INSTITUTO POLITECNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

IMPLEMENTACION DE LOS MODELOS ALFA-BETA CON LOGICA RECONFIGURABLE

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS EN INGENIERIA DE COMPUTO CON ESPECIALIDAD EN SISTEMAS DIGITALES

PRESENTA:

MARIO ALDAPE PEREZ

DIRECTOR DE TESIS:

DR. CORNELIO YÁÑEZ MÁRQUEZ

CO-DIRECTOR DE TESIS:

M en C. AMADEO J. ARGÜELLES CRUZ

MÉXICO, D.F. MAYO DE 2007

Índice General

Resumen

El presente trabajo de tesis se divide en dos partes; en la primera de ellas se desarrolla un algoritmo para la selección de rasgos (*Feature Selection*), que permite reducir dimensionalmente los patrones de entrada del conjunto fundamental.

En la segunda parte se propone una arquitectura de cómputo optimizada, para llevar a cabo tareas de aprendizaje y recuperación de patrones; logrando así, la implementación del modelo de Memorias Asociativas Alfa-Beta en dispositivos basados en lógica reconfigurable, usando lenguajes de descripción de hardware y esquemas de unidades funcionales operando en paralelo.

Se presenta un estudio experimental de la eficacia del algoritmo propuesto para la reducción dimensional de los datos, haciendo uso de la Herramienta HCM ver1.0. Además, se muestran los resultados del desempeño de la Arquitectura de Cómputo propuesta, haciendo uso de la herramienta Model*Sim* XE II/Starter 5.8c.

Con el desarrollo de este trabajo de tesis se muestra la robustez de las memorias asociativas Alfa-Beta. Asimismo, se hace notar la importancia que hasta ahora han adquirido estas memorias, en las áreas de aprendizaje y recuperación de patrones debido a su capacidad para representar el conocimiento de manera compacta. Adicionalmente, se muestran las ventajas derivadas de la aplicación de esquemas de unidades funcionales operando en paralelo.

Abstract

This thesis is divided into two main sections; in the first one, a new algorithm that conducts a data dimensionality reduction process (*Feature Selection*) is presented, this approach obtains a mask value that identifies and eliminates irrelevant information for classification purposes.

In the second one, an optimized computing architecture, which is capable of learning and recalling patterns, is proposed, in other words, the Alpha-Beta Associative Memories model is implemented on reconfigurable logic devices using hardware description languages (HDL) and parallel operational units schemes.

An experimental study that reveals the advantages and effectiveness of the proposed approach is developed using the HCM ver1.0 computational tool. Furthermore, the performance results of the optimized computing architecture are evaluated using Model*Sim* XE II/Starter 5.8c.

With this thesis, Alpha-Beta Associative Memories advantages are shown. In the same manner, it is to be noticed that the growing popularity of the associative memories is due to the ability of knowledge management in an efficient way. Finally, parallel operational units schemes advantages are shown.

CAPÍTULO 1

Introducción

En este trabajo de tesis se desarrolla el diseño de una arquitectura de cómputo que permite la implementación del modelo original de las memorias asociativas Alfa-Beta en dispositivos basados en lógica reconfigurable, usando lenguajes de descripción de hardware y técnicas de reducción dimensional de los datos.

1.1 Antecedentes

Aun cuando los primeros modelos de Memorias Asociativas surgieron hace algunas décadas, no es sino hasta los años setenta cuando se vuelven el foco de atención de importantes grupos de investigación. Los prolíficos trabajos científicos de los años ochenta, las posicionaron como entidades capaces de modelar no solo fenómenos biológicos asociativos, sino también como elementos fundamentales en los algoritmos concernientes a la teoría del reconocimiento de patrones y a sus aplicaciones.

El propósito fundamental de una memoria asociativa es recuperar correctamente patrones completos a partir de patrones de entrada, los cuales pueden estar alterados con ruido aditivo, sustractivo o combinado [1]. En toda memoria asociativa, previo a la fase de recuperación de patrones, se lleva a cabo la fase de aprendizaje; proceso mediante el cual se obtiene la memoria asociativa a través de la asociación de patrones, uno de entrada y uno de salida. Si para cada asociación se cumple que el patrón de entrada es igual al de salida, la memoria es autoasociativa; en caso contrario, la memoria es heteroasociativa: esto implica que las memorias autoasociativas son un caso particular de las memorias heteroasociativas [2].

A través del tiempo las memorias asociativas se han desarrollado paralelamente a las Redes Neuronales, desde la concepción del primer modelo de neurona artificial [3], hasta los modelos de redes neuronales basados en conceptos modernos como la morfología matemática [4], pasando por los importantes trabajos de los pioneros en las redes neuronales tipo *perceptron* [5, 6, 7] y el modelo de Hopfield, quien en 1982 presenta al mundo su red neuronal que también funciona como una memoria asociativa [8]. Con este importante trabajo de investigación, Hopfield propició el resurgimiento de las redes neuronales después del período posterior a la publicación del libro *Perceptrons* [9], en donde se demostró que el *perceptron* tenía severas limitaciones.

El primer modelo de memoria asociativa apareció 21 años antes que el modelo de Hopfield, y se debe a Karl Steinbuch, científico alemán quien en 1961 desarrolla una memoria heteroasociativa que funciona como un clasificador de patrones binarios: la *Lernmatrix* [10]. Ocho años después, los investigadores escoceses Willshaw, Buneman y Longuet-Higgins [11] presentan el *Correlograph*, dispositivo óptico elemental capaz de funcionar como una memoria asociativa. Dos modelos clásicos de memorias asociativas fueron presentados en 1972 por Anderson [12] y Kohonen [13] de manera independiente; debido a su importancia y a la similitud de los conceptos involucrados,

ambos modelos reciben el nombre genérico de *Linear Associator*. Una década después surgiría el ya mencionado modelo Hopfield de memoria asociativa.

No obstante su innegable relevancia, el modelo Hopfield tiene dos claras desventajas: primeramente, la memoria Hopfield es sólo autoasociativa, por lo que es incapaz de asociar patrones diferentes; y en segundo lugar, resulta evidente el hecho de que la capacidad de recuperación de patrones es muy pequeña, sólo de $0.15n$, siendo n la dimensión de los patrones almacenados.

Con objeto de subsanar la primera desventaja del modelo Hopfield, en 1988 Bart Kosko [14] crea un modelo de memoria heteroasociativa a partir de la memoria Hopfield: la memoria asociativa bidireccional BAM (Bidirectional Associative Memory) la cual, al igual que la memoria Hopfield, se basa en un algoritmo iterativo. A pesar de que el intento de Kosko fue exitoso al obtener una memoria heteroasociativa, la segunda desventaja de la memoria Hopfield no fue subsanada con la BAM: la memoria asociativa bidireccional de Kosko presenta muy bajas capacidades de aprendizaje y recuperación de patrones, al igual que los modelos subsecuentes de memorias asociativas bidireccionales [15-18].

El siguiente paso importante en la búsqueda de mejorar significativamente la capacidad de aprendizaje y recuperación de patrones de los modelos Hopfield y Kosko, fue la creación de las memorias asociativas morfológicas, a finales de la década de los noventa del siglo pasado. Para ello, Ritter y su equipo de investigación aplicaron los conceptos de la morfología matemática y de las redes neuronales morfológicas [4] para hacer surgir un nuevo tipo de memorias asociativas que lograron superar las capacidades de aprendizaje y recuperación de patrones de todos los modelos conocidos en ese momento [19].

Las memorias asociativas morfológicas sirvieron de inspiración para la creación de las memorias asociativas Alfa-Beta, desarrolladas en el año 2002 por investigadores del Centro de Investigación en Computación del IPN [20]. Cabe mencionar que este modelo no solo constituye el fundamento teórico de un nuevo tipo de memorias asociativas bidireccionales, las BAM Alfa-Beta [21-23], sino que además establece las bases para la implementación de las memorias asociativas Alfa-Beta en hardware.

En los años cincuenta, surgen los primeros trabajos de investigación concernientes al campo de la implementación de las redes neuronales en hardware. El primer aspecto que despertó el interés de la comunidad científica, fue la codificación de información mediante señales eléctricas controladas por voltajes y corrientes, consecuentemente, la interpretación de los datos usando números reales fue posible.

Marvin Minsky utilizó las propiedades resistivas de los materiales para simular pesos adaptativos por medio de potenciómetros [38]. Frank Rosenblatt representó los pesos mediante resistencias en redes eléctricas, logrando resolver tareas de reconocimiento de patrones [39]. En los años sesenta, Karl Steinbuch, construye la primera memoria asociativa, interconectando columnas de elementos resistivos que producen cambios en las corrientes de entrada [40].

En la década siguiente no aparecen implementaciones notables, excepto por el trabajo de Kunihiko Fukushima [41]. Durante los años ochenta, la mayoría de los esfuerzos se

centraron en la adaptación de los modelos conocidos hasta entonces sobre supercomputadoras de procesamiento paralelo.

No es sino hasta la década de los noventa, con el trabajo de Jacek Zurada [42], que se presenta una descripción mas extensa acerca de las implementaciones de redes neuronales en hardware, consecuentemente, los intentos por obtener arquitecturas de procesamiento capaces de emular el paralelismo existente en el mundo biológico, marcaron una clara tendencia hacia el aprovechamiento de la respuesta no lineal de los transistores de efecto de campo (FET) y el almacenamiento de información en dispositivos de cargas interconectadas (CCD) [43].

No obstante las notables ventajas que actualmente ofrecen los materiales semiconductores empleados en el diseño de circuitos integrados de alta escala de integración (VHSIC), hoy en día, la dimensionalidad de los datos concernientes al reconocimiento de patrones y a sus aplicaciones, supera los cientos o miles de entradas. Esto pone de manifiesto que, incluso para el diseño de novedosas arquitecturas de cómputo implementadas en dispositivos basados en lógica reconfigurable, debe llevarse a cabo algún tipo de preproceso en la información que conlleve no solo a la reducción dimensional de los datos, sino también a la optimización de los recursos de cómputo disponibles.

1.2 Justificación

El modelo de las Memorias Asociativas Alfa-Beta, desarrollado como trabajo de tesis doctoral [20], iguala y en ocasiones supera, tanto la capacidad de aprendizaje como de almacenamiento de las memorias asociativas morfológicas; sin embargo, por la naturaleza de las operaciones empleadas en la fase de aprendizaje, el proceso de asociación de patrones se realiza de forma serial, es decir, bit por bit. Lo anterior implica que, cuando el número de componentes presentes en un patrón de entrada es elevado (patrones altamente dimensionales), la fase de aprendizaje puede ser particularmente demandante.

Hasta septiembre de 2006, en la literatura científica se reportan algunos trabajos donde se ha modificado el modelo original de las memorias asociativas Alfa-Beta para lograr que acepten patrones con componentes enteras o reales e incluso imágenes en tonos de gris [24-27]. Sin embargo, no se han reportado trabajos relacionados con el diseño de arquitecturas especiales de cómputo que permitan la implementación del modelo original de las memorias asociativas Alfa-Beta en hardware.

La investigación científica desarrollada en este trabajo de tesis, responde a lo anterior, mediante el diseño y obtención de una arquitectura de cómputo, que no solo permite la implementación del modelo original de las memorias asociativas Alfa-Beta en hardware, sino que además, reduce sustancialmente el tiempo de ejecución de la fase de aprendizaje, al aplicar esquemas de unidades funcionales operando en paralelo.

1.3 Objetivo

Obtener una arquitectura de cómputo que permita la implementación del modelo original de las memorias asociativas Alfa-Beta en dispositivos basados en lógica reconfigurable, usando lenguajes de descripción de hardware (HDL) y técnicas de reducción dimensional de los datos (*Feature Selection*).

1.4 Contribuciones

Las contribuciones de este trabajo de tesis son:

- El desarrollo de un algoritmo que permite reducir dimensionalmente los patrones de entrada del conjunto fundamental.
- La obtención de una arquitectura de cómputo, capaz de llevar a cabo tareas de aprendizaje y recuperación de patrones de manera eficiente.
- El desarrollo de los bancos de prueba que aseguran el correcto funcionamiento tanto de la fase de aprendizaje como de recuperación, usando leguajes de descripción de hardware.
- El desarrollo de la herramienta computacional HCM v1.0, para facilitar el proceso de selección de rasgos.

1.5 Organización del documento

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

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

El capítulo 3 inicia con los fundamentos de las memorias asociativas Alfa-Beta, sigue con las definiciones y criterios aplicados para lograr la reducción dimensional de los datos, continúa con el análisis de los dos enfoques clásicos para la selección de rasgos y termina con una introducción al diseño de los sistemas digitales.

El capítulo 4 es la parte central de este trabajo. Es aquí, donde se introducen formalmente tanto el algoritmo desarrollado para alcanzar la reducción dimensional de los datos, como los elementos que hacen posible la obtención de una arquitectura de computo capaz de llevar a cabo tareas de aprendizaje y recuperación de información, de manera eficiente.

Los resultados experimentales, sobre 8 diferentes bases de datos, se incluyen en el capítulo 5; asimismo, se muestran los resultados del desempeño de las dos fases (aprendizaje y recuperación), involucradas en el desarrollo de la arquitectura de computo propuesta.

