{mj}, x_j) \end{pmatrix}$$

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

$$(\mathbf{P}_{m\times n} \triangle_\beta \mathbf{x})_i = \bigwedge_{j=1}^{n} \beta(p_{ij}, x_j) \tag{3.5}$$

■

3.1.1. Memorias heteroasociativas $\alpha\beta$

Se tienen dos tipos de memorias heteroasociativas $\alpha\beta$: tipo $\bigvee$ y tipo $\bigwedge$. Para la generación de ambos tipos de memorias se usará el operador $\boxtimes$, de la siguiente forma:

$$\left[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t\right]_{ij} = \alpha\left(y_i^\mu, x_j^\mu\right);\ \mu \in \{1, 2, \ldots, p\},\ i \in \{1, 2, \ldots, m\},\ j \in \{1, 2, \ldots, n\} \tag{3.6}$$

3.1.2. Memorias heteroasociativas $\alpha\beta$ tipo $\bigvee$

El algoritmo de las memorias heteroasociativas $\alpha\beta$ tipo $\bigvee$ se describe a continuación:

Fase de aprendizaje

La *fase de aprendizaje* consta de dos pasos; en el primero se utiliza el operador $\boxtimes$, mientras que en el segundo paso echaremos mano del operador máximo $\bigvee$.

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

$$\left[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t\right]_{m\times n} \tag{3.7}$$

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

$$\mathbf{V} = \bigvee_{\mu=1}^{p} \left[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t\right] \tag{3.8}$$

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

$$\nu_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right) \tag{3.9}$$

y de acuerdo con la definición de $\alpha : A \times A \longrightarrow B$, es posible observar que $\nu_{ij} \in B$, $\forall i \in \{1, 2, \ldots, m\},\ \forall j \in \{1, 2, \ldots, n\}$

Fase de recuperación

La *fase de recuperación* tiene dos posibles casos. En el primer caso, el patrón de entrada $\mathbf{x}^\omega$ es un patrón fundamental, es decir $\omega \in \{1, 2, \ldots, p\}$. En el segundo caso, el patrón de entrada no es un patrón fundamental, sino que se entiende como una versión distorsionada de al menos uno de los patrones del conjunto fundamental. Es decir, para el patrón alterado $\widetilde{\mathbf{x}}$, debe existir por lo menos un valor de índice $\omega \in \{1, 2, \ldots, p\}$ correspondiente al conjunto fundamental, para el cual $\widetilde{\mathbf{x}}$ es una versión con alguno de los tres tipos de alteración considerados en la *Definición 2.1.*

Caso 1: Patrón fundamental Se presenta un patrón $\mathbf{x}^\omega$, con $\omega \in \{1, 2, \ldots, p\}$, a la memoria heteroasociativa $\alpha\beta$ tipo $\bigvee$ y se realiza la operación $\triangle_\beta$:

$$\mathbf{V} \triangle_\beta \mathbf{x}^\omega \tag{3.10}$$

Dado que las dimensiones de la matriz $\mathbf{V}$ son $m \times n$ 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 posible obtener como sigue:

$$\left(\mathbf{V} \triangle_\beta \mathbf{x}^\omega\right)_i = \bigwedge_{j=1}^{n} \beta\left(\nu_{ij}, x_j^\omega\right) \tag{3.11}$$

$$\left(\mathbf{V} \triangle_\beta \mathbf{x}^\omega\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right)\right], x_j^\omega\right\} \tag{3.12}$$

Teorema 3.6 (Teorema 4.7 de acuerdo con [2]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ representada por $\mathbf{V}$. *Si* ω *es un valor de índice arbitrario tal que* $\omega \in \{1, 2, \ldots, p\}$, *y si además para cada* $i \in \{1, 2, \ldots, m\}$ *se cumple que* $\exists j = j_0 \in \{1, 2, \ldots, n\}$, *el cual depende de* ω *y de* i, *tal que* $\nu_{ij_0} = \alpha\left(y_i^\omega, x_{j_0}^\omega\right)$, *entonces la recuperación* $\mathbf{V} \triangle_\beta \mathbf{x}^\omega$ *es correcta, en términos de que* $\mathbf{V} \triangle_\beta \mathbf{x}^\omega = \mathbf{y}^\omega$.

Demostración Ver detalles de la demostración en [2].

Caso 2: Patrón alterado Se presenta un patrón binario $\widetilde{\mathbf{x}}$ que es un vector columna de dimensión n, a la memoria heteroasociativa $\alpha\beta$ tipo $\bigvee$ y se realiza la operación $\triangle_\beta$:

$$\mathbf{V} \triangle_\beta \widetilde{\mathbf{x}} \tag{3.13}$$

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

$$(\mathbf{V} \triangle_\beta \widetilde{\mathbf{x}})_i = \bigwedge_{j=1}^{n} \beta\left(\nu_{ij}, \widetilde{\mathbf{x}}_j\right) \tag{3.14}$$

$$(\mathbf{V} \triangle_\beta \widetilde{\mathbf{x}})_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right)\right], \widetilde{x}_j\right\} \tag{3.15}$$

Teorema 3.7 (Teorema 4.13 de acuerdo con [2]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ representada por $\mathbf{V}$, *y sea* $\widetilde{\mathbf{x}} \in A^n$ *un patrón con alteración aditiva respecto de algún patrón fundamental* $\mathbf{x}^\omega$ *con* $\omega \in \{1, 2, \ldots, p\}$. *Si se presenta* $\widetilde{\mathbf{x}}$ *a la memoria* $\mathbf{V}$ *como entrada, y si además para cada* $i \in \{1, 2, \ldots, m\}$ *se cumple que* $\exists j = j_0 \in \{1, 2, \ldots, n\}$, *el cual depende de* ω *y de* i, *tal que* $\nu_{ij_0} \leq \alpha\left(y_i^\omega, \widetilde{x}_{j_0}\right)$, *entonces la recuperación* $\mathbf{V} \triangle_\beta \mathbf{x}^\omega$ *es correcta, en términos de que* $\mathbf{V} \triangle_\beta \widetilde{\mathbf{x}} = \mathbf{y}^\omega$.

Demostración Ver detalles de la demostración en [2].

3.1.3. Memorias heteroasociativas $\alpha\beta$ tipo $\bigwedge$

El algoritmo de las memorias heteroasociativas $\alpha\beta$ tipo $\bigwedge$ se describe a continuación:

Fase de aprendizaje

La *fase de aprendizaje* consta de dos pasos; en el primero se utiliza el operador $\boxtimes$, mientras que en el segundo paso echaremos mano del operador mínimo $\bigwedge$.

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

$$\left[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t\right]_{m\times n} \tag{3.16}$$

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

$$\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p} \left[\mathbf{y}^\mu \boxtimes (\mathbf{x}^\mu)^t\right] \tag{3.17}$$

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

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(y_i^\mu, x_j^\mu\right) \tag{3.18}$$

y de acuerdo con la definición de $\alpha : A \times A \longrightarrow B$, es posible observar que $\lambda_{ij} \in B$, $\forall i \in \{1, 2, \ldots, m\},\ \forall j \in \{1, 2, \ldots, n\}$

Fase de recuperación

Al igual que con las memorias heteroasociativas $\alpha\beta$ tipo $\bigvee$, la *fase de recuperación* para las memorias heteroasociativas $\alpha\beta$ tipo $\bigwedge$ tiene también dos posibles casos.En el primer caso, el patrón de entrada $\mathbf{x}^\omega$ es un patrón fundamental, es decir $\omega \in \{1, 2, \ldots, p\}$. En el segundo caso, el patrón de entrada no es un patrón fundamental, sino que se entiende como una versión distorsionada de al menos uno de los patrones del conjunto fundamental. Es decir, para el patrón alterado $\widetilde{\mathbf{x}}$, debe existir por lo menos un valor de índice $\omega \in \{1, 2, \ldots, p\}$ correspondiente al conjunto fundamental, para el cual $\widetilde{\mathbf{x}}$ es una versión distorsionada con alguno de los tres tipos de alteración considerados en la *Definición 2.1.*

Caso 1: Patrón fundamental Se presenta un patrón $\mathbf{x}^\omega$, con $\omega \in \{1, 2, \ldots, p\}$, a la memoria heteroasociativa $\alpha\beta$ tipo $\bigwedge$ y se realiza la operación $\bigtriangledown_\beta$:

$$\mathbf{\Lambda} \bigtriangledown_\beta \mathbf{x}^\omega \tag{3.19}$$

De manera análoga a lo que sucede con las memorias heteroasociativas $\alpha\beta$ tipo $\bigvee$, las dimensiones de la matriz $\mathbf{\Lambda}$ son $m \times n$ y $\mathbf{x}^\omega$ es un vector columna de dimensión n; por lo cual, el resultado de la operación anterior debe ser un vector columna de dimensión m, cuya $i-$ésima componente es posible obtener como sigue:

$$\left(\mathbf{\Lambda} \bigtriangledown_\beta \mathbf{x}^\omega\right)_i = \bigvee_{j=1}^{n} \beta\left(\lambda_{ij}, x_j^\omega\right) \tag{3.20}$$

$$(\mathbf{\Lambda} \bigtriangledown_\beta \mathbf{x}^\omega)_i = \bigvee_{j=1}^{n} \beta \left\{ \left[\bigwedge_{\mu=1}^{p} \alpha \left(y_i^\mu, x_j^\mu \right) \right], x_j^\omega \right\} \tag{3.21}$$

Teorema 3.8 (Teorema 4.20 de acuerdo con [2]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ *representada por* $\mathbf{\Lambda}$. *Si* ω *es un valor arbitrario fijo tal que* $\omega \in \{1, 2, \ldots, p\}$, *y si además para cada* $i \in \{1, 2, \ldots, m\}$ *se cumple que* $\exists j = j_0 \in \{1, 2, \ldots, n\}$, *el cual depende de* ω *y de* i, *tal que* $\lambda_{ij_0} = \alpha \left(y_i^\omega, x_{j_0}^\omega \right)$, *entonces la recuperación* $\mathbf{\Lambda} \bigtriangledown_\beta \mathbf{x}^\omega$ *es correcta; es decir que* $\mathbf{\Lambda} \bigtriangledown_\beta \mathbf{x}^\omega = \mathbf{y}^\omega$.

Demostración Ver detalles de la demostración en [2].

Caso 2: Patrón alterado Se presenta un patrón binario $\widetilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^\omega$) que es un vector columna de dimensión n, a la memoria heteroasociativa $\alpha\beta$ tipo $\bigwedge$ y se realiza la operación $\bigtriangledown_\beta$:

$$\mathbf{\Lambda} \bigtriangledown_\beta \mathbf{x}^\omega \tag{3.22}$$

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

$$(\mathbf{\Lambda} \bigtriangledown_\beta \widetilde{\mathbf{x}})_i = \bigvee_{j=1}^{n} \beta \left(\lambda_{ij}, \widetilde{x}_j \right) \tag{3.23}$$

$$(\mathbf{\Lambda} \bigtriangledown_\beta \widetilde{\mathbf{x}})_i = \bigvee_{j=1}^{n} \beta \left\{ \left[\bigwedge_{\mu=1}^{p} \alpha \left(y_i^\mu, x_j^\mu \right) \right], \widetilde{x}_j \right\} \tag{3.24}$$

Teorema 3.9 (Teorema 4.23 de acuerdo con [2]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ *representada por* $\mathbf{\Lambda}$, *y sea* $\widetilde{\mathbf{x}} \in A^n$ *un patrón con alteración sustractiva respecto de algún patrón fundamental* $\mathbf{x}^\omega$ *con* $\omega \in \{1, 2, \ldots, p\}$. *Si se presenta* $\widetilde{\mathbf{x}}$ *a la memoria* $\mathbf{\Lambda}$ *como entrada, y si además para cada* $i \in \{1, 2, \ldots, m\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, 2, \ldots, n\}$, *el cual depende de* ω *y de* i, *tal que* $\lambda_{ij_0} \geq \alpha(y_i^\omega, \widetilde{x}_{j_0})$, *entonces la recuperación* $\mathbf{\Lambda} \bigtriangledown_\beta \widetilde{\mathbf{x}}$ *es correcta; es decir que* $\mathbf{\Lambda} \bigtriangledown_\beta \widetilde{\mathbf{x}} = \mathbf{y}^\omega$.

Demostración Ver detalles de la demostración en [2].

3.1.4. Memorias autoasociativas $\alpha\beta$

Si a una memoria heteroasociativa se le impone la condición de que $\mathbf{y}^\mu = \mathbf{x}^\mu \ \forall \mu \in \{1, 2, \ldots, p\}$, entonces deja de ser heteroasociativa y se le denomina *autoasociativa*. A continuación se muestra una lista de algunas de las características de las memorias autoasociativas $\alpha\beta$

1. El conjunto fundamental toma la forma $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, \ldots, p\}$
2. Los patrones fundamentales de entrada y salida son de la misma dimensión, denotada por n.
3. La memoria es una matriz cuadrada, para ambos tipos $\bigvee$ y $\bigwedge$. Si $\mathbf{x}^\mu \in A^n$, entonces $\mathbf{V} = [\nu_{ij}]_{n\times n}$ y $\Lambda = [\lambda_{ij}]_{n\times n}$

3.1.5. Memorias autoasociativas $\alpha\beta$ tipo $\bigvee$

El algoritmo de las memorias autoasociativas $\alpha\beta$ tipo $\bigvee$ se describe a continuación:

Fase de aprendizaje

La *fase de aprendizaje*, al igual que las memorias heteroasociativas $\alpha\beta$ anteriormente descritas, consta también de dos pasos, a saber:

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

$$\left[\mathbf{x}^\mu \boxtimes (\mathbf{x}^\mu)^t\right]_{n\times n} \tag{3.25}$$

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

$$\mathbf{V} = \bigvee_{\mu=1}^{p} \left[\mathbf{x}^\mu \boxtimes (\mathbf{x}^\mu)^t\right] \tag{3.26}$$

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

$$\nu_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_j^\mu\right) \tag{3.27}$$

y de acuerdo con la definición de $\alpha : A \times A \longrightarrow B$, es posible observar que $\nu_{ij} \in B$, $\forall i \in \{1, 2, \ldots, n\}, \ \forall j \in \{1, 2, \ldots, n\}$

Fase de recuperación

La *fase de recuperación* de las memorias autoasociativas $\alpha\beta$ tipo $\bigvee$ tiene dos posibles casos, similarmente a como sucede con las memorias heteroasociativas $\alpha\beta$. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, \ldots, p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino una versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\widetilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, \ldots, p\}$, que corresponde al patrón fundamental respecto del cual $\widetilde{\mathbf{x}}$ es una versión distorsionada con alguno de los tres tipos de alteración considerados en la *Definición 2.1.*

Caso 1: Patrón fundamental Se presenta un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, \ldots, p\}$, a la memoria autoasociativa $\alpha\beta$ tipo $\bigvee$ y se realiza la operación $\triangle_{\beta}$:

$$\mathbf{V} \triangle_{\beta} \mathbf{x}^{\omega} \tag{3.28}$$

El resultado de la operación anterior será un vector columna de dimensión n, cuya $i-$ésima componente es posible obtener como sigue:

$$\left(\mathbf{V} \triangle_{\beta} \mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta\left(\nu_{ij}, x_j^{\omega}\right) \tag{3.29}$$

$$\left(\mathbf{V} \triangle_{\beta} \mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)\right], x_j^{\omega}\right\} \tag{3.30}$$

Lema 3.10 (Lema 4.27 de acuerdo con [2]). *Una memoria autoasociativa $\alpha\beta$ tipo $\bigvee$ tiene únicamente unos en su diagonal principal.*

Demostración La $ij-$ésima entrada de una memoria autoasociativa $\alpha\beta$ tipo $\bigvee$ está dada por $\nu_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$ de acuerdo con la fase de aprendizaje. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$, de tal forma que:

$$\nu_{ii} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_i^{\mu}\right), \ \forall i \in \{1, 2, \ldots, n\} \tag{3.31}$$

Debido a la propiedad de isoargumentos en α, de acuerdo con la definición de α dada en la *Tabla 3.1*, se tiene que $\alpha\left(x_i^{\mu}, x_i^{\mu}\right) = 1$, por lo que la expresión anterior, se transforma en:

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

■

Teorema 3.11 (Teorema 4.28 de acuerdo con [2]). *Una memoria autoasociativa $\alpha\beta$ tipo $\bigvee$ recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje..*

Demostración Sea $\omega \in \{1, 2, \ldots, p\}$ arbitrario. De acuerdo con el *Lema 3.10*, para cada $i \in \{1, 2, \ldots, n\}$ escogida arbitrariamente:

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

Es decir, para $i \in \{1, 2, \ldots, n\}$ escogida arbitrariamente, $\exists j_0 = i \in \{1, 2, \ldots, n\}$ que cumple con:

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

Por lo tanto, de acuerdo con el *Teorema 3.6*

$$\mathbf{V} \triangle_{\beta} \mathbf{x}^{\omega} = \mathbf{x}^{\omega}, \quad \forall \omega \in \{1, 2, \ldots, p\}$$

Esto significa que la memoria autoasociativa $\alpha\beta$ tipo $\bigvee$ recuperará de manera correcta el conjunto fundamental completo. Además, en la demostración de este Teorema, no aparece restricción alguna sobre p, el cual es la cardinalidad del conjunto fundamental; esto quiere decir que el conjunto fundamental puede crecer tanto como se desee. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa $\alpha\beta$ tipo $\bigvee$, con recuperación correcta, es máximo.

■

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

$$\mathbf{V} \triangle_{\beta} \widetilde{\mathbf{x}} \tag{3.32}$$

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:

$$\left(\mathbf{V} \triangle_{\beta} \widetilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left(\nu_{ij}, \widetilde{x}_j\right) \tag{3.33}$$

$$\left(\mathbf{V} \triangle_{\beta} \widetilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)\right], \widetilde{x}_j\right\} \tag{3.34}$$

Teorema 3.12 (Teorema 4.30 de acuerdo con [2]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu \in 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria autoasociativa* $\alpha\beta$ *representada por* $\mathbf{V}$*, y sea* $\widetilde{\mathbf{x}} \in A^n$ *un patrón con alteración aditiva respecto de algún patrón fundamental* $\mathbf{x}^\omega$ *con* $\omega \in \{1, 2, \ldots, p\}$*. Si se presenta* $\widetilde{\mathbf{x}}$ *a la memoria* $\mathbf{V}$ *como entrada, y si además para cada* $i \in \{1, 2, \ldots, n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, 2, \ldots, n\}$*, el cual depende de* ω *y de* i*, tal que* $\nu_{ij_0} \leq \alpha\left(x_i^\omega, \widetilde{x}_{j_0}\right)$*, entonces la recuperación* $\mathbf{V} \triangle_\beta \widetilde{\mathbf{x}} = \mathbf{x}^\omega$.

Demostración Por hipótesis se tiene que $\mathbf{y}^\mu = \mathbf{x}^\mu \ \forall \mu \in \{1, 2, \ldots, p\}$ y, por ende $m = n$. Al establecer estas dos condiciones en el Teorema 3.7, se obtiene el resultado: $\mathbf{V} \triangle_\beta \widetilde{\mathbf{x}} = \mathbf{x}^\omega$

■

3.1.6. Memorias autoasociativas $\alpha\beta$ tipo $\bigwedge$

Las fases de *aprendizaje* y *recuperación* son similares a las ya discutidas para las memorias heteroasociativas $\alpha\beta$.

Fase de aprendizaje

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

$$\left[\mathbf{x}^\mu \boxtimes (\mathbf{x}^\mu)^t\right]_{n \times n} \tag{3.35}$$

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

$$\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p} \left[\mathbf{x}^\mu \boxtimes (\mathbf{x}^\mu)^t\right] \tag{3.36}$$

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

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^\mu, x_j^\mu\right) \tag{3.37}$$

y de acuerdo con la definición de $\alpha : A \times A \longrightarrow B$, es posible observar que $\lambda_{ij} \in B$, $\forall i \in \{1, 2, \ldots, n\}, \ \forall j \in \{1, 2, \ldots, n\}$

Fase de recuperación

La fase de recuperación de las memorias autoasociativas $\alpha\beta$ tipo $\bigwedge$ tiene dos posibles casos, similarmente a como sucede con las memorias heteroasociativas $\alpha\beta$; en el primer caso el patrón de entrada es un patrón fundamental; es decir la entrada es un patrón $\mathbf{x}^\omega$, con $\omega \in \{1, 2, \ldots, p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino una versión distorsionada de por lo menos uno de los

patrones fundamentales; esto significa que si el patrón de entrada es $\widetilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, \ldots, p\}$, que corresponde al patrón fundamental respecto del cual $\widetilde{\mathbf{x}}$ es una versión distorsionada con alguno de los tres tipos de alteración considerados en la *Definición 2.1.*

Caso 1: Patrón fundamental Se presenta un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, \ldots, p\}$, a la memoria autoasociativa $\alpha\beta$ tipo $\bigwedge$ y se realiza la operación $\bigtriangledown_{\beta}$:

$$\mathbf{\Lambda} \bigtriangledown_{\beta} \mathbf{x}^{\omega} \tag{3.38}$$

El resultado de la operación anterior será un vector columna de dimensión n, cuya $i-$ésima componente es posible obtener como sigue:

$$\left(\mathbf{\Lambda} \bigtriangledown_{\beta} \mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left(\lambda_{ij}, x_j^{\omega}\right) \tag{3.39}$$

$$\left(\mathbf{\Lambda} \bigtriangledown_{\beta} \mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left\{\left[\bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)\right], x_j^{\omega}\right\} \tag{3.40}$$

Lema 3.13 (Lema 4.31 de acuerdo con [2]). *Una memoria autoasociativa $\alpha\beta$ tipo $\bigwedge$ tiene únicamente unos en su diagonal principal.*

Demostración La $ij-$ésima entrada de una memoria autoasociativa $\alpha\beta$ tipo $\bigwedge$ está dada por $\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$ de acuerdo con la fase de aprendizaje. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$, de tal forma que:

$$\lambda_{ii} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_i^{\mu}\right), \ \forall i \in \{1, 2, \ldots, n\} \tag{3.41}$$

Debido a la propiedad de isoargumentos en α, de acuerdo con la definición de α dada en la *Tabla 3.1*, se tiene que $\alpha\left(x_i^{\mu}, x_i^{\mu}\right) = 1$, por lo que la expresión anterior, se transforma en:

$$\lambda_{ii} = \bigwedge_{\mu=1}^{p} (1) = 1, \ \forall i \in \{1, 2, \ldots, n\}$$

■

Teorema 3.14 (Teorema 4.32 de acuerdo con [2]). *Una memoria autoasociativa $\alpha\beta$ tipo $\bigwedge$ recupera de manera correcta el conjunto fundamental completo; además tiene máxima capacidad de aprendizaje.*

Demostración Sea $\omega \in \{1, 2, \ldots, p\}$ arbitrario. De acuerdo con el *Lema 3.13*, para cada $i \in \{1, 2, \ldots, n\}$ escogida arbitrariamente:

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

Es decir, para $i \in \{1, 2, \ldots, n\}$ escogida arbitrariamente, $\exists j_0 = i \in \{1, 2, \ldots, n\}$ que cumple con:

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

Por lo tanto, de acuerdo con el *Teorema 3.8*

$$\mathbf{\Lambda} \bigtriangledown_{\beta} \mathbf{x}^{\omega} = \mathbf{x}^{\omega}, \quad \forall \omega \in \{1, 2, \ldots, p\}$$

Esto significa que la memoria autoasociativa $\alpha\beta$ tipo $\bigwedge$ recupera de manera correcta el conjunto fundamental completo. Además, en la demostración de este Teorema, no aparece restricción alguna sobre p, el cual es la cardinalidad del conjunto fundamental; esto quiere decir que el conjunto fundamental puede crecer tanto como se desee. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa $\alpha\beta$ tipo $\bigwedge$, con recuperación correcta, es máximo.

■

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

$$\mathbf{\Lambda} \bigtriangledown_{\beta} \widetilde{\mathbf{x}} \tag{3.42}$$

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:

$$\left(\mathbf{\Lambda} \bigtriangledown_{\beta} \widetilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta\left(\lambda_{ij}, \widetilde{x}_j\right) \tag{3.43}$$

$$\left(\mathbf{\Lambda} \bigtriangledown_{\beta} \widetilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta\left\{\left[\bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)\right], \widetilde{x}_j\right\} \tag{3.44}$$

Teorema 3.15 (Teorema 4.33 de acuerdo con [2]). *Sea* $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mu \in 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria autoasociativa* $\alpha\beta$ *representada por* $\mathbf{\Lambda}$*, y sea* $\widetilde{\mathbf{x}} \in A^n$ *un patrón con alteración sustractiva respecto de algún patrón fundamental* $\mathbf{x}^{\omega}$ *con* $\omega \in \{1, 2, \ldots, p\}$*. Si se presenta* $\widetilde{\mathbf{x}}$ *a la memoria* $\mathbf{\Lambda}$ *como entrada, y si además para cada* $i \in \{1, 2, \ldots, n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, 2, \ldots, n\}$*, el cual depende de* ω *y de* i*, tal que* $\lambda_{ij_0} \geq \alpha\left(x_i^{\omega}, \widetilde{x}_{j_0}\right)$*, entonces la recuperación* $\mathbf{\Lambda} \bigtriangledown_{\beta} \widetilde{\mathbf{x}} = \mathbf{x}^{\omega}$*.*

Demostración Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}\ \forall \mu \in \{1, 2, \ldots, p\}$ y, por ende $m = n$. Al establecer estas dos condiciones en el Teorema 3.9, se obtiene el resultado: $\mathbf{\Lambda} \bigtriangledown_{\beta} \widetilde{\mathbf{x}} = \mathbf{x}^{\omega}$

■

3.2. CAINN

A fin de describir el algoritmo utilizado por CAINN, es preciso presentar las tres operaciones originales: α_g, σ_α y σ_β.

3.2.1. Operación α_g

Para la caracterización del modelo usando las operaciones Alfa y Beta, se requiere la definición de una nueva operación, a la que se le denominará: *Alfa Generalizada* y que denotaremos con el símbolo α_g. Esta nueva operación, como su nombre lo indica, es una generalización de la operación binaria *Alfa* original, de tal forma que pueda aceptar valores reales como argumentos, entregando a la salida valores del conjunto $B = \{0, 1, 2\}$, es decir, $\alpha_g = \mathbb{R} \times \mathbb{R} \longrightarrow B$; y se define de la siguiente manera:

$$\alpha_g = \begin{cases} 0 & \text{si } x < y \\ 1 & \text{si } x = y \\ 2 & \text{si } x > y \end{cases} \tag{3.45}$$

Por ejemplo, si $a = 5{,}3,\ b = 15,\ c = -2{,}9,\ d = 5{,}3$, se obtendrían los siguientes resultados:

- $\alpha_g(a, b) = 0$
- $\alpha_g(a, c) = 2$
- $\alpha_g(a, d) = 1$
- $\alpha_g(b, c) = 2$
- $\alpha_g(c, d) = 0$

3.2.2. Operaciones σ_α y σ_β

Estas dos operaciones se definen y ejemplifican de la siguiente manera:

- Operación *Sigma-Alfa*: $P_{m\times r}\sigma_\alpha Q_{r\times n} = \left| f_{ij}^{\alpha} \right|_{m\times n}$, donde $f_{ij}^{\alpha} = \sum_{k=1}^{r} \alpha(p_{ik}, q_{kj})$

- Operación *Sigma-Beta*: $P_{m\times r}\sigma_\beta Q_{r\times n} = \left|f_{ij}^{\beta}\right|_{m\times n}$, donde $f_{ij}^{\beta} = \sum_{k=1}^{r} \beta\left(p_{ik}, q_{kj}\right)$

Dados los vectores columna $\mathbf{x} \in A^n$ e $\mathbf{y} \in A^m$, con el conjunto $A = \{0, 1\}$, $\mathbf{y}\ \sigma_\alpha\ \mathbf{x}^t$ es una matriz de dimensiones $m \times n$. La $ij-$ésima componente de la matriz $\mathbf{y}\sigma_\alpha\mathbf{x}^t$ está dada por:

$$\left(\mathbf{y}\sigma_\alpha\mathbf{x}^t\right)_{ij} = \alpha\left(y_i, x_j\right) \tag{3.46}$$

A continuación se presenta un caso en el que se opera una matriz de dimensiones $m \times n$ con un vector de columna de dimensión n usando la operación σ_α.

La operación $\mathbf{P}_{m\times n}\sigma_\alpha\mathbf{x}$ da como resultado un vector columna de dimensión m, cuya $i-$ésima componente tiene la siguiente forma:

$$\left(\mathbf{P}_{m\times n}\sigma_\alpha\mathbf{x}\right)_i = \sum_{j=1}^{n} \alpha\left(p_{ij}, x_j\right) \tag{3.47}$$

Como ejemplo de los resultados para la operación $\mathbf{P}_{m\times n}\sigma_\alpha\mathbf{x}$, tenemos:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \text{ y } \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \text{ entonces } \mathbf{P}\sigma_\alpha\mathbf{x} = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix}$$

Dados los vectores columna $\mathbf{x} \in A^n$ e $\mathbf{y} \in A^m$, con el conjunto $A = \{0, 1\}$, $\mathbf{y}\sigma_\beta\mathbf{x}^t$ es una matriz de dimensiones $m \times n$. La $ij-$ésima componente de la matriz $\mathbf{y}\sigma_\beta\mathbf{x}^t$ está dada por:

$$\left(\mathbf{y}\sigma_\beta\mathbf{x}^t\right)_{ij} = \beta\left(y_i, x_j\right) \tag{3.48}$$

A continuación se presenta un caso en el que se opera una matriz de dimensiones $m \times n$ con un vector de columna de dimensión n usando la operación σ_β.

La operación $\mathbf{P}_{m\times n}\sigma_\beta\mathbf{x}$ da como resultado un vector columna de dimensión m, cuya $i-$ésima componente tiene la siguiente forma:

$$\left(\mathbf{P}_{m\times n}\sigma_\beta\mathbf{x}\right)_i = \sum_{j=1}^{n} \beta\left(p_{ij}, x_j\right) \tag{3.49}$$

Como ejemplo de los resultados para la operación $\mathbf{P}_{m\times n}\ \sigma_\beta\ \mathbf{x}$, tenemos:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \text{ y } \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \text{ entonces } \mathbf{P}\sigma_\beta\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

3.2.3. Algoritmo de la red neuronal Alfa-Beta sin pesos

Con la finalidad de describir el nuevo modelo denominado con el acrónimo *CAINN* (*Computing Artificial Intelligence Neural Network*), se define el conjunto $A = \{0, 1\}$ de donde toman los valores los patrones tanto de entrada como de salida, y sea su conjunto fundamental como sigue: $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \,|\mathbf{x}^\mu \in A^n$ e $\mathbf{y}^\mu \in A^m$, donde $\mu = 1, 2, \ldots, p\}$.

En las fase de *aprendizaje* como en la de *recuperación*, se requiere un *código de clase* para cada pareja de patrones de entrada y salida, el cual está dado por un patrón $\mathbf{z}^\mu \in A^k$, donde k es la dimensión de un vector *one-hot* que corresponde al valor decimal del patrón binario $\mathbf{x}^\mu$, considerando el bit x_1^μ como el más significativo.

Fase de aprendizaje de CAINN

1. Se crean dos matrices nulas (llenas de valores 0) $\mathbf{P} = [p_{ij}]_{k\times n}$ y $\mathbf{Q} = [q_{ij}]_{m\times k}$.
2. Se denota por $p_{ij}\,(0)$ y $q_{ij}\,(0)$ a los valores 0 iniciales de las componentes $ij-$ésimas de las matrices $\mathbf{P}$ y $\mathbf{Q}$, respectivamente.
3. Para cada $\mu = \{1, 2, \ldots, p\}$ se denotan por $p_{ij}\,(\mu)$ y $q_{ij}\,(\mu)$ los valores que adquieren las $ij-$ésimas componentes de las matrices $\mathbf{P}$ y $\mathbf{Q}$ como consecuencia de las operaciones realizadas en la fase de aprendizaje, de acuerdo con las reglas descritas a continuación:

   a) Se propone el código de clase $\mathbf{z}^\mu \in A^k$

   b) Se actualiza la matriz $\mathbf{P}$ de acuerdo con la siguiente regla de actualización para los componentes p_{ij}

   $$p_{ij}\,(\mu) = \beta\left(\alpha\left(p_{ij}\,(\mu-1), 0\right), \beta\left(z_i^\mu, x_j^\mu\right)\right) \tag{3.50}$$

   c) Se actualiza la matriz $\mathbf{Q}$ de acuerdo con la siguiente regla de actualización para los componentes q_{ij}

   $$q_{ij}\,(\mu) = \beta\left(\alpha\left(q_{ij}\,(\mu-1), 0\right), \beta\left(y_i^\mu, z_j^\mu\right)\right) \tag{3.51}$$

   d) Se genera un vector columna adicional $\mathbf{s} \in A^n$, que contiene en su $i-$ésima componente, la suma de los valores positivos en el $i-$ésimo renglón de la matriz $\mathbf{P}$ [65,66]. Es decir

   $$s_i = \sum_{j=1}^{k} p_{ij}, \qquad \text{tal que } p_{ij} > 0 \tag{3.52}$$

Fase de recuperación de CAINN

Considere un patrón de entrada $\widetilde{\mathbf{x}}^{\omega} \in A^n$, con $\omega \in \{1, 2, \ldots, p\}$ (si sucede que $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^{\omega}$, el patrón es del conjunto fundamental) y se lleva a cabo el siguiente procedimiento:

1. Se realiza la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$; es decir,

$$\left(\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}\right)_i = \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{\mathbf{x}}^{\omega}\right), \ \forall i \in \{1, 2, \ldots, k\} \tag{3.53}$$

2. Se calcula el correspondiente vector de transición $\mathbf{t}^{\omega}$ del código de clase $\mathbf{z}^{\omega}$ como sigue

$$t_i^{\omega} = \alpha_g\left(\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right), \bigvee_{h=1}^{k}\left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^{\omega}\right)\right]\right), \ \forall i \in \{1, 2, \ldots, k\} \tag{3.54}$$

3. Se define el conjunto H como sigue

$$H = \left\{h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{h=1}^{k}\left[\sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega}\right]\right\}, \ \forall i \in \{1, 2, \ldots, k\} \tag{3.55}$$

4. Se calcula el código de clase $\mathbf{z}^{\omega}$ de la siguiente manera

$$z_i^{\omega} = \beta\left(\alpha_g\left(t_i^{\omega}, 1\right), \alpha_g\left(\bigwedge_{h \in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \ \forall i \in \{1, 2, \ldots, k\} \tag{3.56}$$

5. Una vez calculado el código de clase $\mathbf{z}^{\omega}$, se realiza la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$, es decir,

$$\left(\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}\right)_i = \sum_{j=1}^{k} \beta\left(q_{ij}, z_j^{\omega}\right), \ \forall i \in \{1, 2, \ldots, m\} \tag{3.57}$$

6. Finalmente, se calcula el vector $\mathbf{y}^{\omega}$ como sigue

$$y_i^{\omega} = \alpha_g\left(\sum_{j=1}^{k} \beta\left(q_{ij}, z_j^{\omega}\right), \bigvee_{h=1}^{m} \beta\left(q_{hj}, z_j^{\omega}\right)\right), \ \forall i \in \{1, 2, \ldots, m\} \tag{3.58}$$

Se espera que el vector $\mathbf{y}^{\omega}$ sea precisamente el patrón de salida, del conjunto fundamental, correspondiente a $\mathbf{x}^{\omega}$.