En el capítulo 6 se presentan tanto las conclusiones como las recomendaciones para trabajos futuros, continúa con dos apéndices (el manual de usuario de la herramienta computacional HCM v1.0 y la simbología empleada) y termina con las referencias bibliográficas.

CAPÍTULO 2

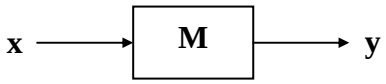
Estado del arte

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

2.1 Conceptos básicos

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

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



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

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

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

El conjunto fundamental se representa de la siguiente manera:

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

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

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

Una memoria es **Autoasociativa** si se cumple que $\mathbf{x}^\mu = \mathbf{y}^\mu \ \forall \mu \in \{1, 2, ..., p\}$, por lo que uno de los requisitos que se debe de cumplir es que $n = m$.

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

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

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

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

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

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

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

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

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

2. **Fase de Recuperación** (Operación de la memoria asociativa). Hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida $\mathbf{y}^\mu$, cuando se opera la memoria **M** con el patrón

fundamental de entrada $\mathbf{x}^{\mu}$; lo anterior para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo.

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

2.2 Estado del arte

A continuación, en esta sección haremos un breve recorrido por los modelos de memorias asociativas más representativos, los cuales sirvieron de base para la creación de modelos matemáticos que sustentan el diseño y operación de memorias asociativas más complejas. Para cada modelo se describe su fase de aprendizaje y su fase de recuperación.

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

2.2.1 *Lernmatrix* de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar* [20]. La importancia de la *Lernmatrix* [10, 29] se evidencia en una afirmación que hace Kohonen [13] en su artículo de 1972, donde apunta que las matrices de correlación, base fundamental de su innovador trabajo, vinieron a sustituir a la *Lernmatrix* de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^{\mu} \in A^n, \quad A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^{\mu} \in A^p$ que le corresponde (de entre p clases diferentes), codificada ésta con un método que en la literatura se le ha llamado *one-hot* [30].

La codificación *one-hot* funciona así: para representar la clase $k \in \{1, 2, ..., p\}$, se asignan a las componentes del vector de salida $\mathbf{y}^{\mu}$ los siguientes valores: $y_k^{\mu} = 1$, y $y_j^{\mu} = 0$ para $j = 1, 2, ..., k\text{-}1, k\text{+}1, ..., p$.

Algoritmo de la *Lernmatrix*

Fase de Aprendizaje

Se genera el esquema (*crossbar*) al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \in A^n \times A^p$. Cada uno de los componentes m_{ij} de **M**, la *Lernmatrix* de

Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

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

donde ε una constante positiva escogida previamente: es usual que ε es igual a 1.

Fase de Recuperación

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 $\vee$ es el operador *máximo*:

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

2.2.2 *Correlograph* de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa [11, 31]. En palabras de los autores "el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental".

Algoritmo del *Correlograph*

Fase de Aprendizaje

La *red asociativa* se genera al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^m$. Cada uno de los componentes m_{ij} de la *red asociativa* **M** tiene valor cero al inicio, y se actualiza de acuerdo con la regla:

$$m_{ij} = \begin{cases} 1 & \text{si } y_i^\mu = 1 = x_j^\mu \\ \textit{valor anterior} & \text{en otro caso} \end{cases}$$

Fase de Recuperación

Se le presenta a la *red asociativa* **M** un vector de entrada $\mathbf{x}^\omega \in A^n$. Se realiza el producto de la matriz **M** por el vector $\mathbf{x}^\omega$ y se ejecuta una operación de umbralizado, de acuerdo con la siguiente expresión:

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

donde *u* es el valor de umbral. Una estimación aproximada del valor de umbral *u* se puede lograr con la ayuda de un número indicador mencionado en el artículo [11] de Willshaw *et al.* de 1969: $log_2 n$.

2.2.3 *Linear Associator* de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen [12, 13, 36].

Para presentar el *Linear Associator* consideremos de nuevo el conjunto fundamental:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\} \text{ con } A = \{0, 1\}, \quad \mathbf{x}^\mu \in A^n \quad \text{y} \quad \mathbf{y}^\mu \in A^m$$

Algoritmo del *Linear Associator*

Fase de Aprendizaje

1) Para cada una de las *p* asociaciones $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se encuentra la matriz $\mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t$ de dimensiones *m* x *n*

2) Se suman la *p* matrices para obtener la memoria

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

de manera que la *ij*-ésima componente de la memoria M se expresa así:

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

Fase de Recuperación

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

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

2.2.4 La memoria asociativa Hopfield

El artículo de John J. Hopfield de 1982, publicado por la prestigiosa y respetada *National Academy of Sciences* (en sus *Proceedings*), impactó positivamente y trajo a la palestra internacional su famosa memoria asociativa [8].

En el modelo que originalmente propuso Hopfield, cada neurona x_i tiene dos posibles estados, a la manera de las neuronas de McCulloch-Pitts: $x_i = 0$ y $x_i = 1$; sin embargo, Hopfield observa que, para un nivel dado de exactitud en la recuperación de patrones, la capacidad de almacenamiento de información de la memoria se puede incrementar por un factor de 2, si se escogen como posibles estados de las neuronas los valores $x_i = -1$ y $x_i = 1$ en lugar de los valores originales $x_i = 0$ y $x_i = 1$.

Al utilizar el conjunto {-1, 1} y el valor de umbral cero, la fase de aprendizaje para la memoria Hopfield será similar, en cierta forma, a la fase de aprendizaje del *Linear Associator*. La intensidad de la fuerza de conexión de la neurona x_i a la neurona x_j se representa por el valor de m_{ij}, y se considera que hay simetría, es decir, $m_{ij} = m_{ji}$. Si x_i no está conectada con x_j entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona consigo misma, lo cual significa que $m_{ij} = 0$. El estado instantáneo del sistema está completamente especificado por el vector columna de dimensión n cuyas componentes son los valores de las n neuronas.

La memoria Hopfield es autoasociativa, simétrica, con ceros en la diagonal principal. En virtud de que la memoria es autoasociativa, el conjunto fundamental para la memoria Hopfield es $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, ..., p\}$ con $\mathbf{x}^\mu \in A^n$ y $A = \{-1, 1\}$

Algoritmo Hopfield

Fase de Aprendizaje

La fase de aprendizaje para la memoria Hopfield es similar a la fase de aprendizaje del *Linear Associator*, con una ligera diferencia relacionada con la diagonal principal en ceros, como se muestra en la siguiente regla para obtener la ij-ésima componente de la memoria Hopfield **M**:

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

Fase de Recuperación

Si se le presenta un patrón de entrada $\tilde{\mathbf{x}}$ a la memoria Hopfield, ésta cambiará su estado con el tiempo, de modo que cada neurona x_i ajuste su valor de acuerdo con el resultado que arroje la comparación de la cantidad $\sum_{j=1}^{n} m_{ij} x_j$ con un valor de umbral, el cual normalmente se coloca en cero.

Se representa el estado de la memoria Hopfield en el tiempo t por $\mathbf{x}(t)$; entonces $x_i(t)$ representa el valor de la neurona x_i en el tiempo t y $x_i(t+1)$ el valor de x_i en el tiempo siguiente $(t+1)$.

Dado un vector columna de entrada $\tilde{\mathbf{x}}$, la fase de recuperación consta de tres pasos:

1) Para $t = 0$, se hace $\mathbf{x}(t) = \tilde{\mathbf{x}}$; es decir, $x_i(0) = \tilde{x}_i$, $\forall i \in \{1,2,3,...,n\}$

2) $\forall i \in \{1,2,3,...,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) > 0 \\ x_i(t) \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) = 0 \\ -1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) < 0 \end{cases}$$

3) Se compara $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1, 2, 3,...,n\}$. Si $\mathbf{x}(t+1) = \mathbf{x}(t)$ el proceso termina y el vector recuperado es $\mathbf{x}(0) = \tilde{\mathbf{x}}$. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar a un valor $t = \tau$ para el cual $x_i(\tau+1) = x_i(\tau)$ $\forall i \in \{1, 2, 3,...,n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

En el artículo original de 1982, Hopfield había estimado empíricamente que su memoria tenía una capacidad de recuperar $0.15n$ patrones, y en el trabajo de Abu-Mostafa & St. Jacques [32] se estableció formalmente que una cota superior para el número de vectores de estado arbitrarios estables en una memoria Hopfield es n.

2.2.5 Memoria Asociativa Bidireccional (BAM) de Kosko.

Bart Kosko, investigador de la *University of Southern California*, propuso en 1988 la *Bidireccional Associative Memory* (BAM) [14] para subsanar la clara desventaja de la autoasociatividad de la memoria Hopfield. La BAM maneja pares de vectores (A_1,B_1), ... (A_m,B_m), donde $A \in \{0, 1\}^n$ y $B \in \{0, 1\}^p$.

Del mismo modo que Austin ensambló dos redes asociativas de Willshaw para diseñar su ADAM [33], Kosko ideó un arreglo de dos memorias Hopfield, y demostró que este diseño es capaz de asociar patrones de manera heteroasociativa.

La matriz $\mathbf{M}$ es una memoria Hopfield con la única diferencia que la diagonal principal es diferente de cero. $\mathbf{M}^T$ es la matriz transpuesta de $\mathbf{M}$ que, ahora como entrada, recibe a B y la salida será A. El proceso bidireccional anteriormente ilustrado continúa hasta que A y B convergen a una pareja estable (A_i, B_i).

$$\begin{aligned} A &\rightarrow \mathbf{M} \rightarrow B \\ A' &\leftarrow \mathbf{M}^T \leftarrow B \\ A' &\rightarrow \mathbf{M} \rightarrow B' \\ A'' &\leftarrow \mathbf{M}^T \leftarrow B' \\ &\cdots \\ A_i &\rightarrow \mathbf{M} \rightarrow B_i \\ A_i &\leftarrow \mathbf{M}^T \leftarrow B_i \end{aligned}$$

Kosko descubrió que su memoria funcionaba mejor con patrones bipolares que con patrones binarios (a la manera de Hopfield), de ahí que: $A \in \{-1, 1\}^n$ y $B \in \{-1, 1\}^p$.

Para la codificación de la BAM se consideran las *m* asociaciones; sumándolas para formar la matriz de correlación, de acuerdo a la siguiente expresión:

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

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

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

En el proceso de decodificación, cada neurona a_i que se encuentra en el campo *A* y cada neurona b_i localizada en el campo *B*, de forma independiente y asíncrona, examina la suma de entrada de las neuronas del otro campo, entonces puede o no cambiar su estado si la suma de entrada es mayor, igual o menor que un umbral dado. Si la suma de entrada es igual al umbral, entonces la neurona no cambia su estado. La suma de entrada para b_j es el producto interno columna:

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

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

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

donde $\mathbf{M}_i$ es la *i*-ésima fila de **M**. Se toma el 0 como el umbral para todas las neuronas. Las funciones de umbral para a_i y b_j son:

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

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

Cuando se le presenta un patrón (*A*, *B*) a la BAM, las neuronas en los campos *A* y *B* se prenden o se apagan de acuerdo a la ocurrencia de 1's y 0's en los vectores de estado *A* y *B*. Las neuronas continúan sus cambios de estado hasta que se alcance un estado estable bidireccional (A_f, B_f).

2.2.6 Memorias Asociativas Morfológicas

La diferencia fundamental entre las memorias asociativas clásicas (*Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Asociativa Hopfield) y las memorias asociativas morfológicas radica en los fundamentos operacionales de éstas últimas, que son las operaciones morfológicas de *dilatación* y *erosión*; el nombre de las memorias asociativas morfológicas está inspirado precisamente en estas dos operaciones básicas.

Estas memorias rompieron con el esquema utilizado a través de los años en los modelos de memorias asociativas clásicas, que utilizan operaciones convencionales entre vectores y matrices para la fase de aprendizaje y suma de productos para la recuperación de patrones. Las memorias asociativas morfológicas cambian los productos por sumas y las sumas por máximos o mínimos en ambas fases, tanto de aprendizaje como de recuperación de patrones [19, 34, 35].

Hay dos tipos de memorias asociativas morfológicas: las memorias *max*, simbolizadas con **M**, y las memorias *min*, cuyo símbolo es **W**; en cada uno de los dos tipos, las memorias pueden funcionar en ambos modos: heteroasociativo y autoasociativo.

Se definen dos nuevos productos matriciales:

El *producto máximo* entre **D** y **H**, denotado por $\mathbf{C} = \mathbf{D} \nabla \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya *ij*-ésima componente c_{ij} es

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

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

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

Los productos máximo y mínimo contienen a los operadores máximo y mínimo, los cuales están íntimamente ligados con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

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 $\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^{t}$ de dimensiones *m* x *n*, donde el negado transpuesto del patrón de entrada $\mathbf{x}^{\mu}$ se define como $(-\mathbf{x}^{\mu})^{t} = (-x_1^{\mu}, -x_2^{\mu}, \ldots, -x_n^{\mu})$.
2. Se aplica el operador máximo $\vee$ a las *p* matrices para obtener la memoria **M**.

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

Fase de Recuperación

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

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

Las fases de aprendizaje y de recuperación de las **memorias morfológicas *min*** se obtienen por dualidad.

Memorias Autoasociativas Morfológicas

Para este tipo de memorias se utilizan los mismos algoritmos descritos anteriormente y que son aplicados a las memorias heteroasociativas; lo único que cambia es el conjunto fundamental. Para este caso, se considera el siguiente conjunto fundamental:

$$\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mathbf{x}^{\mu} \in A^{n}, \text{ donde } \mu = 1, 2, ..., p\}$$

2.2.7 Memorias Asociativas Alfa-Beta

Las memorias asociativas Alfa-Beta [20] utilizan máximos y mínimos, y dos operaciones binarias originales 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:

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

La operación binaria α: A x $A \rightarrow B$ se define como:

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

La operación binaria β: B x $A \rightarrow A$ se define como:

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

El fundamento teórico de las memorias asociativas Alfa-Beta se presenta en el siguiente capítulo de forma más completa, debido a que éstas, constituyen la base fundamental para este trabajo de tesis.

2.2.8 Memorias Asociativas Mediana

Las Memorias Asociativas Mediana [27] utilizan los operadores A y B, definidos de la siguiente forma:

$$A(x, y) = x - y$$
$$B(x, y) = x + y$$

Las operaciones utilizadas se describen a continuación.

Sean $P = [p_{ij}]_{mxr}$ y $Q=[q_{ij}]_{rxn}$ dos matrices.

Operación $\lozenge_A$: $P_{mxr} \lozenge_A Q_{rxn} = [f_{ij}^A]_{mxn}$ donde $f_{ij}^A = \underset{k=1}{\overset{r}{\mathbf{med}}} A(p_{ik}, q_{k,j})$

Operación $\lozenge_B$: $P_{mxr} \lozenge_B Q_{rxn} = [f_{ij}^B]_{mxn}$ donde $f_{ij}^B = \underset{k=1}{\overset{r}{\mathbf{med}}} B(p_{ik}, q_{k,j})$

Algoritmo de las Memorias Mediana

Fase de Aprendizaje

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

$$[\mathbf{y}^\xi \lozenge_A (\mathbf{x}^\xi)^t]_{mxn}$$

Paso 2. Se aplica el operador media a las matrices obtenidas en le paso 1 para obtener la matriz **M**, como sigue:

$$\mathbf{M} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \left[y^\xi \lozenge_A \left(x^\xi \right)^t \right]$$

El *ij*-ésimo componente **M** está dado como sigue:

$$m_{ij} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} A\left(y_i^\xi, x_j^\xi \right)$$

Fase de Recuperación

Se tienen dos casos:

Caso 1. Recuperación de un patrón fundamental. Un patrón $\mathbf{x}^w$, con $w \in \{1, 2, ..., p\}$ se le presenta a la memoria **M** y se realiza la siguiente operación:

$$\mathbf{M} \lozenge_B \mathbf{x}^w$$

El resultado es un vector columna de dimensión *n*, con la *i*-ésima componente dada como:

$$\left(\mathbf{M} \lozenge_B x^w \right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} B\left(m_{ij}, x_j^w \right)$$

Caso 2. Recuperación de un patrón alterado. Un patrón $\tilde{\mathbf{x}}$, que es una versión alterada de un patrón $\mathbf{x}^w$, se le presenta a la memoria $\mathbf{M}$ y se realiza la siguiente operación:

$$\mathbf{M} \lozenge_{\mathrm{B}} \tilde{\mathbf{x}}$$

De nuevo, el resultado es un vector de dimensión *n*, con la *i*-ésima componente dada como:

$$\left(\mathbf{M} \lozenge_B \tilde{\mathbf{x}}\right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left(m_{ij}, \tilde{x}_j\right)$$

CAPÍTULO 3

Materiales y Métodos

Este capítulo contiene tres secciones. En la sección 3.1 se describe el modelo matemático de las Memorias Asociativas Alfa-Beta; el cual constituye la base fundamental de la arquitectura de cómputo propuesta en este trabajo de tesis. La sección 3.2, por otro lado, versa sobre algunos conceptos relacionados con la reducción dimensional de los datos. Finalmente, se cierra este capítulo con una introducción al diseño de sistemas digitales en la sección 3.3.

3.1 Memorias Asociativas Alfa-Beta

En esta sección se presenta el fundamento teórico que sustenta a las memorias asociativas Alfa-Beta tal como aparece en [20]; para ello, se presentan las definiciones y propiedades de las operaciones α y β, las operaciones matriciales que se derivan de estas operaciones originales, y se describen las fases de aprendizaje y recuperación de la memorias heteroasociativas y autoasociativas Alfa-Beta, tanto $\vee$ (*max*) como $\wedge$ (*min*).

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

3.1.1 Operaciones binarias α y β: definiciones y propiedades

Las memorias Alfa-Beta 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:

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

La operación binaria $\alpha: A \times A \rightarrow B$ se define como se muestra en la Tabla 3.1.1.1.

Tabla 3.1.1.1. Operación binaria $\alpha: A \times A \rightarrow B$

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

La operación binaria $\beta: B \times A \rightarrow A$ se define como se muestra en la Tabla 3.1.1.2.

Tabla 3.1.1.2. Operación binaria $\beta: B \times A \rightarrow A$

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

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

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

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

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

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

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

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

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

Demostración.

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

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

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

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

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

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

Dado el resultado del lema anterior, es conveniente escoger un símbolo único, digamos el símbolo ⊗, que represente a las dos operaciones ∇_{α} y Δ_{α} cuando se opera un vector columna de dimensión *m* con un vector fila de dimensión *n*:

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

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

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

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

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

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

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

Demostración.

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

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

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

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

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

Demostración.

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

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

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

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

3.1.2 Memorias Heteroasociativas Alfa-Beta

Se tienen dos tipos de memorias heteroasociativas Alfa-Beta: tipo $\mathbf{V}$ y tipo $\mathbf{\Lambda}$. En la generación de ambos tipos de memorias se usará el operador $\otimes$ el cual tiene la siguiente forma:

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

Algoritmo Memorias Alfa-Beta tipo V

Fase de Aprendizaje

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

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

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

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

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

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

Fase de Recuperación

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

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

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

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

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

Algoritmo Memorias Alfa-Beta tipo Λ

Fase de Aprendizaje

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

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

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

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

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

$$\lambda = \bigwedge_{\mu=1}^{p} \alpha(y_i^\mu, x_j^\mu)$$

Fase de Recuperación

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

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

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

Teorema 4.20. (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, \ldots, p\}$ *el conjunto fundamental de una memoria heteroasociativa* $\alpha\beta$ *representada por* $\mathbf{\Lambda}$. *Si* ω *es un valor arbitrario fijo tal que* $\omega \in \{1, 2, \ldots, p\}$, *y si además para cada* $i \in \{1, \ldots, m\}$ *se cumple que* $\exists j = j_0 \in \{1, \ldots, n\}$, *el cual depende de* ω *y de* i, *tal que* $\lambda_{ij_0} = \alpha(y_i^\omega, x_{j_0}^\omega)$, *entonces la recuperación* $\mathbf{\Lambda} \nabla_\beta \mathbf{x}^\omega$ *es correcta; es decir* $\mathbf{\Lambda} \nabla_\beta \mathbf{x}^\omega = \mathbf{y}^\omega$.

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

3.1.3 Memorias Autoasociativas Alfa-Beta

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

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

1. El conjunto fundamental toma la forma $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, ..., p\}$

2. Los patrones fundamentales de entrada y salida son de la misma dimensión; denotada por *n*.
3. La memoria es una matriz cuadrada, para ambos tipos, **V** y **Λ**. Si $\mathbf{x}^{\mu} \in A^n$ entonces

$$\mathbf{V} = \lfloor v_{ij} \rfloor_{nxn} \text{ y } \Lambda = \lfloor \lambda_{ij} \rfloor_{nxn}$$

Algoritmo Memorias Autoasociativas Alfa-Beta tipo V

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

Fase de Aprendizaje

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

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

Paso 2. Se aplica el operador binario máximo $\mathbf{V}$ a las matrices obtenidas en el paso 1:

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

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

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

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

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

CASO 1: Patrón fundamental. Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa Alfa-Beta tipo V y se realiza la operación Δ_{β}:

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

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

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

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

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

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

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

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

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

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

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

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

De la tabla de propiedades en [20] se tiene que $\alpha(x_i^\mu, x_i^\mu) = 1$, por lo que la expresión anterior se transforma en:

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

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

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

$$v_{ii} = 1 = \alpha(x_i^{\omega}, x_i^{\omega})$$

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

$$v_{ij_0} = \alpha(x_i^{\omega}, x_{j_0}^{\omega})$$

Por lo tanto, de acuerdo con el Teorema 4.7

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

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

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

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

Teorema 4.30. (*Numeración tal como aparece en* [20]). *Sea* $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta representada por* **V**, *y sea* $\tilde{\mathbf{x}} \in A^n$ *un patrón alterado con ruido aditivo respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* **V** *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* ω *y de i tal* que $v_{ij_0} \le \alpha(x^{\omega}, \tilde{x}_{j_0})$, *entonces la recuperación* $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}}$ *es correcta, es decir,* $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

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

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

Algoritmo Memorias Autoasociativas Alfa-Beta tipo Λ

Fase de Aprendizaje

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

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

Paso 2. Se aplica el operador binario máximo $\mathbf{\Lambda}$ a las matrices obtenidas en el paso 1:

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

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

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

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

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

CASO 1: Patrón fundamental. Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa $\alpha\beta$ tipo V y se realiza la operación ∇_{β}:

$$\mathbf{\Lambda}\, \Delta_{\beta}\, \mathbf{x}^{\omega}$$

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

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

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

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

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

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

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

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

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

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

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

De la tabla de propiedades en [20] tiene que $\alpha(x_i^{\mu}, x_i^{\mu}) = 1$, por lo que la expresión anterior se transforma en:

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

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

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

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

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

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

Por lo tanto, de acuerdo con el Teorema 4.20

$$\boldsymbol{\Lambda\nabla}_{\beta}\, \mathbf{x}^{\omega} = \mathbf{x}^{\omega}, \forall \omega \in \{1, 2, ..., p\}$$

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

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre *p* que es la cardinalidad del conjunto fundamental; y esto quiere decir que

el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo Λ, con recuperación correcta, es máximo.

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

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

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

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

3.2 Selección de Rasgos

La selección de rasgos se ha vuelto el foco de atención de muchos trabajos de investigación, principalmente en áreas del conocimiento humano donde los conjuntos de datos (entrenamiento/prueba) consisten en cientos o miles de rasgos. Algunos ejemplos clásicos de su aplicación son: la selección de características en microarreglos genéticos, la predicción de ocurrencia de enfermedades en el ser humano y la detección de estructuras de electrones libres en la ionosfera entre muchos otros.

Diversos son los aspectos que deben ser observados para llevar a cabo procesos de aprendizaje automático (*Machine Learning*); sin embargo, la discriminación de información redundante y la preservación de información relevante son piezas clave para lograr la reducción dimensional de los datos. La mayoría de los autores coinciden en que la selección de rasgos (*Feature Selection*) puede ser dividida en dos grandes tareas: decidir que rasgos son los que mejor describen el contexto y seleccionar cual es la mejor combinación de éstos que mejore la precisión predictiva [48]; sin embargo, las discrepancias ocurren por los criterios usados para definir tanto la relevancia como la redundancia en los datos.

3.2.1 Algunas definiciones

En 1991 varios autores coincidieron en que la relevancia de la información debe ser considerada booleana, libre de ruido y definida en términos de distribuciones de probabilidad, que permitan obtener una estimación confiable que sugiera la eliminación de rasgos claramente identificables [49].

Definición 1
Un rasgo X_i se dice que es **relevante** en un concepto *C*, si X_i aparece en toda la representación del concepto *C* y varía sistemáticamente para cada categoría o clase.

Definición 2
Un rasgo X_i se dice que es **relevante** si existe un x_i y *y* para el cual $p(X_i = x_i) > 0$, tal que

$$p(Y = y \mid X_i = x_i) \neq p(Y = y).$$

Bajo esta definición, X_i es relevante, si al conocer su valor, produce un cambio en la asignación de la etiqueta de clase. En otras palabras, si *Y* es condicionalmente dependiente de X_i; sin embargo, cuando se tienen datos sin etiquetar, esta definición es incapaz de identificar información relevante, ya que, la asignación de la etiqueta de clase bajo estas condiciones es equiprobable para toda instancia considerada en la fase de aprendizaje. Cabe mencionar que esta situación es meramente hipotética, ya que, en procesos relacionados con el aprendizaje automático no se utilizan instancias sin etiquetar para la fase de entrenamiento.

Definición 3
Sea, $S_i = \{X_1,\ldots,X_{i-1},\ldots,X_{i+1},\ldots,X_m\}$ el conjunto de todos los rasgos sin contar a X_i.
Sea, s_i el valor asignado para todos los rasgos en S_i.
Un rasgo X_i se dice que es **relevante**, si existe un x_i, *y* y s_i para el cual $p(X_i = x_i) > 0$, tal que

$$p(Y = y, S_i = s_i \mid X_i = x_i) \neq p(Y = y, S_i = s_i).$$

Bajo esta definición, X_i es relevante, cuando la información derivada del conocimiento de X_i puede cambiar la asignación de la etiqueta de clase.