3.2.4. El funcionamiento de CAINN

Con la finalidad de ilustrar el funcionamiento del algoritmo CAINN, llevaremos a cabo el siguiente ejemplo, para el cual echaremos mano de lo establecido anteriormente. Para realizar este ejemplo, los patrones de entrada y salida son dados como sigue:

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

Fase de aprendizaje

De acuerdo con los patrones mostrados previamente, podemos obtener los siguientes valores: $n = 4,\ m = 3,\ k = 16,\ p = 3$

1 Creación de las matrices nulas

$$\mathbf{P}_{k\times n} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{Q}_{m\times k} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2 Generación de los códigos de clase

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

3 Se actualizan las matrices de acuerdo con la regla de actualización

$$\mathbf{P}_{k\times n} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{Q}_{m\times k} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4 Se genera el vector columna $\mathbf{s}$

$$\mathbf{s} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0 \\ 0 \\ 3 \\ 0 \end{pmatrix}$$

Fase de recuperación

En esta fase probaremos el algoritmo con los patrones del conjunto fundamental. En este caso, sea $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^1$

1 Se aplica la operación $\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^{\omega}$

$$\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sigma_\beta \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$$

2 Cálculo del vector de transición $\mathbf{t}^{\omega}$

$$\mathbf{t}^{\omega} = \begin{pmatrix} \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(2,2) \\ \alpha_g(2,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(2,2) \\ \alpha_g(0,2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

3 Se define el conjunto H

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij}\widetilde{x}_j^{\omega} \right] \right\}, \text{ de tal forma que}$$

$$H = \{11, 12, 15\}$$

4 Se calcula el código de clase $\mathbf{z}^{\omega}$

$$z_i^{\omega} = \beta\left(\alpha_g\left(t_i^{\omega}, 1\right), \alpha_g\left(\bigwedge_{h\in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \qquad \forall i \in \{1, 2, \ldots, k\}$$

$$\begin{array}{ll} z_1^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_9^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 \\ z_2^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{10}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 \\ z_3^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{11}^{\omega} = \beta\left(\alpha_g(1,1), \alpha_g(2,2)\right) = \beta(1,1) = 1 \\ z_4^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{12}^{\omega} = \beta\left(\alpha_g(1,1), \alpha_g(2,3)\right) = \beta(1,0) = 0 \\ z_5^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{13}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 \\ z_6^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{14}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 \\ z_7^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{15}^{\omega} = \beta\left(\alpha_g(1,1), \alpha_g(2,3)\right) = \beta(1,0) = 0 \\ z_8^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 & z_{16}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(2,0)\right) = \beta(0,2) = 0 \end{array}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

5 Se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

6 Por último, se calcula el vector de salida $\mathbf{y}^{\omega}$

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta \left(q_{ij}, z_j^{\omega} \right), \bigvee_{h=1}^{m} \beta \left(q_{hj}, z_j^{\omega} \right) \right), \qquad \forall i \in \{1, 2, \ldots, m\}$$

$$\begin{aligned} y_1^{\omega} &= \alpha_g \left(1, \bigvee (1,0,0)\right) = \alpha_g (1,1) = 1 \\ y_2^{\omega} &= \alpha_g \left(0, \bigvee (1,0,0)\right) = \alpha_g (0,1) = 0 \\ y_3^{\omega} &= \alpha_g \left(0, \bigvee (1,0,0)\right) = \alpha_g (0,1) = 0 \end{aligned}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \mathbf{y}^1$ $\quad \therefore$ la recuperación es correcta.

Tomemos ahora el caso para el cual el vector $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^2$

1 Se aplica la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$

$$\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 3 \\ 0 \end{pmatrix}$$

2 Cálculo del vector de transición $\mathbf{t}^{\omega}$

$$\mathbf{t}^{\omega} = \begin{pmatrix} \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(2,3) \\ \alpha_g(2,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(3,3) \\ \alpha_g(0,3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

3 Se define el conjunto H

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij}\widetilde{x}_j^{\omega} \right] \right\}, \text{ de tal forma que}$$

$$H = \{15\}$$

4 Se calcula el código de clase $\mathbf{z}^{\omega}$

$$z_i^{\omega} = \beta\left(\alpha_g\left(t_i^{\omega}, 1\right), \alpha_g\left(\bigwedge_{h\in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \qquad \forall i \in \{1, 2, \ldots, k\}$$

$$\begin{array}{ll} z_1^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_9^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_2^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{10}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_3^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{11}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,2)\right) = \beta(0,2) = 0 \\ z_4^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{12}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,3)\right) = \beta(0,1) = 0 \\ z_5^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{13}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_6^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{14}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_7^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{15}^{\omega} = \beta\left(\alpha_g(1,1), \alpha_g(3,3)\right) = \beta(1,1) = 1 \\ z_8^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{16}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \end{array}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

5 Se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

6 Por último, se calcula el vector de salida $\mathbf{y}^{\omega}$

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta \left(q_{ij}, z_j^{\omega} \right), \bigvee_{h=1}^{m} \beta \left(q_{hj}, z_j^{\omega} \right) \right), \qquad \forall i \in \{1, 2, \ldots, m\}$$

$$\begin{aligned} y_1^{\omega} &= \alpha_g \left(0, \bigvee (0,1,0)\right) = \alpha_g (0,1) = 0 \\ y_2^{\omega} &= \alpha_g \left(1, \bigvee (0,1,0)\right) = \alpha_g (1,1) = 1 \\ y_3^{\omega} &= \alpha_g \left(0, \bigvee (0,1,0)\right) = \alpha_g (0,1) = 0 \end{aligned}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \mathbf{y}^2$ $\therefore$ la recuperación es correcta.

Ahora sólo resta realizar la fase de recuperación para el caso en que el vector $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^3$

1 Se aplica la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$

$$\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$$

2 Cálculo del vector de transición $\mathbf{t}^{\omega}$

$$\mathbf{t}^{\omega} = \begin{pmatrix} \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(2,3) \\ \alpha_g(3,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(2,3) \\ \alpha_g(0,3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3 Se define el conjunto H

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij}\widetilde{x}_j^{\omega} \right] \right\}, \text{ de tal forma que}$$

$$H = \{12\}$$

4 Se calcula el código de clase $\mathbf{z}^{\omega}$

$$z_i^{\omega} = \beta\left(\alpha_g\left(t_i^{\omega}, 1\right), \alpha_g\left(\bigwedge_{h\in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \qquad \forall i \in \{1, 2, \ldots, k\}$$

$$\begin{array}{ll} z_1^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_9^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_2^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{10}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_3^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{11}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,2)\right) = \beta(0,2) = 0 \\ z_4^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{12}^{\omega} = \beta\left(\alpha_g(1,1), \alpha_g(3,3)\right) = \beta(1,1) = 1 \\ z_5^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{13}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_6^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{14}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \\ z_7^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{15}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,3)\right) = \beta(0,1) = 0 \\ z_8^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 & z_{16}^{\omega} = \beta\left(\alpha_g(0,1), \alpha_g(3,0)\right) = \beta(0,2) = 0 \end{array}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

5 Se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

6 Por último, se calcula el vector de salida $\mathbf{y}^{\omega}$

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta \left(q_{ij}, z_j^{\omega} \right), \bigvee_{h=1}^{m} \beta \left(q_{hj}, z_j^{\omega} \right) \right), \qquad \forall i \in \{1, 2, \ldots, m\}$$

$$\begin{aligned} y_1^{\omega} &= \alpha_g \left(0, \bigvee (0,0,1)\right) = \alpha_g (0,1) = 0 \\ y_2^{\omega} &= \alpha_g \left(0, \bigvee (0,0,1)\right) = \alpha_g (0,1) = 0 \\ y_3^{\omega} &= \alpha_g \left(1, \bigvee (0,0,1)\right) = \alpha_g (1,1) = 1 \end{aligned}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{y}^3$ $\quad \therefore$ la recuperación es correcta.

Capítulo 4

Modelo propuesto

En este capítulo daremos a conocer las aportaciones originales de la tesis. Tras haber definido ciertos conjuntos y operadores, comenzaremos la sección demostrando una serie de lemas y teoremas que servirán como fundamentación para que posteriormente se proponga un algoritmo alternativo que optimiza el tiempo de procesamiento de CAINN. Finalmente, se evidenciarán las condiciones suficientes para la recuperación correcta de patrones del conjunto fundamental y para patrones desconocidos.

4.1. Definiciones, Lemas y Teoremas

Durante todas las definiciones y demostraciones echaremos mano de la siguiente notación:

- $A = \{0, 1\}$
- $B = \{0, 1, 2\}$
- $n \in \mathbb{Z}^+$
- $m \in \mathbb{Z}^+$
- $k \in \mathbb{Z}^+$
- $k = 2^n$
- $p \in \mathbb{Z}^+$
- $\mu = 1, 2, \ldots, p$
- $\omega \in \{1, 2, \ldots, p\}$

Notación 4.1 *Las matrices que utilizaremos se representan con esta notación: si m, n y k son números enteros positivos, $\mathbf{P} = [p_{ij}]_{k\times n}$ representa una matriz de dimensiones $k \times n$, cuya $ij-$ésima entrada es p_{ij} y $\mathbf{Q} = [q_{ij}]_{m\times k}$ representa una matriz de dimensiones $m \times k$, cuya $ij-$ésima entrada es q_{ij}.*

Nota 4.2 *De acuerdo con las operaciones que el modelo CAINN realiza, los valores de las componentes p_{ij} y q_{ij} pertenecen al conjunto A.*

Definición 4.3 [66] *Sea $x^\omega \in A^n$, un patrón de entrada. La suma de los valores positivos de los componentes de dicho patrón es:*

$$U_{x^\omega} = \sum_{i=1}^{n} x_i^\omega, \qquad \text{tal que } x_i^\omega > 0$$

Ejemplo 4.4 *Sea* $x^1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$; *entonces* $U_{x^1} = 3$

Definición 4.5 *Sea $y^\omega \in A^m$, un patrón de salida. La suma de los valores positivos de los componentes de dicho patrón es:*

$$U_{y^\omega} = \sum_{i=1}^{m} y_i^\omega, \qquad \text{tal que } y_i^\omega > 0$$

Ejemplo 4.6 *Sea* $y^9 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$; *entonces* $U_{y^9} = 2$

Definición 4.7 *Sean $x^a, x^b \in A^n$ dos vectores; entonces:*

$$\mathbf{x}^a < \mathbf{x}^b \longleftrightarrow \forall i,\ x_i^a \leq x_i^b \text{ y } \exists j | x_j^a < x_j^b$$

Ejemplo 4.8 $x^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} < x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

Definición 4.9 *Sean* $x^a, x^b \in A^n$ *dos vectores; entonces:*

$$\mathbf{x}^a \leq \mathbf{x}^b \longleftrightarrow x_i^a \leq x_i^b, i = \{1, 2, \ldots, n\}$$

Ejemplo 4.10 $x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \leq x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

Ejemplo 4.11 $x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \leq x^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$

Definición 4.12 *Sea* $x^\omega \in A^n$*, un patrón fundamental de entrada. Si tomamos como dígito menos significativo a* x_n^ω *y como el más significativo a* x_1^ω*, denotaremos con el símbolo* d^ω *el correspondiente valor decimal del vector* $x^\omega + 1$*.*

Ejemplo 4.13 *Sea* $x^1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$; *entonces* $d^1 = 14$

Ejemplo 4.14 *Sea* $x^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$; *entonces* $d^2 = 10$

Ejemplo 4.15 *Sea* $x^3 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$; *entonces* $d^3 = 16$

Definición 4.16 *El conjunto de todos los valores de* d^μ *será denotado por* D*, es decir*

$$D = \{d^\mu | \mu = 1, 2, \ldots, p\}$$

Ejemplo 4.17 *Tomando en cuenta el* Ejemplo 4.13*, el* Ejemplo4.14 *y el* Ejemplo4.15 *como patrones fundamentales, entonces:* $D = \{14, 10, 16\}$

Lema 4.18 *Sea* $\mathbf{x}^\omega \in A^n$ *un patrón fundamental de entrada tomado arbitrariamente. Durante la fase de aprendizaje,* $\mathbf{x}^\omega$ *sólo contribuye en la* $i-$*ésima fila de la matriz* $\mathbf{P}$ *con* $i = d^\omega$*, y contribuye con exactamente* U_{x^ω} $1's$ *y* $n - U_{x^\omega}$ $0's$*. Es decir:*

$$x_j^\omega = 1 \longrightarrow p_{ij} = 1,\ i = d^\omega$$

Demostración. De acuerdo con la fase de aprendizaje, la matriz $\mathbf{P}$ se actualiza con la sig regla:

$$p_{ij}(\mu) = \beta\left(\alpha\left(p_{ij}(\mu - 1), 0\right), \beta\left(z_i^\mu, x_j^\mu\right)\right)$$

Si el valor de $p_{ij}(\mu - 1) = 1$, automáticamente $p_{ij}(\mu) = 1$ dadas las definiciones de α y β.

Por otro lado, si $p_{ij}(\mu - 1) = 0$, $p_{ij}(\mu) = 1$ sólo podría obtenerse en las posiciones de i, j en las cuales $x_j^\omega = 1$ e $i = d^\omega$. Así pues, la matriz $\mathbf{P}$ sólo es afectada por el vector de entrada $\mathbf{x}^\omega$ en su $i-$ésima fila, cuando $i = d^\omega$.

Además, puesto que la matriz $\mathbf{P}$ sólo es modificada cuando $i \in D$ y de acuerdo con la *Definición 4.12*, el conjunto D posee un único valor d^ω para cada patrón fundamental de entrada, entonces $\mathbf{x}^\omega$ contribuirá en una y sólo una fila de $\mathbf{P}$ con tantos $1's$ como éste tenga y en las mismas posiciones, *i.e.*, con U_{x^ω} $1's$ y con $n - U_{x^\omega}$ $0's$. ■

Lema 4.19 *Sea* $\mathbf{s}$ *el vector construido en la fase de aprendizaje. Para cada una de las componentes de* $\mathbf{s}$*:*

$$s_{d^\omega} = U_{x^\omega}$$

Demostración. De acuerdo con la fase de aprendizaje, el vector $\mathbf{s}$ se calcula como:

$$s_i = \sum_{j=1}^{n} p_{ij} \tag{4.1}$$

Sabemos por el *Lema 4.18* que $\mathbf{x}^\omega$ sólo contribuye en la $i-$ésima fila de $\mathbf{P}$ cuando $i = d^\omega$, por lo que podemos decir que

$$\forall j, x_j^\omega = 1 \longrightarrow p_{d^\omega j} = 1 \tag{4.2}$$

Tomando en cuenta la expresión 4.2 podemos reescribir la expresión 4.1 como

$$s_{d^\omega} = \sum_{j=1}^{n} p_{d^\omega j}$$

pero además también sabemos que la $i-$ésima fila tendrá tantos $1's$ como tenga $\mathbf{x}^\omega$ cuando $i = d^\omega$, es decir, tendrá U_{x^ω} $1's$, así pues:

$$s_{d^\omega} = U_{x^\omega}$$

■

Lema 4.20 *Sea* $\mathbf{z}^\omega$ *un código de clase y* $\mathbf{t}^\omega$ *su respectivo vector de transición, obtenidos en la fase de recuperación de CAINN . Durante esta fase, el algoritmo pondrá en* $0's$ *todas las componentes de* $\mathbf{z}^\omega$ *excepto aquella en donde la* $i-$*ésima componente del vector* $\mathbf{s}$ *obtenido en la fase de aprendizaje, es mínima para las filas en las que el vector* $\mathbf{t}^\omega$ vale 1*, esto es:*

$$z_i^\omega = \begin{cases} 1 & \text{si } \ s_i = \bigwedge_{h\in H}^{k} s_h \\ 0 & \text{otro caso} \end{cases}$$

Demostración. De acuerdo con la fase de recuperación de CAINN, $\mathbf{z}^\omega$ se calcula de la siguiente manera:

$$z_i^\omega = \beta\left(\alpha_g\left(t_i^\omega, 1\right), \alpha_g\left(\bigwedge_{h\in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \quad i \in \{1, 2, \ldots, k\} \tag{4.3}$$