Definición 4
Un rasgo X_i se dice que es **relevante**, si existe un x_i, *y* y s_i para el cual $p(X_i = x_i, S_i = s_i) > 0$, tal que

$$p(Y = y \mid X_i = x_i, S_i = s_i) \neq p(Y = y \mid S_i = s_i).$$

Bajo esta definición, X_i es relevante, si la probabilidad de que sea asignada una determinada etiqueta de clase puede cambiar cuando es eliminada la información que proporciona el valor del rasgo X_i.

Sería deseable que, con las definiciones antes mencionadas, se pudiera identificar información relevante en un conjunto de instancias determinadas; sin embargo, en situaciones donde existe alta correlación en los datos que describen un problema, las definiciones anteriores no son suficientes, consecuentemente, surge la necesidad de definir dos grados de relevancia: fuerte y débil.

3.2.1.1 Relevancia fuerte y débil

Existen dos grados de relevancia en la información: fuerte y débil. Para poder diferenciar ambos grados, es necesario considerar que, la relevancia debe ser definida en términos de un clasificador Bayesiano, basado en el principio matemático de que la mayoría de los sucesos son dependientes y que la probabilidad de que un suceso ocurra en el futuro puede ser deducida de las ocurrencias anteriores de dicho evento [50].

Un rasgo X_i es **fuertemente relevante** si al eliminar la información que aporta éste, se observa un decremento en la precisión predictiva del modelo en cuestión, es decir, la simple remoción de la información que proporciona este rasgo, deteriora el desempeño del proceso de clasificación. En términos de funciones de probabilidad condicional para la asignación de la etiqueta de clase, se puede enunciar que un rasgo X_i es **fuertemente relevante** si cumple con la siguiente expresión.

$$p(Y = y \mid X_i = x_i, S_i = s_i) \neq p(Y = y \mid S_i = s_i).$$

Un rasgo X_i es **débilmente relevante**, si no es fuertemente relevante y existe un subconjunto de rasgos, *S*, para el cual el desempeño del modelo en cuestión es inferior que el alcanzado por el subconjunto de rasgos dado por $S \cup \{X\}$; en otras palabras, la precisión predictiva alcanzada por el subconjunto de rasgos *S*, es menor que la precisión predictiva alcanzada cuando se considera la información contenida en el rasgo X_i. En términos de funciones de probabilidad condicional para la asignación de la etiqueta de clase, se puede enunciar que un rasgo X_i es **débilmente relevante** si existe un subconjunto de características $S'_i \subset S_i$ para el cual existe alguna x_i, *y* y s'_i con $p(X_i = x_i, S'_i = s'_i) > 0$ y cumple con la siguiente expresión.

$$p(Y = y \mid X_i = x_i, S'_i = s'_i) \neq p(Y = y \mid S'_i = s'_i).$$

Dadas ambas definiciones para los dos grados de relevancia posibles, se dice que un rasgo X_i es relevante, ya sea fuerte o débilmente relevante; de lo contrario se dice que X_i es irrelevante.

3.2.1.2 Optimalidad del subconjunto de rasgos

En la búsqueda de subconjuntos de rasgos que permitan representar la información de manera reducida, el algoritmo de inducción, típicamente está dado por un clasificador que usa todos los rasgos fuertemente relevantes y posiblemente algunos de relevancia débil, no obstante que, la selección de rasgos propuesta, generalmente es un subconjunto de carácter subóptimo, ocasionado por el desconocimiento de la distribución de probabilidad sobre los datos nunca antes vistos (conjunto de prueba).

Otro aspecto delicado es que, con las definiciones anteriormente propuestas, no se puede asegurar que un rasgo fuertemente relevante aparezca en el subconjunto de rasgos seleccionado, así como tampoco se puede asegurar que algunos rasgos débilmente relevantes no aparezcan en esta selección.

Claramente estas limitaciones apuntan a la existencia de algún tipo de procedimiento encargado de esta tarea nada trivial. Desde una perspectiva subyacente a los requerimientos de cómputo necesarios para conducir esta búsqueda, sería deseable que, el algoritmo de inducción se comportara adecuadamente aún en presencia de información irrelevante, no obstante que, la complejidad computacional, generalmente aumenta de manera exponencial en presencia de instancias con estas características [45]. Algunos beneficios derivados de la selección de rasgos son:

- Facilitar la visualización y comprensión de los datos
- Reducción en los requerimientos de almacenamiento
- Reducción en los tiempos de entrenamiento y recuperación
- Incrementos en la precisión predictiva

Dependiendo del tipo de algoritmo de inducción o del criterio considerado para la evaluación de los rasgos que incrementan la precisión predictiva sobre instancias no conocidas, algunos autores definen dos grandes enfoques: filtrado (*filter*) y envoltura (*wrapper*).

3.2.2 Método de Filtrado (*Filter*)

Para la selección de características usando el método de filtrado de rasgos, generalmente aparece una etapa de preprocesamiento previa a la estimación del desempeño alcanzado por el subconjunto de rasgos propuesto. En esta etapa, típicamente se recurre a métricas o criterios estadísticos (media, varianza, correlación y desviación estándar) aplicados directamente sobre los datos originales que definen el problema; esto deja claro que, bajo este enfoque, no es necesario considerar el algoritmo de inducción como parámetro de referencia para lograr la reducción dimensional de los datos. (Figura 3.2.i).

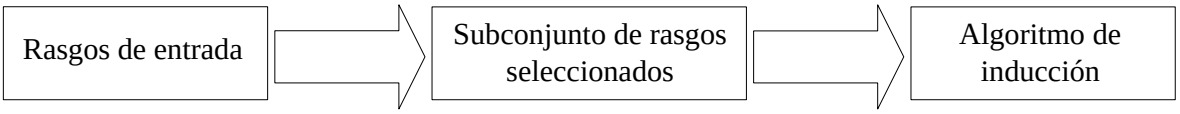


Figura 3.2.i *En el enfoque de filtrado, los rasgos son filtrados independiente del algoritmo de inducción.*

En otras palabras, a partir del conjunto original de rasgos que definen el problema, se realiza una exploración en los datos buscando algún tipo de regularidad en las métricas propuestas, que permitan vislumbrar mediante criterios estadísticos cuáles son los rasgos que incrementan la separación entre clases o cuáles de éstos describen mejor las fronteras entre categorías, de modo que, se tenga un intervalo de confianza mayor con aquellas instancias que presentan alta correlación; existen algoritmos como el FOCUS [52] que examinan todos los subconjuntos posibles en busca de aquel, con el menor número de rasgos suficientes para determinar la etiqueta correcta en todas las instancias presentes en la fase de entrenamiento. Este algoritmo realiza una búsqueda exhaustiva en todo el espacio de características en aras de obtener un subconjunto de rasgos tan reducido como sea posible; generalmente son encontrados aquellos rasgos que maximizan las diferencias entre instancias conocidas.

Contrario a lo que se esperaría de un subconjunto con tales características, la precisión predictiva alcanzada para datos no conocidos suele ser muy pobre (sobreadaptación al problema).

En la mayoría de los casos cuando el subconjunto de rasgos propuesto se comporta extremadamente bien en la fase de aprendizaje, se observa un decremento en la precisión predictiva sobre instancias no conocidas, es decir, es poco probable que se puedan inferir reglas claras para casos desconocidos a partir de la sobreadaptación al problema en cuestión.

Existe un caso práctico, que hace evidentes las debilidades de este algoritmo para selección de rasgos:

Suponga que se tiene un conjunto de rasgos que predicen la ocurrencia de cáncer en seres humanos y que uno de los rasgos es el número de paciente; dado que este enfoque opera bajo el criterio de "mínimo número de rasgos suficientes" para asignar correctamente la etiqueta de clase a cada una de las instancias presentes en la fase de entrenamiento, resulta que el rasgo seleccionado que mejor clasifica a cada uno de los patrones es el número de paciente, lo cual claramente no será de utilidad práctica para instancias nunca antes vistas.

Otro algoritmo frecuentemente referido bajo el enfoque de filtrado, es el Relief [53]. Este busca minimizar los efectos ocasionados por sobreadaptación al problema, mediante la asignación de valores de relevancia (pesos) asociados a la información contenida en el conjunto de instancias presentes en la fase de entrenamiento; en otras palabras, el procedimiento para encontrar un subconjunto de rasgos mínimos, capaz de incrementar la precisión predictiva sobre instancias desconocidas, se lleva a cabo mediante el ajuste sucesivo de los valores de los pesos sobre los diferentes rasgos. A diferencia del FOCUS, la función objetivo si está acotada por el algoritmo de inducción. Cabe mencionar que el algoritmo Relief, generalmente busca mejorar la precisión predictiva mediante el ajuste sucesivo de pesos, pero tiene fuertes limitaciones cuando existe información redundante, que provoca que el proceso de convergencia o criterio de paro sea difícil de asegurar.

3.2.3 Método de Envoltura (*Wrapper*)

El enfoque de Envoltura para la selección de rasgos (*wrapper*), fue ampliamente difundido en 1997 [50]. En un principio surge como una alternativa para solucionar el problema de sobreadaptación a los datos, sin embargo, la popularidad de este enfoque resulta del hecho de poder establecer un criterio de paro, basado en el número máximo de iteraciones requeridas para explorar la totalidad de subconjuntos posibles y encontrar aquel que maximice la precisión predictiva con el menor número de rasgos.

En otras palabras, este enfoque permite que dos aspectos críticos relacionados con la búsqueda del subconjunto óptimo de rasgos, sean cubiertos:

- ❖ Asegurar el criterio de paro del algoritmo
- ❖ Conocer el número máximo de iteraciones requeridas

Principalmente se usa el algoritmo de inducción como parte central de una envoltura que conduce la búsqueda del mejor subconjunto de rasgos, es decir, el algoritmo de inducción es utilizado como criterio de evaluación para obtener la mejor combinación de características (Figura 3.2.ii).

La idea central de este enfoque se basa en utilizar el algoritmo de inducción como "caja negra", es decir, se escoge un conjunto de instancias de entrenamiento y se propone un primer subconjunto de características. El desempeño de este subconjunto es evaluado usando el algoritmo de inducción como parámetro de la precisión alcanzada. De este modo se inicia un proceso iterativo alrededor de todas las combinaciones de rasgos posibles, buscando el subconjunto que cumpla mejor con la función objetivo definida en el algoritmo de inducción. En otras palabras, se hace una búsqueda exhaustiva sobre todos los posibles subconjuntos de rasgos hasta encontrar aquel que alcanza el mejor índice de desempeño. (Figura 3.2.iii).

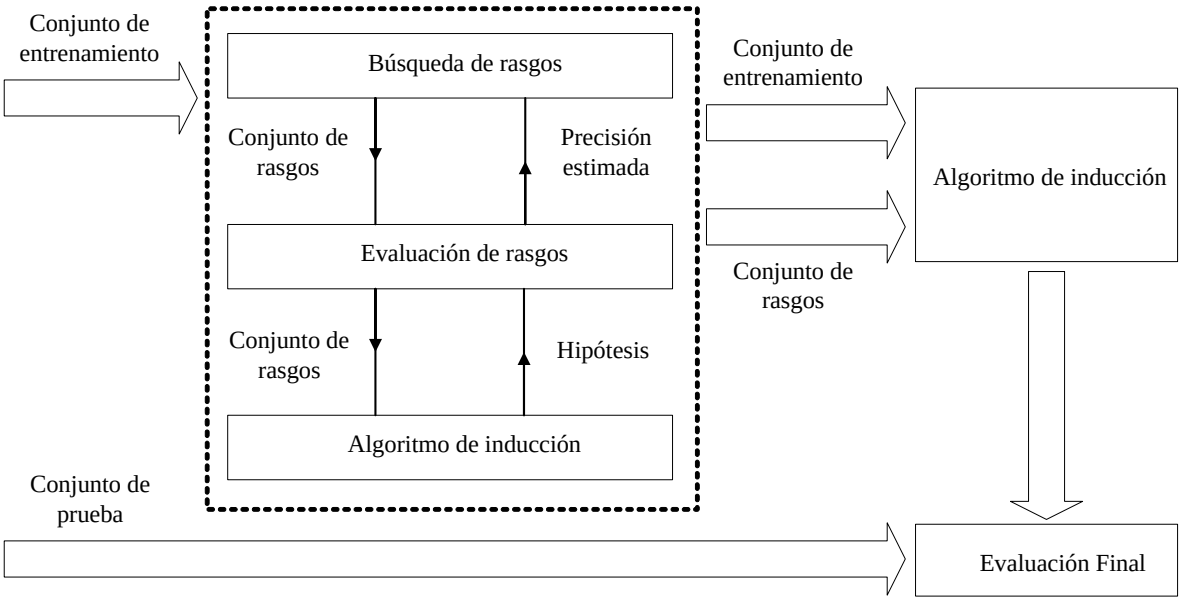


Figura 3.2.ii *El enfoque de Envoltura (wrapper) para la selección de rasgos.*

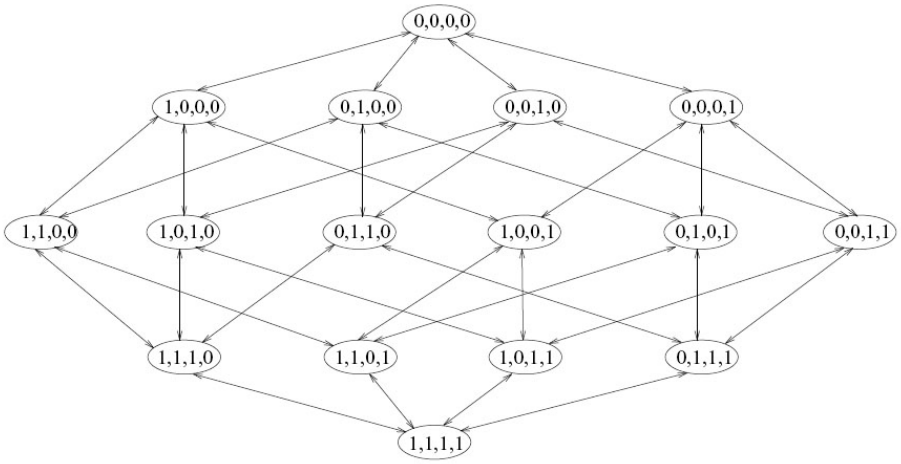


Figura 3.2.iii *Cada nodo está conectado a otro que tiene un rasgo agregado o eliminado.*

Como resultado de la fase de aprendizaje, se obtiene un subconjunto de rasgos que mejoran la precisión predictiva, cabe mencionar que los índices de desempeño alcanzados, únicamente son válidos para instancias conocidas.

Para obtener un parámetro confiable que represente el comportamiento del subconjunto de rasgos propuesto sobre instancias nunca antes vistas, es recomendable emplear métodos de validación cruzada.

Entre los más frecuentemente empleados tenemos:

- Hold out cross validation
- K-fold cross validation
- Leave-one-out cross validation

3.2.4 Métodos de Validación Cruzada

Los métodos de validación cruzada son comúnmente aplicados cuando se busca obtener una estimación confiable del comportamiento de algún modelo propuesto en presencia de instancias no conocidas, es decir, se pretende saber si el subconjunto de rasgos seleccionado es el adecuado y si la precisión predictiva sobre instancias desconocidas es aceptable.

3.2.4.1 Hold Out Cross Validation

Estrictamente hablando, el método de Hold Out Cross Validation, no debería ser considerado como técnica de validación cruzada porque los datos realmente nunca son cruzados, sin embargo, tomando los elementos al azar y repitiendo el procedimiento algunas veces (más de 10), la estimación del error total, es aceptable (resampling). Una ventaja de esta técnica es que su aplicación no requiere altos costos computacionales. Este método consiste en tomar el conjunto completo de datos disponibles y dividirlo en dos subconjuntos; el primero de ellos aplicable a la fase de entrenamiento y el segundo para la fase de prueba. Generalmente, se consideran dos terceras partes para el conjunto de entrenamiento y la otra tercera parte para el conjunto de prueba (Figura 3.2.iv).

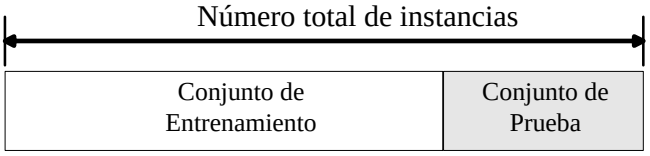


Figura 3.2.iv *Estimación del error usando Hold Out cross validation.*

Cabe mencionar que, cuando se tienen pocas instancias disponibles o existe alta dispersión en los datos, puede presentarse alta varianza en la estimación del error total; consecuentemente, es recomendable aplicar criterios estadísticos adicionales ya que el

resultado depende fuertemente de las instancias que hayan sido seleccionadas para formar el conjunto de entrenamiento.

3.2.4.2 K-fold Cross Validation

Una alternativa para minimizar los efectos de la dispersión de los datos y las debilidades inherentes a la técnica de Hold Out cross validation, es el método de *K*-fold cross validation. En este método el conjunto de datos disponibles es dividido en *K* particiones que dan lugar a *K* subconjuntos mutuamente excluyentes. Para cada una de las *K* estimaciones de error, uno de los *K* subconjuntos es usado como conjunto de prueba y los otros *K*-1 restantes son agrupados para formar el conjunto de entrenamiento (Figura 3.2.v).

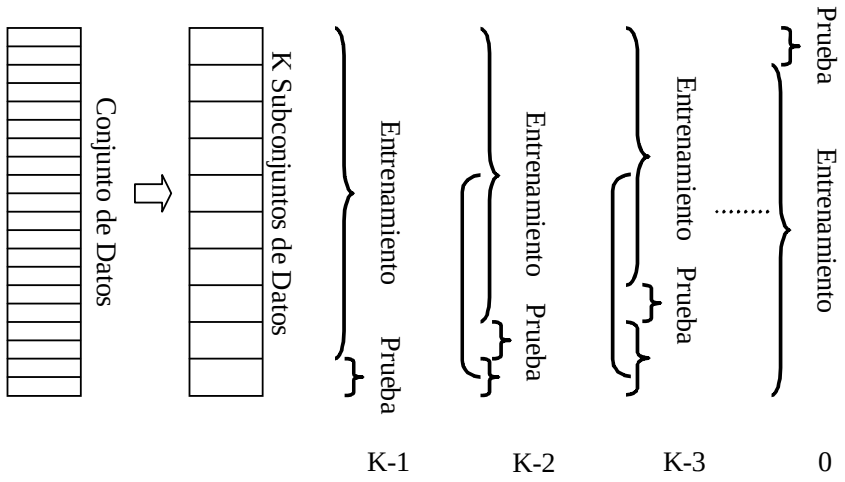


Figura 3.2.v *Estimación del error usando K-fold cross validation.*

El procedimiento es repetido *K* veces para asegurar que los *K* subconjuntos han sido utilizados en la fase de prueba, de modo tal, que se tienen *K* estimaciones de error E_i, que serán promediadas para obtener el error total de predicción E del modelo en cuestión.

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Una de las ventajas que tiene este método es que, aún cuando se disponga de pocas instancias para ambas fases (entrenamiento/prueba), la estimación total del error E, es confiable, debido a que todos los patrones son considerados en la fase de prueba, en otras palabras, todos los patrones se consideran una vez en la fase de prueba y *K*-1 veces en la fase de entrenamiento. Cabe mencionar que la varianza en la estimación del error de predicción disminuye conforme el valor de *K* aumenta.

La desventaja evidente de este método es que la fase de entrenamiento tiene que ejecutarse en *K* ocasiones, lo que implica *K* veces más tiempo de computo que la técnica de Hold Out cross validation.

3.2.4.3 Leave-One-Out Cross Validation

Este método de validación cruzada puede verse como la versión de *K*-fold cross validation cuando *K* es igual al número *N* de instancias disponibles. Esto implica que, para cada una de las *N* veces que se realice la fase de aprendizaje, será necesario considerar *N*-1 instancias para entrenamiento y solo una instancia para prueba, de modo que, se obtienen *N* estimaciones parciales de error. Análogamente, se puede calcular el error total en la precisión predictiva usando la siguiente expresión:

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i$$

Una de las ventajas de este método es que, el problema de alta varianza en la estimación del error total *E*, derivado ya sea de un número reducido de instancias disponibles o de la dispersión en los datos, se vuelve casi despreciable.

Esto no debiera sorprender, ya que conforme crece el número de particiones, decrece el tamaño de cada una de éstas, haciendo que el número de instancias involucradas en la fase de aprendizaje sea menor, consecuentemente, la varianza tiende a disminuir.

Contrario a lo que se pudiera pensar, la precisión predictiva de un clasificador no gira en torno a una mayor cantidad de instancias procesadas en la fase de aprendizaje. Algunos trabajos de investigación [52], sugieren que entre mayor sea la cantidad de instancias conocidas durante el entrenamiento del clasificador (*overfitting*), pueden surgir comportamientos no deseados en la fase de recuperación, es decir, únicamente se observará comportamiento adecuado con los patrones conocidos, pero no se asegura que la precisión predictiva sobre nuevas instancias sea la misma [53].

Se puede concluir que el enfoque para selección de rasgos conocido como Envoltura (*wrapper*), se comporta mejor cuando se tienen suficientes instancias para establecer subconjuntos independientes, tanto para la fase de entrenamiento como para la de prueba. Cabe mencionar que, este enfoque puede alcanzar índices de precisión predictiva notables, asumiendo los costos computacionales derivados de una búsqueda exhaustiva entre todas las posibles combinaciones de rasgos.

Este método generalmente es aplicado cuando el número de rasgos f presentes en cada una de las instancias es chico, ya que el número total de posibles combinaciones a evaluar es 2^f.

Toda vez que, los costos computacionales de la búsqueda del subconjunto de rasgos óptimos sean asumidos, pueden aplicarse métodos de validación cruzada para minimizar los efectos asociados a la dispersión en los datos, así como la alta varianza en la estimación del error total de precisión predictiva.

Aún en situaciones cuando no se tienen suficientes patrones para llevar a cabo las dos fases (entrenamiento/prueba) con diferentes instancias, el método de validación cruzada más comúnmente aplicado en la estimación del desempeño de clasificadores de patrones es el de *K*-fold cross validation con un valor de *K*=10.

3.3 Diseño de Sistemas Digitales

Tradicionalmente, el diseño digital era un proceso manual en el que la organización de los circuitos se llevaba a cabo usando herramientas de captura de diagramas esquemáticos. Con el paso del tiempo, esta práctica cada vez es menos común como recomendable; excepto en aquellos casos donde la naturaleza del diseño lo justifique.

Actualmente, para el diseño de los sistemas digitales se han adoptado nuevas metodologías (*Top-Down*) que permiten alcanzar no solo niveles más complejos de abstracción, sino que además es posible obtener una representación jerárquica de los elementos que conforman el sistema. Esta representación modular de los elementos que constituyen un sistema digital, ha provocado que los lenguajes de descripción de hardware (HDL) cobren relevancia no solo para el diseño del sistema digital en si mismo, sino también para establecer restricciones en las etapas posteriores (aplicación de estímulos de entrada y verificación de bancos de pruebas).

Algunas de las ventajas asociadas al uso de las prácticas actuales de diseño de sistemas digitales son:

- Reutilización de los elementos involucrados
- Incremento en la flexibilidad a los cambios de diseño
- Ágil exploración de arquitecturas alternativas de procesamiento
- Garantía de operación, verificación y corrección de fallas

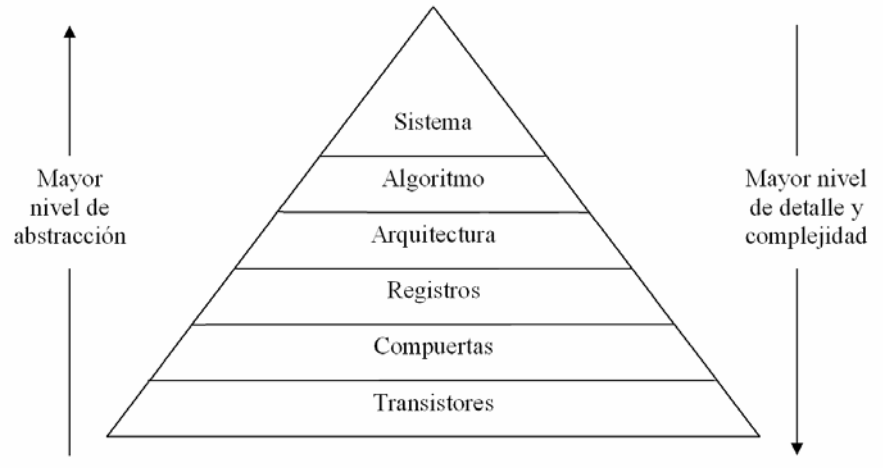


Figura 3.3.vi *Representación modular de los elementos que constituyen un sistema digital. Cuando el nivel de abstracción disminuye, el grado de detalle y complejidad aumenta.*

3.3.1 Dispositivos Basados en Lógica Reconfigurable

Anteriormente, los dispositivos comúnmente usados en el diseño de sistemas digitales eran circuitos integrados de funcionalidad fija. La operación de estos componentes, estaba completamente definida por el fabricante y fuertemente restringida por el proceso de manufactura; tal es el caso de la tan conocida familia 7400 de circuitos integrados basados en lógica de transistor-transistor (TTL), que permitieron la construcción de los primeros equipos de computo.

Actualmente, en el ámbito del diseño y la fabricación de prototipos, existe una clara tendencia hacia el uso de dispositivos basados en lógica reconfigurable. La razón principal del por qué son un recurso frecuentemente empleado, resulta del hecho de que la funcionalidad en este tipo de dispositivos no es fija, consecuentemente, tanto en los circuitos integrados de aplicación específica (ASIC) como en los arreglos de compuertas programables (FPGAs), la funcionalidad del dispositivo queda completamente en manos del diseñador.