Si $t_i^\omega = 0$, el resultado para z_i^ω sería 0 dadas las definiciones de α_g y β. Por otro lado si $t_i^\omega = 1$,

$$\alpha_g\left(t_i^\omega, 1\right) = 1 \tag{4.4}$$

sustituyendo la expresión 4.4 en 4.3, tenemos que

$$z_i^\omega = \beta\left(1, \alpha_g\left(\bigwedge_{h\in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \quad i \in \{1, 2, \ldots, k\} \tag{4.5}$$

dada la expresión 4.5 y de acuerdo con la definición de β y α_g, $z_i^\omega = 1$ sólo podriá ser obtenido en las filas para las cuales

$$\bigwedge_{h\in H}\left[\sum_{j=1}^{n} p_{hj}\right] = \sum_{j=1}^{n} p_{ij}$$

En la fase de aprendizaje de CAINN se construye el vector **s** de forma tal que

$$s_i = \sum_{j=1}^{n} p_{ij} \tag{4.6}$$

sustituyendo la expresión 4.6 en 4.5 tendremos que

$$z_i^{\omega} = \beta\left(1, \alpha_g\left(\bigwedge_{h\in H}\left[s_h\right], s_i\right)\right), \quad i \in \{1, 2, \ldots, k\} \tag{4.7}$$

Por lo tanto, para que $z_i^{\omega} = 1$, s_i debe ser mínimo cuando $t_i^{\omega} = 1$. Dado que el conjunto H contiene los valores para los cuales $t_i^{\omega} = 1$ La expresión 4.7 ahora queda como

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge\limits_{h\in H}^{k} s_h \\ 0 & \text{otro caso} \end{cases}$$

■

Teorema 4.21 *Sea $\mathbf{x}^{\omega} \in A^n$ un patrón que se presenta como entrada a $\mathbf{P}$. Durante la fase de recuperación se realiza la operación $\mathbf{P}\ \sigma_\beta\ \mathbf{x}^{\omega}$*, cuyo *resultado sólo podrá tener valores iguales a cero en los índices i que correspondan a la $i-$ésima fila de $\mathbf{P}$, cuando ésta no fue modificada por algún patrón $\mathbf{x}^{\omega}$. Puesto de otra forma:*

$$\left(\mathbf{P}\ \sigma_\beta\ \mathbf{x}^{\omega}\right)_i = 0 \longleftrightarrow i \notin D$$

Demostración. Sabemos por el *Lema 4.18* que $\mathbf{x}^{\omega}$ modifica la $i-$ésima fila de $\mathbf{P}$ cuando $i = d^{\omega}$, y que, dado esto, la matriz $\mathbf{P}$ sólo es modificada cuando $i \in D$. Es decir que si $i \notin D$, la matriz $\mathbf{P}$ sólo contendrá $0s$ en su $i-$ésima fila, de forma tal que al realizar la operación σ_β entre $\mathbf{P}$ y $\mathbf{x}^{\omega}$, el resultado siempre será 0 dadas las definiciones de β y de σ_β. ■

Teorema 4.22 *Sea $\mathbf{x}^{\omega} \in A^n$ un patrón fundamental de entrada y sea la matriz $\mathbf{P}_{k\times n}$ construida durante la fase de aprendizaje de CAINN. Dada la forma en que esta matriz se utiliza a lo largo del algoritmo, el tamaño de la misma puede reducirse hasta ser una matriz con dimensiones $p\times n$, donde p es el número de asociaciones entre los patrones fundamentales.*

Demostración. Dado el *Teorema 4.21*, hemos observado que:

$$\left(\mathbf{P}\ \sigma_\beta\ \mathbf{x}^{\omega}\right)_i = 0 \longleftrightarrow i \notin D$$

es decir que la matriz $\mathbf{P}$ tendrá filas llenas de ceros en aquellos índices de filas que no fueron modificadas por un patrón fundamental de entrada. Por otro lado, por el *Lema 4.18* sabemos que cada uno de los patrones fundamentales de entrada contribuye únicamente en una y solo una fila de $\mathbf{P}$ de tal forma:

$$x_j^{\omega} = 1 \longrightarrow p_{ij} = 1,\ i = d^{\omega}$$

Por lo tanto la matriz $\mathbf{P}$ tiene tantas filas útiles como patrones fundamentales existan, luego entonces la matriz $\mathbf{P}$ puede ser reducida de un número de k filas, a sólo p filas, donde p es el número de asociaciones del conjunto fundamental. ■

Lema 4.23 *Sea $\mathbf{y}^{\omega} \in A^m$ un patrón fundamental de salida tomado arbitrariamente. Durante la fase de aprendizaje, $\mathbf{y}^{\omega}$ sólo contribuye en la $j-$ésima columna de la matriz $\mathbf{Q}$ con $j = d^{\omega}$, y contribuye con exactamente $U_{y^{\omega}}$ $1's$ y $m - U_{y^{\omega}}$ $0's$. Es decir:*

$$y_i^{\omega} = 1 \longrightarrow q_{ij} = 1,\ j = d^{\omega}$$

Demostración. De acuerdo con la fase de aprendizaje, la matriz $\mathbf{Q}$ se actualiza con la sig regla:

$$q_{ij}(\mu) = \beta\left(\alpha\left(q_{ij}(\mu-1), 0\right), \beta\left(y_i^{\mu}, z_j^{\mu}\right)\right)$$

Si el valor de $q_{ij}(\mu - 1) = 1$, automáticamente $q_{ij}(\mu) = 1$ dadas las definiciones de α y β.

Por otro lado, si $q_{ij}(\mu - 1) = 0$, $q_{ij}(\mu) = 1$ sólo podría obtenerse en las posiciones de i, j en las cuales $y_i^{\omega} = 1$ y $z_j^{\omega} = 1$, es decir que $j = d^{\omega}$. Así pues, la matriz $\mathbf{Q}$ sólo es afectada por el vector de entrada $\mathbf{y}^{\omega}$ en su $j-$ésima columna, cuando $j = d^{\omega}$.

Además, puesto que la matriz $\mathbf{Q}$ sólo es modificada cuando $j \in D$ y de acuerdo con la *Definición 4.12*, el conjunto D posee un único valor d^{ω} para cada patrón fundamental, entonces $\mathbf{y}^{\omega}$ contribuirá en una y sólo una fila de $\mathbf{Q}$ con tantos $1's$ como éste tenga y en las mismas posiciones, *i.e.*, con $U_{y^{\omega}}$ $1's$ y con $m - U_{y^{\omega}}$ $0's$. ■

Teorema 4.24 *Sea $\mathbf{y}^{\omega} \in A^m$ un patrón que se utiliza para calcular la matriz $\mathbf{Q}$. Durante la fase de aprendizaje se realiza la actualización de $\mathbf{Q}$; durante esta actualización su $j-$ésima columna estará llena de valores iguales a cero cuando j no pertenezca al conjunto D. Puesto de otra forma:*

$$q_{ij} = 0 \longleftrightarrow j \notin D$$

Demostración. Sabemos por el *Lema 4.23* que $\mathbf{y}^{\omega}$ modifica la $j-$ésima columna de $\mathbf{Q}$ cuando $j = d^{\omega}$, y que, dado esto, la matriz $\mathbf{Q}$ sólo es modificada cuando $j \in D$. Es decir que si $j \notin D$, la matriz $\mathbf{Q}$ seguirá estando llena de ceros en su $j-$ésima columna dadas las definiciones de β y de σ_{β}. ■

Teorema 4.25 *Sea $\mathbf{y}^{\omega} \in A^m$ un patrón fundamental de salida y sea la matriz $\mathbf{Q}_{m \times k}$ construida durante la fase de aprendizaje de CAINN. Dada la forma en que esta matriz se utiliza a lo largo del algoritmo, el tamaño de la misma puede reducirse hasta ser una matriz con dimensiones $m \times p$, donde p es el número de asociaciones entre los patrones fundamentales.*

Demostración. Dado el *Teorema 4.24*, hemos observado que

$$q_{ij} = 0 \longleftrightarrow j \notin D$$

es decir que la matriz $\mathbf{Q}$ contendrá sólo ceros en aquellas columnas en donde su índice no pertenezca al conjunto D dado en la *Definición 4.16*. Por otro lado, el *Lema 4.23* nos dice que cada uno de los patrones fundamentales de salida contribuye únicamente en una y solo una columna de $\mathbf{Q}$, de tal forma:

$$y_i^{\omega} = 1 \longrightarrow q_{ij} = 1,\ j = d^{\omega}$$

Por lo tanto la matriz $\mathbf{Q}$ tiene tantas columna útiles como patrones fundamentales existan, luego entonces la matriz $\mathbf{Q}$ puede ser reducida de un número de k columnas, a sólo p columnas, siendo p el número de asociaciones del conjunto fundamental. ■

Nota 4.26 *Dado que los vectores s, t y z son construidos a partir de las matrices $\mathbf{P}$ y $\mathbf{Q}$, dichos vectores reducen también su dimensión de k a p.*

Notación 4.27 *De acuerdo con el* Teorema 4.22 *y con el* Teorema 4.25*, de ahora en adelante diremos que: si m,n y p son números enteros positivos, $\mathbf{P} = [p_{ij}]_{p\times n}$ representa una matriz de dimensiones $p\times n$, cuya $ij-$ésima entrada es p_{ij} y $\mathbf{Q} = [q_{ij}]_{m\times p}$ representa una matriz de dimensiones $m \times p$, cuya $ij-$ésima entrada es q_{ij}.*

Definición 4.28 *De acuerdo con la* Notación 4.27*, redefiniremos la manera en que se proponen los códigos de clase $\mathbf{z}^{\mu}$, de tal forma que se calculen como*

$$z_i^{\mu} = \begin{cases} 1 & \text{si} \quad i = \mu \\ 0 & \text{otro caso} \end{cases}$$

Teorema 4.29 *Sea $\mathbf{x}^{\omega} \in A^n$ un patrón fundamental de entrada tomado arbitrariamente; durante la fase de aprendizaje, $\mathbf{x}^{\omega}$ sólo contribuye en la $\omega-$ésima fila de la matriz $\mathbf{P}$, y contribuye con exactamente $U_{x^{\omega}}$ $1's$ y $n - U_{x^{\omega}}$ $0's$. Es decir:*

$$x_j^{\omega} = 1 \longrightarrow p_{\omega j} = 1$$

Demostración. Tomando en cuenta el *Teorema 4.22* y la *Notación 4.27*, y de acuerdo con la fase de aprendizaje, la matriz $\mathbf{P}$ se actualiza con la sig regla:

$$p_{ij}(\mu) = \beta\left(\alpha\left(p_{ij}(\mu - 1), 0\right), \beta\left(z_i^{\mu}, x_j^{\mu}\right)\right)$$

Si el valor de $p_{ij}(\mu - 1) = 1$, automáticamente $p_{ij}(\mu) = 1$ dadas las definiciones de α y β. Por otro lado, si $p_{ij}(\mu - 1) = 0$, $p_{ij}(\mu) = 1$ sólo podría obtenerse en las

posiciones de i, j para las cuales $x_j^\omega = 1 = z_i^\omega$. Dado que $z_i^\omega = 1$ siempre y cuando $i = \omega$, podemos decir que la matriz $\mathbf{P}$ sólo es modificada por el vector de entrada $\mathbf{x}^\omega$ en su $\omega-$ésima fila y que, además, el vector $\mathbf{x}^\omega$ sólo contribuirá en una y sólo una fila de $\mathbf{P}$ con tantos $1's$ como éste tenga y en las mismas posiciones, *i.e.*, con U_{x^ω} $1's$ y con $n - U_{x^\omega}$ $0's$. ■

Teorema 4.30 *Sea* $\mathbf{y}^\omega \in A^m$ *un patrón fundamental de salida tomado arbitrariamente; durante la fase de aprendizaje,* $\mathbf{y}^\omega$ *sólo contribuye en la* $\omega-$*ésima columna de la matriz* $\mathbf{Q}$*, y contribuye con exactamente* U_{y^ω} $1's$ *y* $m - U_{y^\omega}$ $0's$*. Es decir:*

$$y_i^\omega = 1 \longrightarrow q_{i\omega} = 1$$

Demostración. Tomando en cuenta el *Teorema 4.25* y la *Notación 4.27*, y de acuerdo con la fase de aprendizaje, la matriz $\mathbf{Q}$ se actualiza con la sig regla:

$$q_{ij}(\mu) = \beta\left(\alpha\left(q_{ij}(\mu - 1), 0\right), \beta\left(y_i^\mu, z_j^\mu\right)\right)$$

Si el valor de $q_{ij}(\mu - 1) = 1$, automáticamente $q_{ij}(\mu) = 1$ dadas las definiciones de α y β. Por otro lado, si $q_{ij}(\mu - 1) = 0$, $q_{ij}(\mu) = 1$ sólo podría obtenerse en las posiciones de i, j para las cuales $y_i^\omega = 1 = z_j^\omega$. Dado que $z_j^\omega = 1$ siempre y cuando $j = \omega$, podemos decir que la matriz $\mathbf{Q}$ sólo es modificada por el vector de salida $\mathbf{y}^\omega$ en su $\omega-$ésima columna y que, además, el vector $\mathbf{y}^\omega$ sólo contribuirá en una y sólo una columna de $\mathbf{Q}$ con tantos $1's$ como éste tenga y en las mismas posiciones, *i.e.*, con U_{y^ω} $1's$ y con $m - U_{y^\omega}$ $0's$. ■

Teorema 4.31 *Sea* $\mathbf{s}$ *el vector construido en la fase de aprendizaje. Para cada una de las componentes de* $\mathbf{s}$*:*

$$s_\omega = U_{x^\omega}$$

Demostración. De acuerdo con la fase de aprendizaje, el vector $\mathbf{s}$ se calcula como:

$$s_i = \sum_{j=1}^{n} p_{ij} \tag{4.8}$$

Sabemos por el *Teorema 4.29* que $\mathbf{x}^\omega$ sólo contribuye en la $\omega-$ésima fila de $\mathbf{P}$, de tal forma que

$$\forall j, x_j^\omega = 1 \longrightarrow p_{\omega j} = 1 \tag{4.9}$$

Tomando en cuenta la expresión 4.9 y la *Notación 4.27*, podemos reescribir la expresión 4.8 como

$$s_\omega = \sum_{j=1}^{n} p_{\omega j}$$

pero además también sabemos que la $\omega-$ésima fila tendrá tantos $1's$ como tenga $\mathbf{x}^\omega$, es decir, tendrá U_{x^ω} $1's$, así pues:

$$s_\omega = U_{x^\omega}$$

■

Tomando en cuenta la *Notación 4.27* modificaremos la expresión del primer paso de la fase de recuperación, de tal forma que quede adaptada a la nueva notación propuesta. Así pues tenemos la siguiente definición:

Definición 4.32 *Sea* $\mathbf{x}^\omega \in A^n$ *un patrón que se presenta como entrada a* $\mathbf{P}$*. En el primer paso para llevar a cabo la fase de recuperación se realiza la operación* $\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^\omega$ *cuya* $i-$*ésima componente queda expresada como:*

$$(\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^\omega)_i = \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{\mathbf{x}}_j^\omega\right),\ \forall i \in \{1,2,\ldots,p\}$$

Lema 4.33 *Sea* $\mathbf{t}^\omega$ *el vector de transición obtenido en la fase de recuperación. El vector* $\mathbf{t}^\omega$ *tendrá* $1s$ *en su* $i-$*ésima componente, siempre y cuando* $(\mathbf{P}\ \sigma_\beta\ \widetilde{\mathbf{x}}^\omega)_i$ *sea máximo, es decir:*

$$t_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^\omega\right) = \bigvee_{i=1}^{p}\left[\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right)\right],\ h \in \{1,2,\ldots p\} \\ 0 & \text{otro caso} \end{cases}$$

Demostración. Dada la fase de recuperación de CAINN y tomando en cuenta la *Notación 4.27*, $\mathbf{t}^\omega$ se calcula de la siguiente manera:

$$t_i^\omega = \alpha_g\left(\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^\omega\right), \bigvee_{i=1}^{p}\left[\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right)\right]\right),\ h \in \{1,2,\ldots,p\}$$

De acuerdo con la definición de α_g, $t_i^\omega = 1$ sólo puede ser obtenido cuando

$$\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right) = \bigvee_{h=1}^{p}\left[\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right)\right]$$

Es decir que el resultado de la operación $\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right)$ debe ser máximo. En el caso de que $\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right)$ no sea máximo (es decir, menor que el máximo), el resultado sería 0 dada la definición de α_g. Puesto de otra forma:

$$t_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^\omega\right) = \bigvee_{i=1}^{p}\left[\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right)\right],\ h \in \{1,2,\ldots p\} \\ 0 & \text{otro caso} \end{cases}$$

■

Lema 4.34 *Sea $\widetilde{\mathbf{x}}^{\omega} \in A^n$ un vector que se presenta como entrada a $\mathbf{P}$. Para el caso de la fase de recuperación en CAINN, la operación $\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right)$ es equivalente a la operación $\sum_{j=1}^{n} p_{ij} \cdot \widetilde{x}_j^{\omega}$.*

Demostración. De acuerdo con el *Teorema 4.29*, el vector $\widetilde{\mathbf{x}}^{\omega}$ sólo contribuye en la $\omega-$ésima fila de $\mathbf{P}$ con tantos $1's$ como este tenga; como $\widetilde{\mathbf{x}}^{\omega} \in A^n$, entonces las componentes de $\mathbf{P}$ sólo podrán ser valores del conjunto A, es decir, 0 o 1. Si multiplicamos la matriz $\mathbf{P}$ por el vector $\widetilde{\mathbf{x}}^{\omega}$:

$$p_{ij} \cdot \widetilde{x}_j^{\omega} \tag{4.10}$$

obtendremos como resultado un 1 siempre y cuando

$$p_{ij} = 1 = \widetilde{x}_j^{\omega} \tag{4.11}$$

de otra forma tendremos como resultado un 0. De la misma manera, si realizamos la operación

$$\beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right) \tag{4.12}$$

dada la definición de β y al igual que para la expresión 4.10, obtendremos como resultado un 1 siempre y cuando

$$p_{ij} = 1 = \widetilde{x}_j^{\omega} \tag{4.13}$$

o un 0 en cualquier otro caso dado que sólo podemos tomar valores dentro del conjunto A, es decir: $0's$ y $1's$. Dicho esto, y de acuerdo con las expresiones 4.11 y 4.13, podemos relacionar las expresiones 4.10 y 4.12, de tal forma que:

$$p_{ij} \cdot \widetilde{x}_j^{\omega} = \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right)$$

■

Teorema 4.35 *Sean $\mathbf{t}^{\omega}$ y H un vector de transición y un conjunto, respectivamente, obtenidos en la fase de recuperación de CAINN. El conjunto H contendrá los índices para los cuales $t_i^{\omega} = 1$. Es decir:*

$$H = i| \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right) = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^{\omega}\right)\right]$$

Demostración. De acuerdo con la *Notación 4.27* y la forma en cómo se obtiene H, tenemos que:

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj} \cdot \widetilde{x}_j^{\omega} = \bigvee_{i=1}^{p} \left[\sum_{j=1}^{n} p_{ij} \cdot \widetilde{x}_j^{\omega}\right] \right\}, \qquad i \in \{1, 2, \ldots, p\}$$

es decir, H tomará los índices para los cuales haya un **máximo** en la operación

$$\sum_{j=1}^{n} p_{ij} \cdot \widetilde{x}_j^{\omega}$$

Por otra parte, el *Lema 4.33* nos dice que $t_i^{\omega} = 1$ siempre y cuando encontremos un **máximo** al realizar la operación

$$\sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right)$$

Pero además, sabemos por el *Lema 4.34* que $p_{ij} \cdot \widetilde{x}_j^{\omega} = \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right)$ por lo tanto, H tomará todos aquellos valores en los que $t_i^{\omega} = 1$. Puesto de otra forma,

$$H = i | \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right) = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^{\omega}\right)\right]$$

■

Teorema 4.36 *Sea* $\mathbf{z}^{\omega}$ *un código de clase y* $\mathbf{t}^{\omega}$ *su respectivo vector de transición, obtenidos en la fase de recuperación de CAINN . Durante esta fase, el algoritmo pondrá en* $0's$ *todas las componentes de* $\mathbf{z}^{\omega}$ *excepto aquella en donde la* $i-$*ésima componente del vector* $\mathbf{s}$ *obtenido en la fase de aprendizaje, es mínima para las filas en las que el vector* $\mathbf{t}^{\omega}$ vale 1 (es decir, las filas cuyo índice esté contenido en H)*, esto es:*

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \ s_i = \bigwedge_{h \in H}^{p} s_h \\ 0 & \text{otro caso} \end{cases}$$

Demostración. De acuerdo con la fase de recuperación de CAINN y tomando en cuenta la *Notación 4.27*, $\mathbf{z}^{\omega}$ se calcula de la siguiente manera:

$$z_i^{\omega} = \beta\left(\alpha_g\left(t_i^{\omega}, 1\right), \alpha_g\left(\bigwedge_{h \in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \quad i \in \{1, 2, \ldots, p\} \tag{4.14}$$

Si $t_i^{\omega} = 0$, el resultado para z_i^{ω} sería 0 dadas las definiciones de α_g y β. Por otro lado si $t_i^{\omega} = 1$,

$$\alpha_g\left(t_i^{\omega}, 1\right) = 1 \tag{4.15}$$

sustituyendo la expresión 4.15 en 4.14, tenemos que

$$z_i^{\omega} = \beta\left(1, \alpha_g\left(\bigwedge_{h \in H}\left[\sum_{j=1}^{n} p_{hj}\right], \sum_{j=1}^{n} p_{ij}\right)\right), \quad i \in \{1, 2, \ldots, p\} \tag{4.16}$$

dada la expresión 4.16 y de acuerdo con la definición de β y α_g, $z_i^{\omega} = 1$ sólo podriá ser obtenido en las filas para las cuales

$$\bigwedge_{h \in H}\left[\sum_{j=1}^{n} p_{hj}\right] = \sum_{j=1}^{n} p_{ij}$$

En la fase de aprendizaje de CAINN se construye el vector **s** de forma tal que

$$s_i = \sum_{j=1}^{n} p_{ij} \tag{4.17}$$

sustituyendo la expresión 4.17 en 4.16 tendremos que

$$z_i^{\omega} = \beta\left(1, \alpha_g\left(\bigwedge_{h\in H}\left[s_h\right], s_i\right)\right), \quad i \in \{1, 2, \ldots, p\} \tag{4.18}$$

Por lo tanto, para que $z_i^{\omega} = 1$, debemos tener $\bigwedge\limits_{h\in H}\left[s_h\right] = s_i$, es decir que s_i debe ser mínimo cuando $h \in H$. Dado esto, la expresión 4.18 ahora queda como

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge\limits_{h\in H}^{p} s_h \\ 0 & \text{otro caso} \end{cases}$$

■

Tomando en cuenta la *Notación 4.27* modificaremos la expresión del último paso de la fase de recuperación, de tal forma que quede adaptada a la nueva notación propuesta. Así pues tenemos la siguiente definición:

Definición 4.37 *Sea $\mathbf{z}^{\omega} \in A^p$ un código de clase que se opera con la matriz $\mathbf{Q}$ para realizar el cálculo del vector de salida $\mathbf{y}^{\omega} \in A^m$. En el último paso para llevar a cabo la fase de recuperación se realiza la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$ cuya $\omega-$ésima componente queda expresada como:*

$$y_i^{\omega} = \sum_{j=1}^{p} \beta\left(q_{ij}, z_j^{\omega}\right), \; \forall i \in \{1, 2, \ldots, m\}$$

4.2. Optimización del algoritmo

Fase de aprendizaje de CAINN

1. Se crean dos matrices nulas (llenas de 0s) $\mathbf{P} = [p_{ij}]_{p\times n} = 0$ y $\mathbf{Q} = [q_{ij}]_{m\times p} = 0$. Se denota por $p_{ij}\,(0)$ y $q_{ij}\,(0)$ a los valores 0 iniciales de las componentes $ij-$ésimas de las matrices **P** y **Q**, respectivamente.

2. Se proponen los códigos de clase $\mathbf{z}^{\mu} \in A^p$ de acuerdo con la *Definición 4.28* como sigue:
$$z_i^{\mu} = \begin{cases} 1 & \text{si} \quad i = \mu \\ 0 & \text{otro caso} \end{cases} \tag{4.19}$$

3. De acuerdo con el *Teorema 4.29* ,se actualiza la matriz **P** de tal forma que
$$p_{\omega j} = x_j^{\omega}, \forall \omega \in \{1, 2, \ldots p\} \tag{4.20}$$

4. De acuerdo con el *Teorema 4.30,* se actualiza la matriz $\mathbf{Q}$ de tal forma que

$$q_{i\omega} = y_i^{\omega}, \forall \omega \in \{1, 2, \dots p\} \tag{4.21}$$

5. Por el *Teorema 4.31,* se genera un vector columna adicional $\mathbf{s} \in A^p$, que contiene en su $\omega-$ésima componente, la suma de los valores del $\omega-$ésimo renglón de la matriz $\mathbf{P}$, entonces:

$$s_{\omega} = \sum_{j=1}^{n} p_{\omega j} \tag{4.22}$$

Fase de recuperación de CAINN

Considere un patrón de entrada $\widetilde{\mathbf{x}}^{\omega} \in A^n$, con $\omega \in \{1, 2, \dots, p\}$ (si sucede que $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^{\omega}$, el patrón es del conjunto fundamental) y se lleva a cabo el siguiente procedimiento:

1. Por la *Definición 4.32*, realizamos la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$; es decir,

$$\left(\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}\right)_i = \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{\mathbf{x}}_j^{\omega}\right), \ \forall i \in \{1, 2, \dots, p\} \tag{4.23}$$

2. Se calcula el conjunto H, de tal forma que, por el *Teorema 4.35*, queda como:

$$H = i | \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right) = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^{\omega}\right)\right] \tag{4.24}$$

3. Por *Teorema 4.36* calculamos el código de clase $\mathbf{z}^{\omega} \in A^p$ de la siguiente manera:

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \ s_i = \bigwedge_{h \in H}^{p} s_h \\ 0 & \text{otro caso} \end{cases} \tag{4.25}$$

4. Una vez calculado el código de clase $\mathbf{z}^{\omega}$, se calcula el vector de salida $\mathbf{y}^{\omega} = \mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$, que por la *Definición 4.37* queda como:

$$y_i^{\omega} = \sum_{j=1}^{p} \beta\left(q_{ij}, z_j^{\omega}\right), \ \forall i \in \{1, 2, \dots, m\} \tag{4.26}$$

Se espera que el vector $\mathbf{y}^{\omega}$ sea precisamente el patrón de salida, del conjunto fundamental, correspondiente a $\mathbf{x}^{\omega}$.

4.3. Condiciones suficientes para la recuperación completa del conjunto fundamental

Lema 4.38 *Sea $\mathbf{x}^\omega \in A^n$ un patrón fundamental tomado arbitrariamente que se presenta como entrada a la matriz de aprendizaje $\mathbf{P}$. Durante la fase de recuperación de CAINN, se construye el vector t^ω, el cual sólo presenta $1's$ en cada componente cuyo índice i es el índice de las filas de $\mathbf{P}$ que fueron modificadas por un patrón $\mathbf{x}^\theta$ del conjunto fundamental que es mayor o igual a $\mathbf{x}^\omega$. Puesto de otra forma:*

$$t_i^\omega = 1 \longrightarrow \mathbf{x}^\theta \geq \mathbf{x}^\omega$$

Demostración. Dada la fase de recuperación de CAINN, sabemos que $t_i^\omega = 1 \longrightarrow \sum_{j=1}^{n} \beta\left(p_{ij}, x_j^\omega\right) = \bigvee_{h=1}^{k} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, x_j^\omega\right)\right]$; Como la operación $\beta\left(p_{ij}, x_j^\omega\right)$ no toma en cuenta los valores de las componentes p_{ij} para los cuales $x_j^\omega = 0$, el resultado máximo de $\sum_{j=1}^{n} \beta\left(p_{ij}, x_j^\omega\right)$ será para los patrones del conjunto fundamental que tengan $1's$ en las mismas componentes que $\mathbf{x}^\omega$, sin importar el valor de las demás componentes; es decir, el resultado máximo será para $\mathbf{x}^\omega$, así como para los patrones $\mathbf{x}^\theta$ que posean $1's$ en las posiciones donde $\mathbf{x}^\omega = 0$. Entonces, $t_i^\omega = 1$ también para los patrones que sean mayores que el correcto, así:

$$t_i^\omega = 1, \mathbf{x}^\theta \neq \mathbf{x}^\omega \longrightarrow \mathbf{x}^\theta > \mathbf{x}^\omega \tag{4.27}$$

■

Teorema 4.39 *Sea $\mathbf{P}$ la matriz de aprendizaje de CAINN, $\mathbf{x}^\mu \in A^n$ los patrones de entrada, $\mathbf{z}^\mu \in A^p$ sus correspondiente código de clase e $\mathbf{y}^\mu \in A^m$ los patrones de salida, del conjunto fundamental; con $n, m, p \in \mathbb{Z}^+; \mu = \{1, 2, \ldots, p\}$. Cuando se presenta un patrón fundamental $\mathbf{x}^\omega$ como entrada a la matriz $\mathbf{P}$, el algoritmo de CAINN siempre obtendrá el correspondiente patrón $\mathbf{y}^\omega$ sin ambigüedad. Dado que $\mathbf{y}^\omega$ se eligió de manera arbitraria, CAINN siempre recupera el conjunto fundamental completo.*

Demostración. Por el *Lema 4.20* sabemos que $\mathbf{z}^\omega$ puede ser calculado de la siguiente manera:

$$z_i^\omega = \begin{cases} 1 & \text{si } \; s_i = \bigwedge_{h \in H}^{k} s_h \\ 0 & \text{otro caso} \end{cases}$$

Para que el algoritmo fallase, tendría que existir algún valor s_i menor o igual a s_j cuando $t_i^\omega = 1 = t_j^\omega$; $i, j \in \{1, 2, \ldots, k\}, i \neq j$; esto puede ser demostrado por contradicción. Primero, asumamos que $\mathbf{z}^\omega$ es el código de clase correcto y que $\mathbf{z}^\theta$ es

algún código de clase erróneo. Cada uno con su propio valor del vector s_i denotado por s_{d^ω} y s_{d^θ}, respectivamente. Asumamos ahora lo contrario a lo que queremos demostrar

$$s_{d^\theta} \leq s_{d^\omega} \tag{4.28}$$

Por el *Lema 4.19* sabemos que s_i es la sumatoria de las componentes del vector $\mathbf{x}^\omega$, es decir U_{x^ω}. De tal forma que $s_{d^\omega} = U_{x^\omega}$ y $s_{d^\theta} = U_{x^\theta}$, así pues, tenemos:

$$U_{x^\theta} \leq U_{x^\omega} \tag{4.29}$$

Por otra parte, el *Lema 4.38* muestra que el vector de transición presenta $1's$ en cada componente cuyo índice i es el índice de las filas de $\mathbf{P}$ que fueron modificadas por $\mathbf{x}^\omega$ así como por un patrón fundamental $\mathbf{x}^\theta$ que es mayor o igual al patrón correcto $\mathbf{x}^\omega$, es decir que si $\mathbf{x}^\theta \neq \mathbf{x}^\omega$, entonces:

$$\mathbf{x}^\theta > \mathbf{x}^\omega \tag{4.30}$$

Por la *Definición 4.7* sabemos que $\mathbf{x}^\omega < \mathbf{x}^\theta \longleftrightarrow \forall i,\ x_i^\omega \leq x_i^\theta$ y $\exists j \mid x_j^\omega < x_j^\theta$. Esto significa que $\mathbf{x}^\theta$ tendrá al menos una componente estrictamente mayor que $\mathbf{x}^\omega$. Como $\mathbf{x}^\omega, \mathbf{x}^\theta \in A^n$, todas sus componente son binarias y solamente pueden tomar dos valores posibles: 0 o 1. Entonces, para que esto último pudiera ocurrir, es necesario que $x_j^\omega = 0 \wedge x_j^\theta = 1$. Así, $\mathbf{x}^\theta$ tendría al menos un 1 más que $\mathbf{x}^\omega$, y por la *Definición 4.3*, esto implica que:

$$U_{x^\theta} > U_{x^\omega} \tag{4.31}$$

De acuerdo con la ecuación 4.29: $U_{x^\theta} \leq U_{x^\omega}$, lo cual, teniendo en cuenta la expresión 4.31, es una contradicción. Así pues, $U_{x^\theta} \leq U_{x^\omega}$ no puede ser cierto, por lo tanto $U_{x^\theta} > U_{x^\omega}$ y consecuentemente, $s_{d^\theta} > s_{d^\omega}$, es decir que $\mathbf{z}^\omega$ siempre se recupera correctamente; de tal forma aseguramos la correcta recuperación de $\mathbf{y}^\omega$ debido a la forma en que éste se calcula. Como $\mathbf{x}^\omega$ fue elegido de manera arbitraria, se garantiza la correcta recuperación del conjunto fundamental completo. ■

4.4. Ejemplo del fundamento de CAINN

Con el fin de ilustrar los lemas y teoremas que dan fundamento al algoritmo CAINN, llevaremos a cabo el siguiente ejemplo. Para la realización de dicho ejemplo, los patrones de entrada y salida son dados como sigue:

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

Fase de aprendizaje

De acuerdo con los patrones mostrados previamente, podemos obtener los siguientes valores: $n = 4,\ m = 3,\ k = 16,\ p = 3$

1 Creación de las matrices nulas

$$\mathbf{P}_{k\times n} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{Q}_{m\times k} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2 Generación de los códigos de clase

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

3 Se actualizan las matrices de acuerdo con la regla de actualización. En el cálculo de la matriz $\mathbf{P}$ veremos reflejado lo demostrado por el *Lema 4.18*, mientras que para $\mathbf{Q}$ será lo establecido en el *Lema 4.23* lo que formalice su obtención.

$$\mathbf{P}_{k\times n} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{Q}_{m\times k} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4 Se genera el vector columna $\mathbf{s}$, que dado el *Lema 4.19* y la *Definición 4.3* queda como

$$\mathbf{s} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0 \\ 0 \\ 3 \\ 0 \end{pmatrix}$$

Fase de recuperación

En esta fase probaremos el algoritmo con los patrones del conjunto fundamental. En este caso, sea $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^1$

1 Se aplica la operación $\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^{\omega}$.

$$\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sigma_\beta \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$$

2 Cálculo del vector de transición $\mathbf{t}^{\omega}$. En este ejemplo podemos apreciar lo establecido en el *Lema 4.38*.

$$\mathbf{t}^{\omega} = \begin{pmatrix} \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(2,2) \\ \alpha_g(2,2) \\ \alpha_g(0,2) \\ \alpha_g(0,2) \\ \alpha_g(2,2) \\ \alpha_g(0,2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

3 Se define el conjunto H

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij}\widetilde{x}_j^{\omega} \right] \right\}, \text{ de tal forma que}$$

$$H = \{11, 12, 15\}$$

4 Se calcula el código de clase $\mathbf{z}^{\omega}$. Como se dijo en la demostración del *Lema 4.20*, entonces:

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge_{h \in H}^{k} s_h \\ 0 & \text{otro caso} \end{cases}$$

$$\begin{array}{ll} z_1^{\omega} = 0 & z_9^{\omega} = 0 \\ z_2^{\omega} = 0 & z_{10}^{\omega} = 0 \\ z_3^{\omega} = 0 & z_{11}^{\omega} = 1 \\ z_4^{\omega} = 0 & z_{12}^{\omega} = 0 \\ z_5^{\omega} = 0 & z_{13}^{\omega} = 0 \\ z_6^{\omega} = 0 & z_{14}^{\omega} = 0 \\ z_7^{\omega} = 0 & z_{15}^{\omega} = 0 \\ z_8^{\omega} = 0 & z_{16}^{\omega} = 0 \end{array}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

5 Se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

6 Por último, se calcula el vector de salida $\mathbf{y}^{\omega}$, que como dice el *Teorema 4.39*, siempre se recuperará correctamente

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta \left(q_{ij}, z_j^{\omega} \right), \bigvee_{h=1}^{m} \beta \left(q_{hj}, z_j^{\omega} \right) \right), \qquad \forall i \in \{1, 2, \ldots, m\}$$

$$\begin{aligned} y_1^{\omega} &= \alpha_g \left(1, \bigvee (1,0,0)\right) = \alpha_g (1,1) = 1 \\ y_2^{\omega} &= \alpha_g \left(0, \bigvee (1,0,0)\right) = \alpha_g (0,1) = 0 \\ y_3^{\omega} &= \alpha_g \left(0, \bigvee (1,0,0)\right) = \alpha_g (0,1) = 0 \end{aligned}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \mathbf{y}^1 \therefore$ la recuperación es correcta.