3.3.1.1 ASICs

Los circuitos integrados de aplicación específica son dispositivos que han sido concebidos para su uso en aplicaciones particulares. Generalmente un ASIC es diseñado para productos donde gran parte de la electrónica necesaria puede ser implementada sobre un chip de silicio y el número de unidades que serán fabricadas es elevado. Actualmente, dependiendo del tipo de proceso de fabricación, existen dos categorías principales: ASIC *parcialmente terminados* y ASIC *completamente terminados*. Aun cuando para ambos tipos de circuitos integrados, el proceso de manufactura consiste en la obtención de arreglos de transistores implementados en un chip de silicio, la diferencia principal radica en el número de capas disponibles para el diseñador. Los ASIC *parcialmente terminados*, en su mayoría son implementados a partir de arreglos de compuertas lógicas donde una sola capa (generalmente la superior), define la funcionalidad del dispositivo. El caso contrario ocurre en los ASIC *completamente terminados,* donde todas las capas definen la funcionalidad del dispositivo; este hecho pone de manifiesto que algún cambio en el diseño de un ASIC *parcialmente terminado*, implica la modificación de una sola capa del circuito integrado, mientras que el mismo cambio en el diseño de un ASIC *completamente terminado*, implica la modificación total del circuito integrado, en otras palabras, un rediseño completo.

Generalmente, por los elevados costos del proceso de fabricación, este tipo de situaciones son inaceptables, haciendo de los ASIC una opción no viable para el diseño de prototipos experimentales.

3.3.1.2 FPGAs

Los arreglos de compuertas programables, son dispositivos semiconductores principalmente estructurados por celdas lógicas interconectadas mediante una matriz de switches programables, Figura 3.3.ii. Típicamente, cada una de estas celdas es capaz de llevar a cabo tareas combinatorias o secuenciales, de manera que cualquier diseño puede ser implementado mediante la descripción funcional de cada celda lógica y la selección de los switches adecuados que permiten la interconexión entre los elementos involucrados.

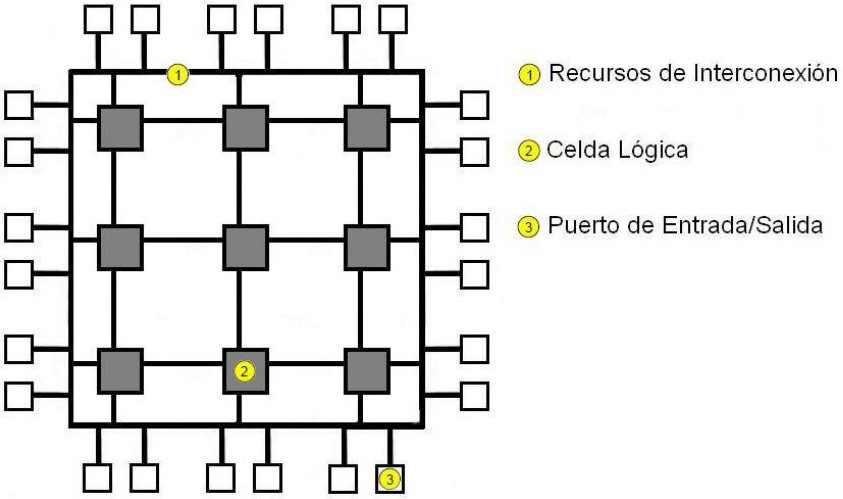


Figura 3.3.vii *Arquitectura General de un FPGA.*

Actualmente, los FPGAs comerciales emplean alguno de los siguientes elementos para la fabricación de las celdas lógicas:

- Pares de transistores
- Compuertas lógicas NAND o XOR de dos entradas
- Multiplexores
- Look-up tables (LUTs)

Existen tres tecnologías mayormente difundidas para la programación de los switches que interconectan cada celda lógica:

- Memoria Estática de Acceso Aleatorio (SRAM), donde el proceso de conmutación lo realiza un transistor de paso, controlado por un bit de estado, almacenado en la memoria.

- Memoria de solo lectura borrable y programable (EPROM), donde el proceso de conmutación puede anularse al aplicar cargas eléctricas.

- Antifusible, donde al programar el dispositivo, se forman rutas de baja resistencia.

Cabe mencionar que, solo será cubierta la descripción de la tecnología SRAM, debido a que el modelo propuesto en el Capítulo 4, está basado en un FPGA de este tipo.

La funcionalidad de cada celda lógica de tecnología SRAM, es controlada mediante compuertas de paso que actúan como elementos de conmutación para formar rutas de señalización, Figura 3.3.iii (a). Cuando se tiene almacenado en la celda lógica un nivel de voltaje alto (valor lógico “1”), la compuerta de paso actúa como corto circuito, permitiendo la interconexión de dos segmentos independientes. La salida de cada multiplexor, es controlada mediante la interconexión de celdas lógicas y líneas de entrada del elemento en cuestión, de modo tal, que a la salida del multiplexor se tenga solo una de las entradas del mismo, Figura 3.3.iii (b).

Figura 3.3.viii *Configuración de celdas lógicas. (a) Control por compuerta de paso, (b) Multiplexor controlado por bits de estado.*

Aun cuando la tecnología SRAM usada en los FPGAs proporciona grandes ventajas como la reprogramación iterativa y el aprovechamiento de los recursos libres para conmutar entre aplicaciones, su mayor desventaja es el área necesaria para implementar cada celda lógica; cuando menos cinco transistores son necesarios para su funcionamiento y un transistor adicional como elemento de conmutación.

Particularmente, el FPGA utilizado para la realización del presente trabajo de tesis, pertenece a la familia Spartan3 del fabricante Xilinx, el cual cuenta con bloques lógicos configurables (CLBs) interconectados mediante recursos versátiles de enrutamiento jerarquizado, Figura 3.3.iv.

Cada CLB puede ser programado para realizar una gran variedad de operaciones lógicas e incluso funcionar como elemento de almacenamiento de datos.

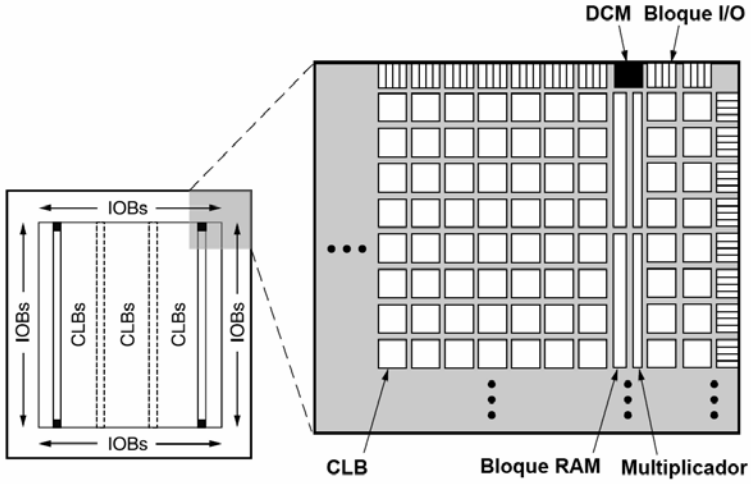


Figura 3.3.ix *Arquitectura de la Familia Spartan3 de Xilinx.*

Los principales elementos que caracterizan a esta familia de dispositivos se describen a continuación:

- Los bloques de entrada/salida (IOBs), son los encargados del control del flujo de datos hacia las terminales de entrada/salida.

- Los Bloques RAM, proporcionan espacio para el almacenamiento de datos, permitiendo ciclos de lectura y escritura simultáneos bajo la modalidad de memoria de puerto dual.

- Los Bloques Multiplicadores, aceptan como argumentos números binarios de 18 bits.

- El Administrador de Reloj Digital (DCM), resuelve problemas comunes de retraso, desfase, multiplicación y división de la señal de reloj principal (reloj maestro).

3.3.2 Lenguajes de Descripción de Hardware

En un principio, los lenguajes de descripción de hardware (HDL) fueron usados meramente como un conjunto de sentencias simples, para definir la interconexión entre los elementos involucrados en el diseño de un sistema digital. A estos lenguajes se les llamó *Net list*, puesto que eran simplemente eso, un archivo con una lista de conexiones.

Años mas tarde, el nivel de abstracción requerido para la verificación funcional de cada uno de estos elementos, puso de manifiesto la falta de herramientas apropiadas para simulación, consecuentemente, los lenguajes de descripción de hardware evolucionaron hacia niveles más altos de abstracción, facilitando la obtención de modelos sintetizables; en otras palabras, cuanto mayor es la funcionalidad de las sentencias disponibles en el lenguaje, mayor es la similitud entre la etapa de modelado y la de verificación funcional.

Aunque existen dos lenguajes principales para la descripción de hardware: VHDL y Verilog, en el presente trabajo de tesis se tomó la decisión de apegarse al primero de ellos.

3.3.2.1 VHDL

VHDL surge como una iniciativa del Departamento de Defensa de los Estados Unidos, para la aplicación de técnicas de diseño jerárquico que faciliten no solo el proceso de documentación de un circuito digital, sino que además, permitan modelar un sistema digital completo desde diferentes niveles de abstracción, que van desde un panorama algorítmico hasta el nivel de compuertas.

Algunas ventajas que ofrece VHDL son:

- Gran cantidad de tipos de datos predefinidos en el lenguaje.
- Las unidades diseñadas pueden ser compiladas por separado.
- Los procedimientos y funciones pueden incluirse en paquetes separados.
- Los paquetes están disponibles para cualquier unidad.
- Mayor número de constructores para modelado de alto nivel.
- Permite tener unidades compiladas, almacenadas en librerías.
- Permite establecer llamadas concurrentes a procedimientos.

3.3.2.1.1 Unidades de diseño en VHDL

Para cualquiera de las dos etapas (diseño o simulación) previas a la implementación de un modelo sobre dispositivos basados en lógica reconfigurable, es necesario dividir el diseño en bloques abstractos, conocidos como componentes o unidades.

La unidad de diseño *entity*, proporciona la información concerniente a los puertos de entrada y/o salida, mientras que la unidad de diseño *architecture*, proporciona la descripción funcional de la entidad; cabe mencionar que, es posible tener múltiples descripciones funcionales de la misma unidad.

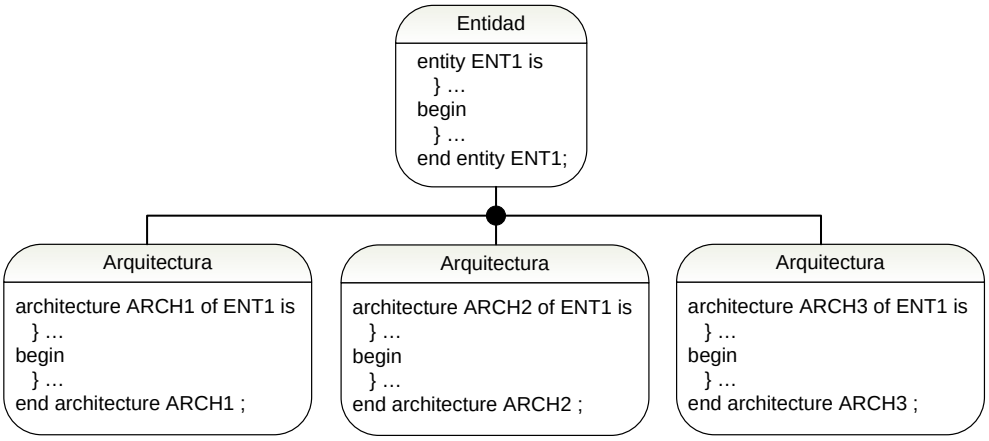


Figura 3.3.x *Unidades de diseño en VHDL.*

3.3.2.1.2 Estructura del código en VHDL

Existen tres tipos de enunciados que permiten instanciar múltiples unidades de diseño:

- Declaraciones. Estas deben aparecer antes de ser instanciada una unidad de diseño.

- Postulados Secuenciales. Este tipo de enunciados son ejecutados dependiendo de las condiciones del flujo de los datos.

- Postulados Concurrentes. Este tipo de enunciados son ejecutados en paralelo, es decir, en el mismo instante de tiempo e independientemente de otros postulados concurrentes o secuenciales.

3.3.2.1.3 Objetos de datos en VHDL

Existen cuatro tipos de objetos de datos en VHDL:

Constantes. Una constante puede mantener un único valor. Este valor es asignado antes de iniciar el proceso de simulación, y no puede ser cambiado durante la ejecución.

Ejemplo:

constant AnchoDePalabra : integer := 16;

Variables. Una variable puede ser usada para almacenar valores temporales dentro de un proceso; sin embargo, estos valores pueden cambiar en diferentes instantes de tiempo.

Ejemplo:

```
variable CuatroBits : bit_vector(3 downto 0);
```

Señales. Una señal puede mantener una lista de valores, que incluyen el valor actual y un conjunto de valores futuros. Las señales pueden ser vistas como conexiones (alambres en un circuito), elementos de almacenamiento simple (*flip-flops*) e incluso esquemas de almacenamiento agrupado (registros).