Tomemos ahora el caso para el cual el vector $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^2$

1 Se aplica la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$

$$\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 3 \\ 0 \end{pmatrix}$$

2 Cálculo del vector de transición $\mathbf{t}^{\omega}$. En este ejemplo podemos apreciar lo establecido en el *Lema 4.38*.

$$\mathbf{t}^{\omega} = \begin{pmatrix} \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(2,3) \\ \alpha_g(2,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(3,3) \\ \alpha_g(0,3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

3 Se define el conjunto H

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij}\widetilde{x}_j^{\omega} \right] \right\}, \text{ de tal forma que}$$

$$H = \{15\}$$

4 Se calcula el código de clase $\mathbf{z}^{\omega}$. Como se dijo en la demostración del *Lema 4.20*, entonces:

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge_{h \in H}^{k} s_h \\ 0 & \text{otro caso} \end{cases}$$

$$\begin{array}{ll} z_1^{\omega} = 0 & z_9^{\omega} = 0 \\ z_2^{\omega} = 0 & z_{10}^{\omega} = 0 \\ z_3^{\omega} = 0 & z_{11}^{\omega} = 0 \\ z_4^{\omega} = 0 & z_{12}^{\omega} = 0 \\ z_5^{\omega} = 0 & z_{13}^{\omega} = 0 \\ z_6^{\omega} = 0 & z_{14}^{\omega} = 0 \\ z_7^{\omega} = 0 & z_{15}^{\omega} = 1 \\ z_8^{\omega} = 0 & z_{16}^{\omega} = 0 \end{array}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

5 Se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

6 Por último, se calcula el vector de salida $\mathbf{y}^{\omega}$, que como dice el *Teorema 4.39*, siempre se recuperará correctamente

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta \left(q_{ij}, z_j^{\omega} \right), \bigvee_{h=1}^{m} \beta \left(q_{hj}, z_j^{\omega} \right) \right), \qquad \forall i \in \{1, 2, \ldots, m\}$$

$$\begin{aligned} y_1^{\omega} &= \alpha_g \left(1, \bigvee (1,1,0)\right) = \alpha_g (1,1) = 1 \\ y_2^{\omega} &= \alpha_g \left(1, \bigvee (1,1,0)\right) = \alpha_g (1,1) = 1 \\ y_3^{\omega} &= \alpha_g \left(0, \bigvee (1,1,0)\right) = \alpha_g (0,1) = 0 \end{aligned}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \mathbf{y}^2 \therefore$ la recuperación es correcta.

Ahora sólo resta realizar la fase de recuperación para el caso en que el vector $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^3$

1 Se aplica la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$

$$\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$$

2 Cálculo del vector de transición $\mathbf{t}^{\omega}$. En este ejemplo podemos apreciar lo establecido en el *Lema 4.38*.

$$\mathbf{t}^{\omega} = \begin{pmatrix} \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(2,3) \\ \alpha_g(3,3) \\ \alpha_g(0,3) \\ \alpha_g(0,3) \\ \alpha_g(2,3) \\ \alpha_g(0,3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3 Se define el conjunto H

$$H = \left\{ h \mid \sum_{j=1}^{n} p_{hj}\widetilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left[\sum_{j=1}^{n} p_{ij}\widetilde{x}_j^{\omega} \right] \right\}, \text{ de tal forma que}$$

$$H = \{12\}$$

4 Se calcula el código de clase $\mathbf{z}^{\omega}$. Como se dijo en la demostración del *Lema 4.20*, entonces:

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge_{h \in H}^{k} s_h \\ 0 & \text{otro caso} \end{cases}$$

$$\begin{array}{ll} z_1^{\omega} = 0 & z_9^{\omega} = 0 \\ z_2^{\omega} = 0 & z_{10}^{\omega} = 0 \\ z_3^{\omega} = 0 & z_{11}^{\omega} = 0 \\ z_4^{\omega} = 0 & z_{12}^{\omega} = 1 \\ z_5^{\omega} = 0 & z_{13}^{\omega} = 0 \\ z_6^{\omega} = 0 & z_{14}^{\omega} = 0 \\ z_7^{\omega} = 0 & z_{15}^{\omega} = 0 \\ z_8^{\omega} = 0 & z_{16}^{\omega} = 0 \end{array}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

5 Se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

6 Por último, se calcula el vector de salida $\mathbf{y}^{\omega}$, que como dice el *Teorema 4.39*, siempre se recuperará correctamente

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta \left(q_{ij}, z_j^{\omega} \right), \bigvee_{h=1}^{m} \beta \left(q_{hj}, z_j^{\omega} \right) \right), \qquad \forall i \in \{1, 2, \ldots, m\}$$

$$\begin{aligned} y_1^{\omega} &= \alpha_g \left(1, \bigvee (1,0,1)\right) = \alpha_g (1,1) = 1 \\ y_2^{\omega} &= \alpha_g \left(0, \bigvee (1,0,1)\right) = \alpha_g (0,1) = 0 \\ y_3^{\omega} &= \alpha_g \left(1, \bigvee (1,0,1)\right) = \alpha_g (1,1) = 1 \end{aligned}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{y}^3 \therefore$ la recuperación es correcta.

4.5. Ejemplo del modelo CAINN Optimizado

Con el fin de traer a un nivel ilustrativo todo el anterior bagaje matemático desarrollado sobre el algoritmo CAINN, llevaremos a cabo el siguiente ejemplo. Para la realización de este ejemplo, los patrones de entrada y salida son dados como sigue:

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

Fase de aprendizaje

De acuerdo con los patrones mostrados previamente, podemos obtener los siguientes valores: $n = 4,\ m = 3,\ p = 3$

1 Creación de las matrices nulas $\mathbf{P} = [p_{ij}]_{p \times n} = 0$ y $\mathbf{Q} = [q_{ij}]_{m \times p} = 0$.

$$\mathbf{P}_{k \times n} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{Q}_{m \times k} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

2 Generación de los códigos de clase, como lo indica la *Definición 4.26*.

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

3 De acuerdo con el *Teorema 4.29* y el *Teorema 4.30*, se actualizan las matrices de acuerdo con la regla de actualización propuesta

$$\mathbf{P}_{p\times n} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{Q}_{m\times p} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

4 Por el *Teorema 4.31*, generamos el vector columna $\mathbf{s}$

$$\mathbf{s} = \begin{pmatrix} 2 \\ 3 \\ 3 \end{pmatrix}$$

Fase de recuperación

En esta fase probaremos el algoritmo con los patrones del conjunto fundamental. En este caso, sea $\widetilde{\mathbf{x}}^\omega = \mathbf{x}^1$

1 Se aplica la operación $\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^\omega$, dada la *Definición 4.32*.

$$\mathbf{P}\sigma_\beta\widetilde{\mathbf{x}}^\omega = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \sigma_\beta \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$$

2 Se define el conjunto H como dice el *Teorema 4.35*:

$$H = i| \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^\omega\right) = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^\omega\right)\right]$$

de tal forma que

$$H = \{1, 2, 3\}$$

3 Por el *Teorema 4.36* se calcula el código de clase $\mathbf{z}^{\omega}$

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge\limits_{h \in H}^{p} s_h \\ 0 & \text{otro caso} \end{cases}$$

$$\begin{aligned} z_1^{\omega} &= 1 \\ z_2^{\omega} &= 0 \\ z_3^{\omega} &= 0 \end{aligned}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

4 Finalmente, dada la *Definición 4.37*, se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\mathbf{y}^{\omega} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; \sigma_{\beta} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \mathbf{y}^1 \therefore$ la recuperación es correcta.

Tomemos ahora el caso para el cual el vector $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^2$

1 Se aplica la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$, dada la *Definición 4.32*.

$$\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \; \sigma_{\beta} \; \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix}$$

2 Se define el conjunto H como dice el *Teorema 4.35*:

$$H = i | \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right) = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^{\omega}\right) \right]$$

de tal forma que

$$H = \{2\}$$

3 Por el *Teorema 4.36* se calcula el código de clase $\mathbf{z}^{\omega}$

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \; s_i = \bigwedge_{h \in H}^{p} s_h \\ 0 & \text{otro caso} \end{cases}$$

$$\begin{aligned} z_1^{\omega} &= 0 \\ z_2^{\omega} &= 1 \\ z_3^{\omega} &= 0 \end{aligned}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

4 Finalmente, dada la *Definición 4.37*, se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\mathbf{y}^{\omega} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; \sigma_{\beta} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \mathbf{y}^2 \therefore$ la recuperación es correcta.

Ahora sólo resta realizar la fase de recuperación para el caso en que el vector $\widetilde{\mathbf{x}}^{\omega} = \mathbf{x}^3$

1 Se aplica la operación $\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega}$, dada la *Definición 4.32*.

$$\mathbf{P}\sigma_{\beta}\widetilde{\mathbf{x}}^{\omega} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \; \sigma_{\beta} \; \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 3 \end{pmatrix}$$

2 Se define el conjunto H como dice el *Teorema 4.35*:

$$H = i| \sum_{j=1}^{n} \beta\left(p_{ij}, \widetilde{x}_j^{\omega}\right) = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} \beta\left(p_{hj}, \widetilde{x}_j^{\omega}\right)\right]$$

de tal forma que

$$H = \{3\}$$

3 Por el *Teorema 4.36* se calcula el código de clase $\mathbf{z}^{\omega}$

$$z_i^{\omega} = \begin{cases} 1 & \text{si } \ s_i = \bigwedge_{h \in H}^{p} s_h \\ 0 & \text{otro caso} \end{cases}$$

$$\begin{aligned} z_1^{\omega} &= 0 \\ z_2^{\omega} &= 0 \\ z_3^{\omega} &= 1 \end{aligned}$$

De lo anterior tenemos que

$$\mathbf{z}^{\omega} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

4 Finalmente, dada la *Definición 4.37*, se ejecuta la operación $\mathbf{Q}\sigma_{\beta}\mathbf{z}^{\omega}$

$$\mathbf{y}^{\omega} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \ \sigma_{\beta} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

De tal suerte que $\mathbf{y}^{\omega} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{y}^3 \therefore$ la recuperación es correcta.

Capítulo 5

Resultados y discusión

Es en este capítulo donde se pone en evidencia la eficacia de la red neuronal Alfa-Beta sin pesos al recuperar y clasificar patrones. Se llevaron a cabo una serie de experimentos que ilustran el comportamiento de este modelo.

En la sección 5.1, se describirán brevemente las bases de datos que fueron utilizadas para esta fase experimental. Dichas bases de datos, están disponibles al público en general en el repositorio de bases de datos de la Universidad de California en Irvine (UCI Machine Learning Repository) [67].

En la sección 5.2, se muestran los resultados obtenidos para los índices de recuperación de patrones del conjunto fundamental.

De igual forma, en la sección 5.3 se presenta el desempeño alcanzado por el algoritmo al momento de realizar clasificación de patrones.

5.1. Bases de datos y equipo utilizado

Durante la realización de esta fase de experimentación, se usaron dos conjuntos de datos que fueron tomados de [67]. En estas bases de datos, cada renglón representa un patrón, y cada patrón está dividido en columnas que indican la clase a la cual pertenece cada uno de éstos, así como sus características propias.

5.1.1. Iris Plant Data Set

Esta es, probablemente, la base de datos más conocida que se encuentra dentro de la literatura del reconocimiento de patrones. El conjunto de datos contiene 3 clases de 50 instancias cada una, donde cada clase hace referencia a un tipo de planta de iris: Iris Setosa, Iris Versicolor e Iris Virginica. Cada patrón que representa una planta de Iris tiene 4 atributos mas uno que representa la clase. En esta base de datos una clase es linealmente separable de las otras dos; pero estas últimas no son linealmente

separables entre sí, los rasgos que representan el largo y ancho de los pétalos están altamente correlacionados. La *Tabla 5.1* nos muestra la forma en cómo se compone cada patrón en este conjunto de datos.

Característica	Descripción
1	Clase: 1: Iris Setosa, 2:Iris Versicolor, 3:Iris Virginica
2	Largo del sépalo en centímetros
3	Ancho del sépalo en centímetros
4	Largo del pétalo en centímetros
5	Ancho del pétalo en centímetros

Tabla 5.1: Características del primer conjunto de datos usado

5.1.2. Contraceptive Method Choice Data Set (CMC)

Este conjunto de datos consiste en un subconjunto de la Encuesta Nacional de Control Anticonceptivo en Indonesia en 1987. Cada uno de los patrones representa información acerca de una mujer casada que no está embarazada, o por lo menos ignora estarlo, al momento de la entrevista. Esta base de datos fue utilizada para predecir la elección en el uso de métodos anticonceptivos en mujeres, tomando en cuenta sus características socio-económicas y demográficas. Estos datos se agrupan en 3 clases: la clase 1 para quienes no usan algún método con 629 patrones; la clase 2 para quienes usan un método a largo plazo con 333 patrones y la clase 3 para aquellas que usan un método a corto plazo con 511 patrones, sumando un total de 1473 patrones cuyas características son ilustradas en la *Tabla 5.2*.

Característica	Descripción
1	Clase
2	Edad
3	Grado educativo
4	Grado educativo de la pareja
5	Cantidad de niños ya concebidos
6	Religión
7	Trabajo
8	Ocupación de la pareja
9	Índice del estándar de vida
10	Riesgo de vida

Tabla 5.2: Características del segundo conjunto de datos usado

5.1.3. Equipo de cómputo utilizado

Los experimentos se realizaron en una computadora personal, que cuenta con un procesador Intel Pentrium D a 3.40 GHz, 2048 MBytes de memoria RAM y 109.5 GBytes de espacio libre en disco duro. Se usó el sistema operativo Windows XP Profesional de Microsoft y se utilizó un software diseñado e implementado en Borland C++ Builder 6.0, ex-profeso para este fin.

5.2. Recuperación de patrones

Para la realización de los experimentos, se tomaron en cuenta los mismos aspectos para la recuperación de patrones que han sido usados en la literatura, donde esencialmente lo que interesa es el índice de recuperación correcta.

5.2.1. Recuperación del conjunto fundamental completo

En primera instancia se probó el modelo de red neuronal Alfa-Beta sin pesos sobre el conjunto fundamental completo para tratar de recuperarlo. Esto se llevó a cabo al aprender el primer patrón y tratar de recuperarlo, aprender el segundo patrón e intentar recuperar los dos patrones aprendidos, continuando de la misma manera hasta haber aprendido e intentado recuperar el conjunto de aprendizaje completo.

Con el fin de obtener resultados confiables del índice de recuperación que se obtiene en ambos conjuntos de datos, este procedimiento se realizó 50 veces para cada conjunto de prueba, y además, en cada una de estas ocasiones se seleccionó de manera aleatoria el orden de los patrones del conjunto fundamental. De tal manera que para la Iris Plant se realizaron 7,500 intentos de recuperación y para la CMC se realizaron 73,650 intentos.

Una vez que este procedimiento de aprendizaje y recuperación ha sido realizado, se reportan los resultados en la *Tabla 5.3*, en la cual se obtiene el promedio de los intentos de recuperación, para cada uno de los conjuntos de prueba.

	Iris	CMC
ADAM	9 %	8 %
CAINN	100 %	100 %

Tabla 5.3: Índice de recuperación del conjunto fundamental

Es por demás claro que el desempeño de CAINN es altamente superior al alcanzado por el modelo ADAM.

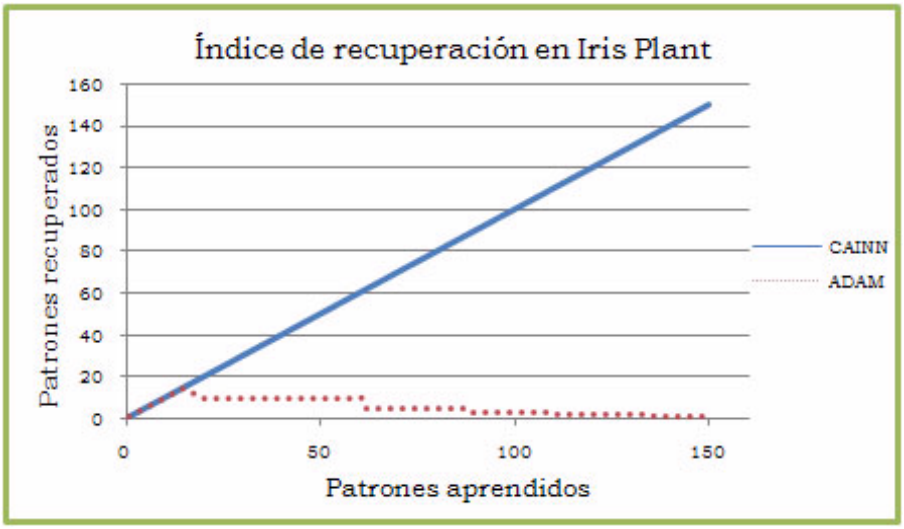

Figura 5.1: *Índice de recuperación en Iris Plant*

Los resultados de la *Tabla 5.3*, se muestran gráficamente en la *Figura 5.1* así como en la *Figura 5.2*.

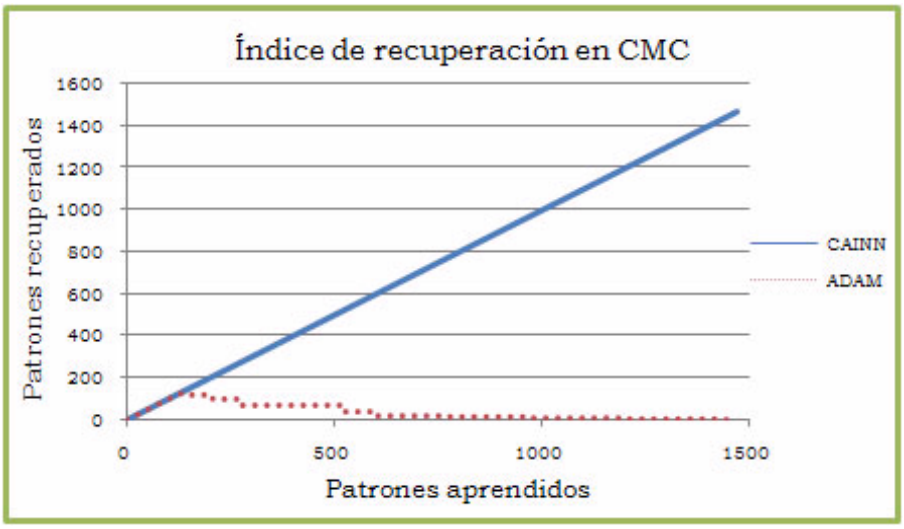

Figura 5.2: *Índice de recuperación en Contraceptive Method Choice*

De igual manera, pudimos observar que ADAM sufre de problemas de saturación temprana, provocando que al realizar la fase de recuperación existan condiciones de ambigüedad para los patrones involucrados, haciendo imposible su recuperación, dado

que una recuperación correcta se da al recuperar todos los bits que conforman un patrón.

5.2.2. Recuperación del conjunto fundamental por particiones

Para realizar una segunda estimación del desempeño del modelo CAINN, se llevó a cabo la recuperación del conjunto fundamental por particiones; esto consistió en aprender el 70 % del conjunto fundamental e intentar recuperar el 30 % restante.

Con el fin de obtener resultados confiables del índice de recuperación que se obtiene en ambos conjuntos de datos, este procedimiento se realizó 50 veces para cada conjunto de prueba seleccionando los patrones de manera aleatoria en cada una de estas ocasiones; los resultados son reportados obteniendo el promedio de los intentos de recuperación para cada conjunto de prueba, y están reflejados en la *Tabla 5.4*.

	Iris	CMC
ADAM	17 %	12 %
CAINN	74 %	72 %

Tabla 5.4: Índice de recuperación por particiones 70/30

Los resultados anteriormente expuestos, se muestran de manera gráfica en la *Figura 5.3* y en la *Figura 5.4*.

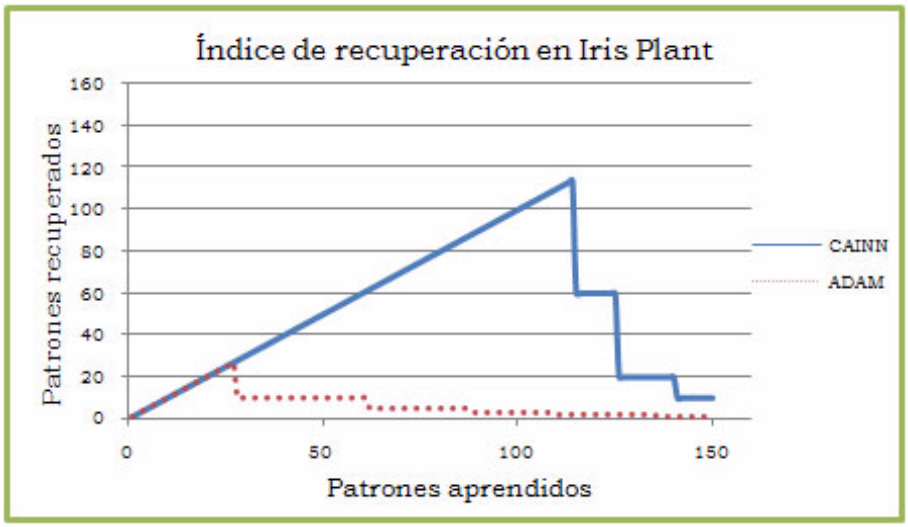

Figura 5.3: *Índice de recuperación en Iris Plant (70/30)*

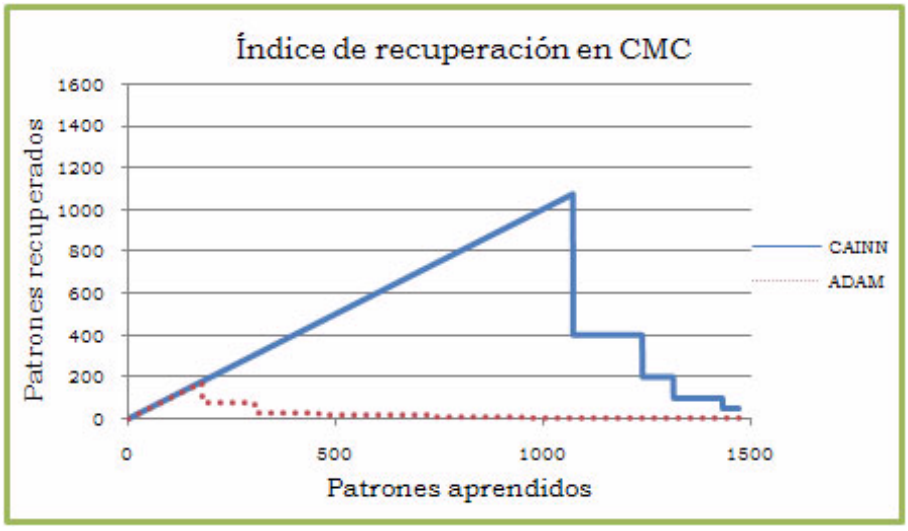

Figura 5.2: *Índice de recuperación en Contraceptive Method Choice (70/30)*

5.3. Clasificación de patrones

Para la realización de los experimentos, se tomaron en cuenta los mismos aspectos para la clasificación de patrones que han sido usados en la literatura, donde esencialmente lo que interesa es el índice de clasificación correcta.

5.3.1. Clasificación del conjunto fundamental completo

Para realizar esos experimentos, se tomaron las mismas bases de datos utilizadas para la recuperación, con la diferencia de que para la clasificación, cada uno de los experimentos se llevó a cabo de la siguiente manera: primeramente, se tomaron de manera aleatoria el mismo número de patrones de entrada para cada clase; esto con la finalidad de obtener un conjunto de prueba balanceado. Posteriormente, se tomo dicho conjunto de prueba para su clasificación.

Con el fin de obtener un índice de clasificación confiable, este procedimiento se realizó 50 veces para ambos conjuntos de datos, y los resultados son mostrados en la *Tabla 5.5*. Es pertinente mencionar que, si bien los índices de clasificación en el modelo ADAM mejoraron notablemente con respecto de sus resultados obtenidos para el proceso de recuperación, los resultados para el modelo CAINN nuevamente aventajan al modelo ADAM al realizar la clasificación correcta del conjunto fundamental completo.

	Iris	CMC
ADAM	70 %	65 %
CAINN	100 %	100 %

Tabla 5.5: Índice de clasificación del conjunto fundamental

Los resultados anteriormente expuestos, se muestran de manera gráfica en la *Figura 5.5* y en la *Figura 5.6*.

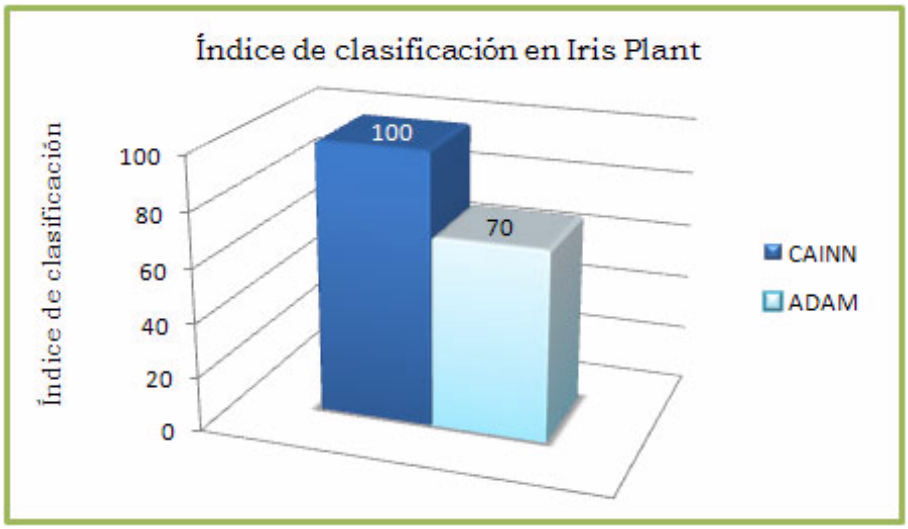

Figura 5.5: *Índice de clasificación en Iris Plant*

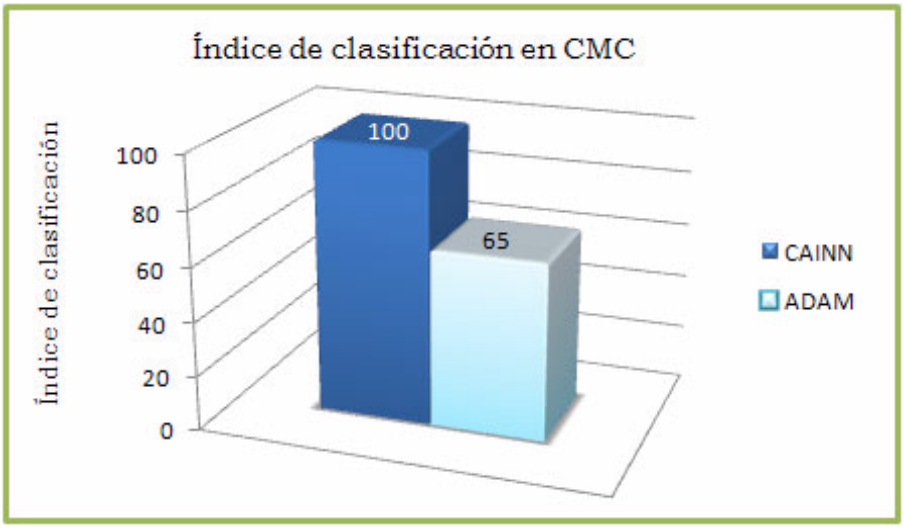

Figura 5.5: *Índice de clasificación en Contraceptive Method Choice*

5.3.2. Clasificación del conjunto fundamental por particiones

Para realizar la cuarta estimación del desempeño del modelo de red neuronal Alfa-Beta sin pesos, se llevó a cabo la clasificación del conjunto fundamental por particiones; esto fue llevado a cabo al aprender el 70 % del conjunto fundamental e intentar clasificar el 30 % restante.

Con la finalidad de obtener resultados confiables para índice de clasificación que se obtiene en ambos conjuntos de datos, este procedimiento se realizó 50 veces para cada conjunto de prueba seleccionando los patrones de manera aleatoria en cada una de estas ocasiones; los resultados son reportados obteniendo el promedio de los intentos de clasificación para cada conjunto de prueba, y son mostrados en la *Tabla 5.6*.

	Iris	CMC
ADAM	58 %	42 %
CAINN	89 %	81 %

Tabla 5.6: Índice de clasificación por particiones 70/30

Los resultados expuestos en la *Tabla 5.6* se muestran de manera gráfica en la *Figura 5.7* y en la *Figura 5.8*.

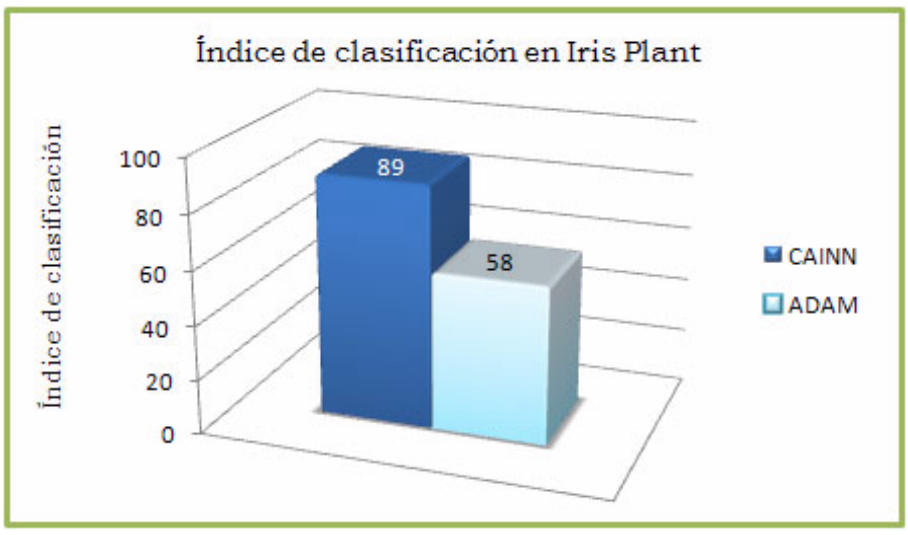

Figura 5.7: *Índice de clasificación en Iris Plant (70/30)*

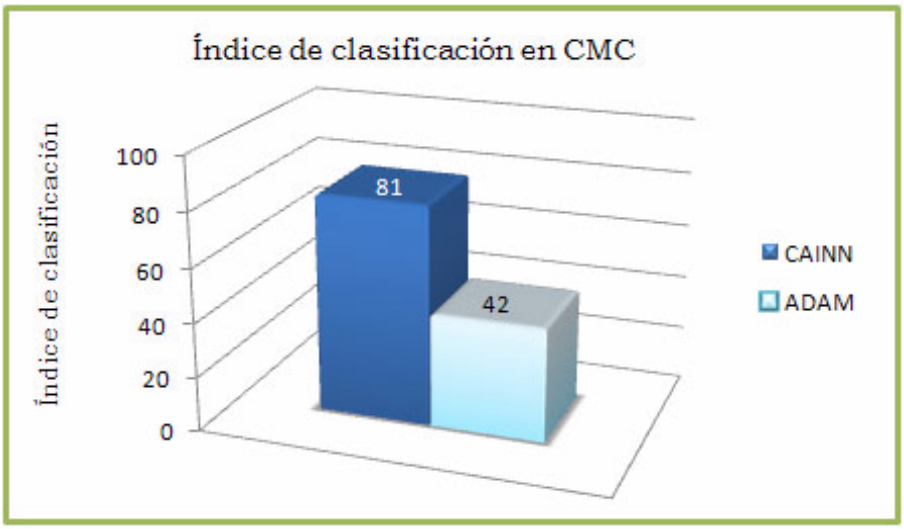

Figura 5.8: *Índice de clasificación en Contraceptive Method Choice (70/30)*

Una vez teniendo los resultados anteriores, se realizó una comparación con algunos de los modelos reportados en la literatura. Para ello, en la *Tabla 5.7* hemos incluido los resultados obtenidos anteriormente, así como los que han sido reportados en [68] para Iris, y en [69] y [70] para Iris y CMC.