Ejemplo:

```
signal reloj : bit;
signal AnchoDeBus : bit_vector(15 downto 0);
signal RegistroTemp : bit_vector(31 downto 0);
signal Propagacion : time := 10 ns;
```

Archivos. Un objeto de tipo archivo (*file*), hace referencia a un sistema de archivos. Este tipo de objeto no es sintetizable; sin embargo, es de vital importancia durante la etapa de simulación, ya que permite proporcionar los estímulos de entrada y obtener archivos de salida.

Ejemplo:

```
file ArchEntrada : text open read_mode is “./arch_entrada.txt”;
file ArchSalida : text open write_mode is “./arch_salida.txt”;
```

CAPÍTULO 4

Modelo Propuesto

Este capítulo es el más relevante del presente documento de tesis. Primeramente, se establecen los operadores que dan paso a la definición de una máscara binaria, representada de manera vectorial, que conlleva a la identificación de información irrelevante para fines de clasificación. Asimismo, se define un criterio de optimalidad que identifica el valor de la máscara que proporciona el menor número de rasgos y además incrementa el índice de clasificación.

Posteriormente, se aplica esta técnica de reducción dimensional de los datos, para obtener arquitecturas de computo parametrizadas, capaces de llevar a cabo tareas de aprendizaje y recuperación de patrones.

Finalmente, se muestra la representación esquemática a nivel de compuertas lógicas de los dos bloques principales que permiten lograr la implementación en hardware del modelo de Memorias Asociativas Alfa-Beta, bajo un esquema paralelo tanto en su fase de aprendizaje como de recuperación.

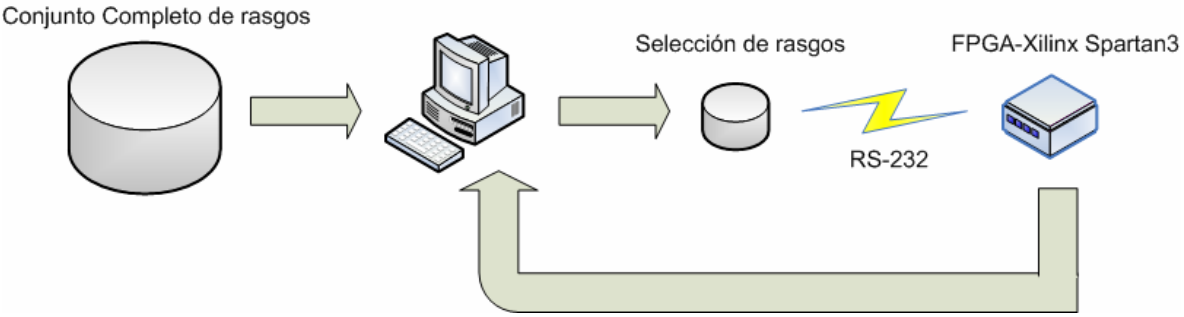


Figura 4.0.i *Modelo Propuesto.*

4.1 Reducción Dimensional de los datos

En esta sección se describe el algoritmo que permite obtener una representación equivalente (dimensionalmente menor), de un conjunto fundamental de patrones.

El punto neurálgico sobre el cual descansa el algoritmo HCM (Clasificación Híbrida con Enmascaramiento) [44], es la obtención de una máscara que representa un subconjunto de rasgos que conservan información relevante para fines de clasificación de patrones.

De todas las máscaras obtenidas que mejoran el índice de clasificación, la mejor de ellas es, la que indica el menor subconjunto de rasgos.

Esto permite optimizar el tamaño de los registros y de las unidades funcionales; logrando así, importantes reducciones tanto en la cantidad de lógica sintetizable como de recursos de almacenamiento necesarios para llevar a cabo tareas de aprendizaje y recuperación de patrones [45].

Para entender el criterio aplicado para la selección de la máscara óptima, algunas definiciones son necesarias.

Definición 1. Sea f el número de rasgos del conjunto original de datos.

Definición 2. Sea r un índice, donde $r \in \{1,2,...,(2^f - 1)\}$.

Definición 3. Sea e^r un vector de enmascaramiento de tamaño *n*, representado como:

$$e^r = \begin{pmatrix} e_1^r \\ e_2^r \\ \vdots \\ e_n^r \end{pmatrix} \in B^n \quad ; \quad B = \{0,1\}$$

Definición 4. Sea $\longleftrightarrow$ una nueva operación llamada IntToVector que toma $r \in \{1,2,...,(2^f - 1)\}$ y entrega un vector e^r con el valor de *r* expresado en forma binaria.

Ejemplo: Si *r* =11, entonces $\longleftrightarrow e^r$ entrega un vector columna con el valor de *r* expresado en forma binaria, así, el vector columna obtenido es:

$$e^{11} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

donde e_n^r es el bit menos significativo (LSB).

Definición 5. Sea $\|$ una nueva operación llamada MagVector, la cual toma un vector columna e^r de tamaño *n*, y regresa un valor entero y positivo de acuerdo a la siguiente regla:

$$\|e^r = \sum_{j=1}^{n} (e_j^r \wedge 1)$$

donde $\wedge$ es el operador lógico AND.

Definición 6. Sean $x^1, x^2, ..., x^p$ elementos del conjunto de patrones de entrada, y sea $\overline{x}$ el vector medio de todos ellos, donde:

$$\overline{x} = \frac{1}{p} \sum_{\mu=1}^{p} x^\mu$$

Definición 7. Sean $x^{1'}, x^{2'}, ..., x^{p'}$ elementos del conjunto de patrones trasladados, mediante la siguiente expresión:

$$x^{\mu'} = \left[x^{\mu} \quad - \quad \overline{x} \right]$$

donde

Definición 8. Sea y_i^{μ}, la *i*-ésima componente del μ-esimo patrón de salida del conjunto fundamental, recuperado mediante la siguiente regla:

$$y_i^{\mu} = \begin{cases} 1 \quad si \quad \sum_{j=1}^{n} m_{ij} \cdot \left(x_j^{\mu} \cdot e_j^{r} \right) = \vee_{h=1}^{m} \left[\sum_{j=1}^{n} m_{hj} \cdot \left(x_j^{\mu} \cdot e_j^{r} \right) \right] \\ 0 \quad en \quad otro \quad caso \end{cases}$$

donde $r \in \{1,2,...,(2^f - 1)\}$ y $\mu \in \{1,2,...,p\}$

4.1.1 Algoritmo HCM para la reducción dimensional de los datos

Los pasos que sigue el algoritmo HCM para encontrar la máscara óptima son los siguientes:

1. Sea X un conjunto fundamental de patrones de entrada de dimensión *n* con valores reales en sus componentes.

2. A cada uno de los patrones de entrada que pertenece a la clase *k* se le asigna el vector formado por ceros, excepto en la *k*-ésima posición, donde el valor es uno.

3. Se calcula el vector medio del conjunto fundamental de patrones.

4. Se toman las coordenadas del vector medio a manera de centro de un nuevo conjunto de coordenadas.

5. Se realiza la traslación de todos los patrones de conjunto fundamental.

6. Aplicamos la fase de aprendizaje.

7. Aplicamos la fase de recuperación.

8. Se almacena el índice de clasificación.

9. Se incrementa el valor de la máscara.

10. ¿Se alcanzó el valor máximo de la máscara?
    - 10.1 *Si*, entonces continúa con el paso 11.
    - 10.2 *No*, entonces continúa con el paso 7.

11. Se aplica el criterio de la Definición 5, para identificar la máscara óptima.

12. FIN de ejecución.

A continuación, se ilustra el proceso de reducción dimensional de los datos mediante un ejemplo:

Sean $\{(x^{\mu}, y^{\mu}) \mid \mu = 1,2,\ldots,p\}$ los elementos que definen el conjunto fundamental de patrones.

En total se tienen 8 asociaciones de patrones, agrupados en dos diferentes clases, con igual número de patrones de entrada en cada una de las clases.

Sean $x^1, x^2, \cdots, x^4$ patrones de entrada del conjunto fundamental, pertenecientes a la clase 1.

$$x^1 = \begin{pmatrix} 1.40 \\ 1.70 \\ 0.69 \\ -1.10 \\ 0.69 \\ 2.14 \\ -0.30 \\ -0.27 \end{pmatrix} \quad x^2 = \begin{pmatrix} 1.50 \\ -1.32 \\ 0.66 \\ 1.40 \\ 0.66 \\ -0.76 \\ -0.40 \\ -0.27 \end{pmatrix} \quad x^3 = \begin{pmatrix} 1.20 \\ -1.27 \\ 0.56 \\ -1.20 \\ 0.56 \\ 1.10 \\ -0.50 \\ -0.27 \end{pmatrix} \quad x^4 = \begin{pmatrix} 1.50 \\ 1.46 \\ -0.05 \\ -1.20 \\ 0.18 \\ -0.30 \\ -0.60 \\ -0.27 \end{pmatrix}; \; y^1 = y^2 = y^3 = y^4 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Sean $x^5, x^6, \cdots, x^8$ patrones de entrada del conjunto fundamental, pertenecientes a la clase 2.

$$x^5 = \begin{pmatrix} -1.20 \\ -1.10 \\ -0.42 \\ 1.40 \\ 0.28 \\ -0.54 \\ 0.07 \\ -0.02 \end{pmatrix} \quad x^6 = \begin{pmatrix} -1.20 \\ -1.58 \\ -0.44 \\ 0.21 \\ 0.58 \\ -0.74 \\ 0.13 \\ -0.27 \end{pmatrix} \quad x^7 = \begin{pmatrix} 1.40 \\ -1.10 \\ -0.28 \\ 1.20 \\ -0.42 \\ -1.20 \\ 0.13 \\ -0.28 \end{pmatrix} \quad x^8 = \begin{pmatrix} 1.10 \\ 1.60 \\ -0.44 \\ 1.40 \\ -0.42 \\ -0.84 \\ 0.13 \\ -0.29 \end{pmatrix}; y^5 = y^6 = y^7 = y^8 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Tal como se especifica en el paso 3, se calcula el vector medio del conjunto fundamental de patrones.

$$\overline{x} = \frac{1}{p}\sum_{\mu=1}^{p} x^{\mu} = \frac{1}{8}\left[x^1 + x^2 + \cdots + x^8\right]$$

$$\overline{x} = \frac{1}{8}\left[\begin{pmatrix}1.40\\1.70\\0.69\\-1.10\\0.69\\2.14\\-0.30\\-0.27\end{pmatrix} + \begin{pmatrix}1.50\\-1.32\\0.66\\1.40\\0.66\\-0.76\\-0.40\\-0.27\end{pmatrix} + \cdots + \begin{pmatrix}1.10\\1.60\\-0.44\\1.40\\-0.42\\-0.84\\0.13\\-0.29\end{pmatrix}\right] = \begin{pmatrix}0.71\\-0.20\\0.04\\0.26\\0.27\\-0.14\\-0.17\\-0.24\end{pmatrix}$$

Una vez teniendo el vector medio, continuamos con la traslación de todos los patrones del conjunto fundamental, aplicando la expresión de la Definición 7.

$$x^{\mu'} = \left[x^{\mu} \quad - \quad \overline{x}\right] \; ; \; \mu \in \{1,2,...,p\}$$

Una vez trasladados todos los vectores que conforman el conjunto fundamental, se obtiene un nuevo conjunto de vectores trasladados, denotado como: $\left\{(x^{\mu'}, y^{\mu'}) \mid \mu \in 1,2,\ldots,p\right\}$.

$$x^{1'} = \begin{pmatrix}0.69\\1.90\\0.66\\-1.36\\0.43\\2.28\\-0.13\\-0.03\end{pmatrix} \quad x^{2'} = \begin{pmatrix}0.79\\-1.12\\0.63\\1.14\\0.40\\-0.62\\-0.23\\-0.03\end{pmatrix} \quad x^{3'} = \begin{pmatrix}0.49\\-1.07\\0.52\\-1.46\\0.29\\1.24\\-0.33\\-0.03\end{pmatrix} \quad x^{4'} = \begin{pmatrix}0.79\\1.66\\-0.09\\-1.46\\-0.08\\-0.16\\-0.43\\-0.03\end{pmatrix}; \; y^{1'} = y^{2'} = y^{3'} = y^{4'} = \begin{pmatrix}1\\0\end{pmatrix}$$

$$x^{5'} = \begin{pmatrix}-1.91\\-0.90\\-0.45\\1.14\\0.01\\-0.40\\0.24\\0.22\end{pmatrix} \quad x^{6'} = \begin{pmatrix}-1.91\\-1.38\\-0.48\\-0.05\\0.32\\-0.60\\0.30\\-0.03\end{pmatrix} \quad x^{7'} = \begin{pmatrix}0.69\\-0.90\\-0.31\\0.94\\-0.68\\-1.06\\0.30\\-0.04\end{pmatrix} \quad x^{8'} = \begin{pmatrix}0.39\\1.80\\-0.47\\1.14\\-0.68\\-0.70\\0.30\\-0.05\end{pmatrix}; \; y^{5'} = y^{6'} = y^{7'} = y^{8'} = \begin{pmatrix}0\\1\end{pmatrix}$$