	Iris	CMC
ADAM	58 %	42 %
CAINN	89 %	81 %
RAMP [68]	97 %	-
STEP [68]	77 %	-
SVM [68][69]	96 %	54 %
KNN [68][69]	96 %	48 %
MLP [68][69]	95 %	53 %
RM [69]	97 %	55 %
C4.5 [70]	94 %	63 %
C4.5+m [70]	93 %	66 %

Tabla 5.7: Comparación de los índices de clasificación

Con respecto a la base de datos de Iris Plant, es claro que CAINN supera ampliamente a ADAM en cuanto al índice de clasificación, superando también a la red neuronal STEP, aunque ante otros clasificadores no presente tan buen desempeño (RAMP, SVM, kNN, MLP, RM, C4.5 y C4.5+m), se aproxima a sus resultados. Esta desventaja mostrada con respecto al último conjunto de clasificadores es resarcida en la base de datos CMC, en donde supera a todos los clasificadores incluidos en la tabla por un margen de 15 % de diferencia comparado con el clasificador C4.5+m, cuyos

resultados fueron los más aproximados a los de CAINN. Por supuesto, este comportamiento no es para nada extraño, puesto que el teorema del *No Free Lunch* [71][72] nos indica que aun cuando un algoritmo de clasificación sea muy bueno con ciertos tipos de problemas, puede o debe no ser tan bueno con otros. Los resultados de esta comparación se grafican en la *Figura 5.9*.

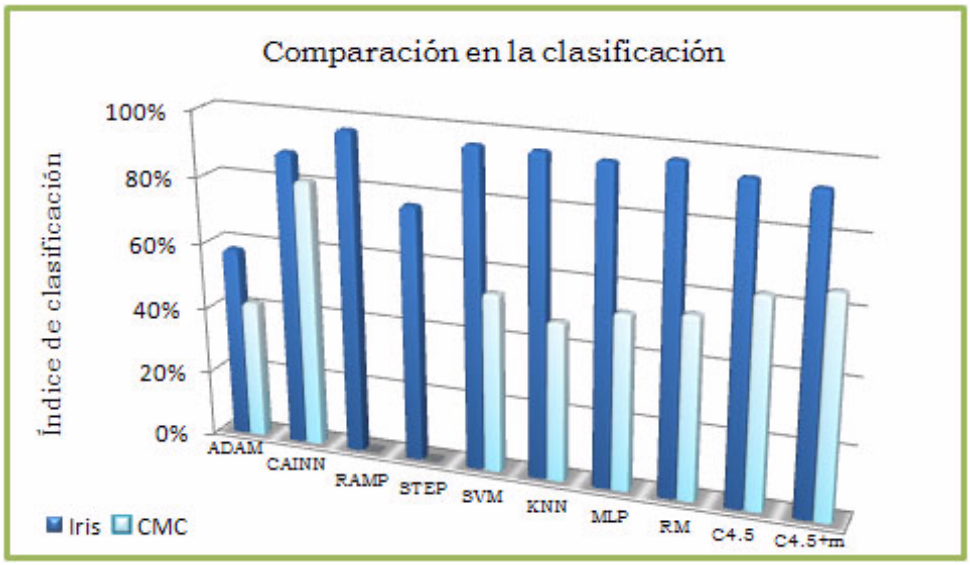

Figura 5.9: *Comparación en el índice de clasificación con otros clasificadores*

Por otra parte, cabe mencionar que todos los algoritmos presentes en la *Tabla 5.8*, y que muestran un mejor desempeño que CAINN con respecto a la base de datos Iris Plant, tienen también una mayor complejidad siendo varios de ellos algoritmos iterativos. Así pues, a pesar de ofrecer un índice de clasificación alrededor de un 5 % menor que dichos clasificadores, CAINN ofrece las ventajas de ser un algoritmo *one–shot* y relativamente más simple.

Capítulo 6

Conclusiones y trabajo a futuro

Este último capítulo consta de dos partes. Por un lado se presentan las conclusiones a las que se ha llegado a partir de los resultados obtenidos en el presente trabajo de tesis, las cuales reflejan la consumación de los objetivos planteados al inicio de esta investigación.

Así también, hemos de proporcionar algunos posibles trabajos que pudieran desarrollarse en pos de continuar con las ideas propuestas en el presente trabajo, dando la pauta para que futuros investigadores tengan elementos para seguir desarrollando esta línea de investigación.

6.1. Conclusiones

1. La proposición y demostración de un total de 9 definiciones, 7 lemas y 10 teoremas cuya finalidad es servir como fundamento para la caracterización y formalización del algoritmo de redes neuronales Alfa-Beta sin pesos.

2. Se establecieron las condiciones suficientes que exhiben la recuperación de patrones del conjunto fundamental.

3. Se demostró mediante lemas y teoremas que la red neuronal Alfa-Beta sin pesos denominada CAINN es capaz de recuperar y clasificar de forma correcta el conjunto fundamental completo.

4. El desempeño que mostró este modelo, al realizarse los experimentos y estudios comparativos sobre las bases de datos Iris Plant y CMC, es superior al alcanzado por el modelo ADAM y otros modelos inmersos en la literatura.

5. Propuesta y fundamentación de un algoritmo alternativo que optimiza el funcionamiento de CAINN, tanto en la fase de aprendizaje como en la de recuperación.

6. Se redujo la dimensión desde 2^n hasta p de las matrices y vectores que son utilizados en CAINN, siendo n la dimensión de los patrones de entrada y p el número de asociaciones en los patrones fundamentales.

7. La implicación directa de la conclusión anterior es la disminución en el tiempo y operaciones requeridas para llevar a cabo las fases de aprendizaje y recuperación del algoritmo.

6.2. Trabajo a futuro

1. Realizar experimentos de manera exhaustiva comparando los resultados obtenidos con aquellos que presentados por otros algoritmos recuperadores y clasificadores de patrones.

2. Experimentar con modificaciones en los dominios A y B; esto con la finalidad de buscar nuevos modelos que se basen en el marco teórico creado por los operadores α y β.

3. La generalización del operador β para que, al igual que el operador α, también acepte entradas en el dominio de los números reales $\mathbb{R}$.

4. La matematización del comportamiento del algoritmo ante patrones desconocidos.

5. Realizar un estudio exhaustivo cuyo objetivo sea cuantificar el comportamiento del modelo ante alteraciones en patrones fundamentales.

6. Llevar a cabo una comparación, en cuanto a tiempo y densidad aritmética, entre el modelo CAINN original y el propuesto en este trabajo.

Bibliografía

[1] Argüelles-Cruz, A. (2007). Redes Neuronales Alfa-Beta Sin Pesos: Teoría y Factibilidad de Implementación. *Tesis Doctoral*. Centro de Investigación en Computación, ciudad de México.

[2] Yáñez-Márquez, C. (2002). Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios. *Tesis Doctoral*, Centro de Investigación en Computación, ciudad de México.

[3] Enquist, M. & Ghirlanda, S. (2005). Neural Networks and Animal Behaviour. (1st edition) Princeton: *Princeton University Press*.

[4] McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.

[5] Hebb, D. O. (1949). Organization of Behaviour. *Wiley*.

[6] Rosenblatt, Frank (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65, 6, 346-408.

[7] Widrow, Bernard, & Hoff, Marcian, E. (1960) Adaptive Switching Circuits, *1960 IRE WESCON Convention Record*, New York: IRE pp. 96-104.

[8] Minsky, M. & Papert, S. (1988). Perceptrons, *Cambridge: MIT Press*.

[9] Steinbuch, K. V. (1961). Die Lernmatrix. *Kybernetik*, 1, 1, pp. 36-45.

[10] Willshaw, D., Buneman, O. & Longuet-Higgins, H. (1969). Non-holographic associative memory, *Nature*, 222, 960-962.

[11] Anderson, J. A. (1972). A simple neural network generating an interactive memory, *Mathematical Biosciences*, 14, 197-220.

[12] Kohonen, T. (1972). Correlation matrix memories, *IEEE Transactions on Computers*, C-21, 4, 353-359.

[13] Nakano, K. (1972). Associatron-A model of associative memory, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2, 3, 380-388.

[14] Amari, S. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements, *IEEE Transactions on Computers*, C-21, 11, 1197-1206.

[15] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences of the USA* (pp. 2554-2558).

[16] Hopfield, J.J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons, *Proceedings of the National Academy of Sciences*, 81, 3088-3092.

[17] Ritter, G. X., Sussner, P. & Diaz de Leon, J. L. (1998). Morphological associative memories. *IEEE Transactions on Neural Networks*, 9, pp. 281-293.

[18] Bledsoe, W.W. & Browning, I. (1959). Pattern Recognition and Reading by machine. In *Proc.Eastern Joint Computer Conference* (pp. 225-232).

[19] Bledsoe, W. W. (1952). Some Aspects of Covering Theory. *Proceedings of the American Mathematical Society*, 3(5), 804-812

[20] Aleksander, I., W.V.Thomas & P.A.Bowden (1984). WISARD, a Radical Step Forward in Image Recognition. *Sensor Review*, 4, 3, 120-124.

[21] Kanerva, P. (1988). Sparse Distributed Memory. *Cambridge, MIT Press.*

[22] Braspenning, P. J., Thuijsman, F. & Weijters, A. J. (1995). Artificial Neural Networks: An Introduction to ANN Theory and Practice. (vols. 931/1995) *Utrecht: Springer Berlin / Heidelberg.*

[23] Ludermir, T. B. et. al. (1999). Weightless Neural Models: A Review of Current and Past Works. *Neural Computing Surveys*, 2, 41-61.

[24] Aleksander, I. & Stonham, T. J. (1979). Guide to pattern recognition using random-access memories. *Computers and Digital Techniques*, 2, 29-40.

[25] Aleksander, I. (1983). Emergent intelligent properties of progressively structured pattern recognition nets. *Pattern Recognition Letters*, 1:375-384.

[26] Christensen, S. S., Andersen, A. W., Jorgensen, T. M., & Liisberg, C. (1996). Visual guidance of a pig evisceration robot using neural networks. *Pattern Recognition Letters*, 17(4):345-355.

[27] Hepplewhite, L., & Stonham. T. J. (1997). N-tuple texture recognition and the zero crossing sketch. *IEEE Electronics Letters*, 33(1):45-46.

[28] Jorgensen, T. M. (1997). Classification of hadwritten digits using a RAM neural net architecture. *International Journal of Neural Systems*, 8(1):17-25.

[29] McCauley, J. D., Thabe, B. R., &Whittaker, A. D. (1994). Fast estimatin in beef ultrasound images using texture and adaptive logic networks. *Transactions of ASAE*, 37(3):997-102.

[30] Ramanan, S., Petersen, R. S., Clarkson, T. G., & Taylor, J. G. (1995). pRAM nets for detection of small targets in sequence of infra-red images. *Neural Networks*, 1227-1337.

[31] Rohwer, R., & Morciniec, M. (1995). A theoretical and experimental account of n-tuple classifier performance. *Neural computing*, 8:629-642.

[32] Wang, Y. S., Griffiths, B. J., & Wilkie, B. A. (1996). A novel system for coloured object recognition. *Computers in industry*, 32(1):69-77.

[33] Aleksander, I. (1990). An introduction to neural computing. *Chapman and Hall, London.*

[34] Filho, E. C. D. B., Fairhurst, M. C., Bisset, D. L. (1992). Analysis of Saturation Problem in RAM-based Neural Networks, *Electronics Letters*, *vol. 28, no. 4*, pp. 345-346.

[35] Aleksander, I. (1989). Ideal Neurons for neural computers. Parallel processing in Neural Systems and Computers, *Elsevier Science*. pp. 225-228.

[36] Aleksander, I., Clarke, T. J. & Braga, A. P. (1994). Binary Neural Systems: combining weighted and weightless properties. *IEEE Intelligent Systems Engineering*, 3:211-221.

[37] Rummelhart, D. E. & McClelland, J. L. (1986). Parallel Distributed Processing, vol. 1, *Foundations. MIT Press.*

[38] Aleksander, I. & Morton, H. (1991). General Neural Unit: retrieval performance. *IEEE Electronics Letters*, 27:1776-1778.

[39] Kan, W. K. & Aleksander, I. (1987). A probabilistic logic neuron network for associative learning. In *proceedings of the IEEE First International Conference on Neural Network* (ICNN87), vol 2, pp. 541-548, San Diego, California.

[40] Al-Alawi, R. (2003). FPGA Implementation of a Pyramidal Weightless Neural Networks Learning System. *International journal of Neural Systems*, 13, 4, pp. 225-237.

[41] Aleksander, I. (1989). Canonical neural nets based on logic nodes. In *First IEE International Conference on Artificial Neural Networks*, London: IEE, pp. 110-114.

[42] Austin, J. (1987). The Design and Application of Associative Memories for Scene Analysis. *Dept. Electrical Engineering. Brunei University.*

[43] Austin, J. & Stonham, T. J. (1986). An Associative Memory for use in Image Recognition and Occlusion Analysis. *Vision and Image Computing.*

[44] Austin, J. (1998). RAM based neural networks, a short history. In RAM-Based Neural Networks, (1st. ed., pp. 3-17). *World Scientific Publishing Co. Pte. Ltd.*

[45] Yáñez-Márquez, C. & Díaz de León, J. L. (2001). Modelo ADAM. *IT 54, Serie Verde*, CIC-IPN. México.

[46] Myers, C. & Aleksander, I. (1988). Learning algorithms for probabilistic logic nodes. In *abstracts of the annual International Neural Networks Society Meeting* (INNS88), pp. 205. Boston.

[47] Myers, C. & Aleksander, I. (1989). Output functions for probabilistic logic nodes. In *proceedings of the IEEE International Conference on Neural Networks*, pp. 310-314. London.

[48] Taylor J.G. (1972). Spontaneous behaviour in neural networks. *Journal of Theoretical Biology, 36*: pp. 513-528.

[49] Gorse, G. & Taylor, J.G. (1988). On the equivalence properties of noisy neural and probabilistic RAM nets. *Physics Letters A*, 131(6): pp. 326-332.

[50] Gorse, G. & Taylor, J.G. (1991). A continuous input RAM-based stochastic neural model. *Neural Networks*, *4*: pp. 657-665.

[51] Gurney, K. N. (1992). Training nets of hardware realizable sigma-pi units. *Neural Networks*, *5*: pp. 289-303.

[52] Hassoun, M. H. 1993, Associative Neural Memories, *Oxford University Press*, New York.

[53] Kohonen, T. 1989, Self-Organization and Associative Memory, *Springer-Verlag*, Berlin.

[54] Acevedo-Mosqueda, M.E. (2006), Memorias Asociativas Bidireccionales Alfa-Beta, *Tesis doctoral*, Centro de Investigación en Computación, ciudad de México.

[55] Simpson, P. K. (1990), Artificial Neural Systems, *Pergamon Press* , New York.

[56] Steinbuch, K. & Frank, H. (1961), "Nichtdigitale Lernmatrizen als Perzeptoren", *Kybernetik*, vol. 1, no.3, pp. 117-124.

[57] 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.

[58] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001). Correlograph de Willshaw, Buneman & Longuet-Higgins, *IT-49, Serie Verde*, CIC-IPN, México.

[59] Anderson, J. A. & Rosenfeld, E. (1990). *Neurocomputing: Fundations of Research*, MIT Press ,Cambridge.

[60] 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.

[61] Austin, J. (1987), "ADAM: A Distributed Associative Memory for Scene Analysis", In *Proceedings of First International Conference on Neural Networks*, pp. 285-295.

[62] 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.

[63] 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.

[64] Yáñez Márquez, C. & Díaz-de-León Santiago, J.L. (2003). Memorias Asociativas Basadas en Relaciones de Orden y Operaciones Binarias, *Computación y Sistemas*, *Vol. 6, No. 4*, México, pp. 300-311. ISSN 1405-5546.

[65] Román-Godínez, I. & Yáñez-Márquez, C. (2007). *Complete Recall on Alpha-Beta Heteroassociative Memory*, Lecture Notes in Computer Science (ISI Proceedings), LNCS 4827, Springer-Verlag Berlin Heidelberg, pp. 193-202. ISBN: 978-3-540-76630-8.

[66] Román-Godínez, I., López-Yáñez, I., & Yáñez-Márquez, C. (2007). *Perfect Recall on the Lernmatrix*, Lecture Notes in Computer Science (ISI Proceedings), LNCS 4492, Springer-Verlag Berlin Heidelberg, pp. 835-841. ISBN: 978-3-540-72392-9.

[67] Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[68] Tran, Q., Toh, K., Srinivasan, D., Wong, K. & Low, S.Q. (2005). An empirical comparison of nine pattern classifiers, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 35, no. 5, pp. 1079-1091.

[69] Abdullah, M.R.B., Toh, K.-. & Srinivasan, D. (2006). A framework for empirical classifiers comparison, 2006 1st IEEE Conference on Industrial Electronics and Applications, art. no. 4026002.

[70] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research, vol. 7, pp. 1-30.

[71] Wolpert, D.H., Macready, W.G. (1997). No free lunch theorems for search. IEEE Transactions on Evolutionary Computation, 1 (1), pp. 83-98.

[72] Wolpert, D.H., et al. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1 (1), pp. 67-82.