Tal como se indica en el paso 6 del algoritmo, se toma el conjunto fundamental de patrones trasladados, para obtener las *p* matrices requeridas en la fase de aprendizaje.

$$y^{\mu'} \cdot \left(x^{\mu'}\right)^t = \begin{pmatrix} y_1^{\mu'}x_1^{\mu'} & y_1^{\mu'}x_2^{\mu'} & \cdots & y_1^{\mu'}x_j^{\mu'} & \cdots & y_1^{\mu'}x_n^{\mu'} \\ y_2^{\mu'}x_1^{\mu'} & y_2^{\mu'}x_2^{\mu'} & \cdots & y_2^{\mu'}x_j^{\mu'} & \cdots & y_2^{\mu'}x_n^{\mu'} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_i^{\mu'}x_1^{\mu'} & y_i^{\mu'}x_2^{\mu'} & \cdots & y_i^{\mu'}x_j^{\mu'} & \cdots & y_i^{\mu'}x_n^{\mu'} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_m^{\mu'}x_1^{\mu'} & y_m^{\mu'}x_2^{\mu'} & \cdots & y_m^{\mu'}x_j^{\mu'} & \cdots & y_m^{\mu'}x_n^{\mu'} \end{pmatrix}$$

El clasificador se obtiene al sumar la *p* matrices resultantes del paso anterior.

$$C = \sum_{\mu=1}^{p} \left[\left(y^{\mu'}\right) \bullet \left(x^{\mu'}\right)^t \right]$$

$$C = \begin{bmatrix} 2.750 & 1.375 & 1.718 & -3.155 & 1.032 & 2.752 & -1.129 & -0.117 \\ -2.750 & -1.375 & -1.718 & 3.155 & -1.032 & -2.752 & 1.129 & 0.117 \end{bmatrix}$$

Una vez calculado el clasificador, se puede continuar con la fase de recuperación.

$$y^{\mu'} = \left[C \bullet (x^{\mu'} \bullet e^r) \right]$$

donde $r \in \{1,2,...,(2^f - 1)\}$ y $\mu \in \{1,2,...,p\}$.

Como puede observarse, en la expresión anterior aparece el vector de enmascaramiento afectando a cada uno de los patrones de entrada trasladados. Al aplicar una máscara cualquiera, el índice de clasificación se verá modificado; consecuentemente, se tendrán *r* estimaciones diferentes del índice de clasificación.

Existen dos máscaras que no forman parte de la solución óptima. La primera de ellas es la “máscara absoluta”, la cual provoca que todas las componentes de un patrón de entrada sean cero. Al tratar de recuperar un patrón con tales características, el resultado es totalmente ambiguo, es decir, no es posible diferenciar a que clase pertenece.

$$e^0 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

La segunda de ellas es la “máscara nula”. Esta se aparta del principio fundamental de reducción dimensional de los datos, ya que no conduce a la identificación de información irrelevante y por consiguiente, no es posible eliminar algún rasgo.

$$e^{2^{n-1}} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Otra desventaja que presenta el uso de la “máscara nula”, es que, en caso de que alguna de las componentes de los vectores de entrada contenga información irrelevante, esta será tomada en cuenta, provocando que el índice de clasificación disminuya.

Aplicando la regla de recuperación expresada en la Definición 8, se observa que al tomar todas las componentes de los vectores de entrada, el patrón de salida $y^{2'}$ asociado al patrón de entrada $x^{2'}$, no es correctamente recuperado.

$$y^{1'} = \left[C \bullet (x^{1'} \bullet e^{255})\right] = \begin{bmatrix} 16.81 \\ -16.81 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{1'} = y^{1} \Rightarrow \quad Correcto$$

$$y^{2'} = \left[C \bullet (x^{2'} \bullet e^{255})\right] = \begin{bmatrix} -2.91 \\ 2.91 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{2'} \neq y^{2} \Rightarrow \quad Incorrecto$$

$$y^{3'} = \left[C \bullet (x^{3'} \bullet e^{255})\right] = \begin{bmatrix} 9.50 \\ -9.50 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{3'} = y^{3} \Rightarrow \quad Correcto$$

$$y^{4'} = \left[C \bullet (x^{4'} \bullet e^{255})\right] = \begin{bmatrix} 8.89 \\ -8.89 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{4'} = y^{4} \Rightarrow \quad Correcto$$

$$y^{5'} = \left[C \bullet (x^{5'} \bullet e^{255})\right] = \begin{bmatrix} -12.24 \\ 12.24 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{5'} = y^{5} \Rightarrow \quad Correcto$$

$$y^{6'} = \left[C \bullet (x^{6'} \bullet e^{255})\right] = \begin{bmatrix} -9.46 \\ 9.46 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{6'} = y^{6} \Rightarrow \quad Correcto$$

$$y^{7'} = \left[C \bullet (x^{7'} \bullet e^{255})\right] = \begin{bmatrix} -6.78 \\ 6.78 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{7'} = y^{7} \Rightarrow \quad Correcto$$

$$y^{8'} = \left[C \bullet (x^{8'} \bullet e^{255})\right] = \begin{bmatrix} -3.81 \\ 3.81 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{8'} = y^{8} \Rightarrow \quad Correcto$$

Después de un proceso iterativo, para encontrar un valor de enmascaramiento que nos permita no solo reducir dimensionalmente los datos, sino que además incremente el índice de clasificación, se encuentra que la máscara e^{43}, cumple con estos dos objetivos.

$$e^{43} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Usando la máscara e^{43} en la regla de recuperación expresada en la Definición 8, se observa que todo el conjunto fundamental es recuperado correctamente.

$$y^{1'} = \left[C \bullet (x^{1'} \bullet e^{43})\right] = \begin{bmatrix} 1.72 \\ -1.72 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{1'} = y^{1} \Rightarrow \quad Correcto$$

$$y^{2'} = \left[C \bullet (x^{2'} \bullet e^{43})\right] = \begin{bmatrix} 1.75 \\ -1.75 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{2'} = y^{2} \Rightarrow \quad Correcto$$

$$y^{3'} = \left[C \bullet (x^{3'} \bullet e^{43})\right] = \begin{bmatrix} 1.58 \\ -1.58 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{3'} = y^{3} \Rightarrow \quad Correcto$$

$$y^{4'} = \left[C \bullet (x^{4'} \bullet e^{43})\right] = \begin{bmatrix} 0.25 \\ -0.25 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{4'} = y^{4} \Rightarrow \quad Correcto$$

$$y^{5'} = \left[C \bullet (x^{5'} \bullet e^{43})\right] = \begin{bmatrix} -1.06 \\ 1.06 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{5'} = y^{5} \Rightarrow \quad Correcto$$

$$y^{6'} = \left[C \bullet (x^{6'} \bullet e^{43})\right] = \begin{bmatrix} -0.82 \\ 0.82 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{6'} = y^{6} \Rightarrow \quad Correcto$$

$$y^{7'} = \left[C \bullet (x^{7'} \bullet e^{43})\right] = \begin{bmatrix} -1.57 \\ 1.57 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{7'} = y^{7} \Rightarrow \quad Correcto$$

$$y^{8'} = \left[C \bullet (x^{8'} \bullet e^{43})\right] = \begin{bmatrix} -1.85 \\ 1.85 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{8'} = y^{8} \Rightarrow \quad Correcto$$

Para mostrar la eficacia del enfoque propuesto en la obtención de una representación equivalente (dimensionalmente menor), a continuación, se alteran los patrones de entrada trasladados, aplicando la máscara e^{43}.

$$x^{1'} = \begin{pmatrix} 0 \\ 0 \\ 0.66 \\ 0 \\ 0.43 \\ 0 \\ -0.13 \\ -0.03 \end{pmatrix} \quad x^{2'} = \begin{pmatrix} 0 \\ 0 \\ 0.63 \\ 0 \\ 0.40 \\ 0 \\ -0.23 \\ -0.03 \end{pmatrix} \quad x^{3'} = \begin{pmatrix} 0 \\ 0 \\ 0.52 \\ 0 \\ 0.29 \\ 0 \\ -0.33 \\ -0.03 \end{pmatrix} \quad x^{4'} = \begin{pmatrix} 0 \\ 0 \\ -0.09 \\ 0 \\ -0.08 \\ 0 \\ -0.43 \\ -0.03 \end{pmatrix}; \; y^{1'} = y^{2'} = y^{3'} = y^{4'} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$x^{5'} = \begin{pmatrix} 0 \\ 0 \\ -0.45 \\ 0 \\ 0.01 \\ 0 \\ 0.24 \\ 0.22 \end{pmatrix} \quad x^{6'} = \begin{pmatrix} 0 \\ 0 \\ -0.48 \\ 0 \\ 0.32 \\ 0 \\ 0.30 \\ -0.03 \end{pmatrix} \quad x^{7'} = \begin{pmatrix} 0 \\ 0 \\ -0.31 \\ 0 \\ -0.68 \\ 0 \\ 0.30 \\ -0.04 \end{pmatrix} \quad x^{8'} = \begin{pmatrix} 0 \\ 0 \\ -0.47 \\ 0 \\ -0.68 \\ 0 \\ 0.30 \\ -0.05 \end{pmatrix}; \; y^{5'} = y^{6'} = y^{7'} = y^{8'} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Del mismo modo se aplica la máscara e^{43} al clasificador.

$$C = \begin{bmatrix} 0 & 0 & 1.718 & 0 & 1.032 & 0 & -1.129 & -0.117 \\ 0 & 0 & -1.718 & 0 & -1.032 & 0 & 1.129 & 0.117 \end{bmatrix}$$

Al eliminar las componentes con valor cero en cada uno de los patrones de entrada trasladados, obtenemos un conjunto de patrones dimensionalmente menores.

$$x^{1"} = \begin{pmatrix} 0.66 \\ 0.43 \\ -0.13 \\ -0.03 \end{pmatrix} \quad x^{2"} = \begin{pmatrix} 0.63 \\ 0.40 \\ -0.23 \\ -0.03 \end{pmatrix} \quad x^{3"} = \begin{pmatrix} 0.52 \\ 0.29 \\ -0.33 \\ -0.03 \end{pmatrix} \quad x^{4"} = \begin{pmatrix} -0.09 \\ -0.08 \\ -0.43 \\ -0.03 \end{pmatrix}$$

$$y^{1"} = y^{2"} = y^{3"} = y^{4"} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$x^{5"} = \begin{pmatrix} -0.45 \\ 0.01 \\ 0.24 \\ 0.22 \end{pmatrix} \quad x^{6"} = \begin{pmatrix} -0.48 \\ 0.32 \\ 0.30 \\ -0.03 \end{pmatrix} \quad x^{7"} = \begin{pmatrix} -0.31 \\ -0.68 \\ 0.30 \\ -0.04 \end{pmatrix} \quad x^{8"} = \begin{pmatrix} -0.47 \\ -0.68 \\ 0.30 \\ -0.05 \end{pmatrix}$$

$$y^{5"} = y^{6"} = y^{7"} = y^{8"} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Al aplicar la regla de aprendizaje sobre el conjunto fundamental de patrones trasladados dimensionalmente menores, se obtienen las *p* matrices requeridas en la fase de aprendizaje; consecuentemente, el clasificador obtenido también es dimensionalmente menor.

$$C' = \begin{bmatrix} 1.718 & 1.032 & -1.129 & -0.117 \\ -1.718 & -1.032 & 1.129 & 0.117 \end{bmatrix}$$

Para comprobar que el conjunto fundamental de patrones trasladados y su representación dimensionalmente menor es equivalente para fines de clasificación, aplicaremos la fase de recuperación, sin enmascaramiento para el conjunto fundamental de patrones trasladados dimensionalmente menores.

$$y^{1"} = \left[C' \bullet x^{1"}\right] = \begin{bmatrix} 1.72 \\ -1.72 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{1"} = y^{1} \Rightarrow \quad Correcto$$

$$y^{2"} = \left[C' \bullet x^{2"}\right] = \begin{bmatrix} 1.75 \\ -1.75 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{2"} = y^{2} \Rightarrow \quad Correcto$$

$$y^{3"} = \left[C' \bullet x^{3"}\right] = \begin{bmatrix} 1.58 \\ -1.58 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{3"} = y^{3} \Rightarrow \quad Correcto$$

$$y^{4"} = \left[C' \bullet x^{4"}\right] = \begin{bmatrix} 0.25 \\ -0.25 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad ; \quad y^{4"} = y^4 \Rightarrow \quad Correcto$$

$$y^{5"} = \left[C' \bullet x^{5"}\right] = \begin{bmatrix} -1.06 \\ 1.06 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{5"} = y^5 \Rightarrow \quad Correcto$$

$$y^{6"} = \left[C' \bullet x^{6"}\right] = \begin{bmatrix} -0.82 \\ 0.82 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{6"} = y^6 \Rightarrow \quad Correcto$$

$$y^{7"} = \left[C' \bullet x^{7"}\right] = \begin{bmatrix} -1.57 \\ 1.57 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{7"} = y^7 \Rightarrow \quad Correcto$$

$$y^{8"} = \left[C' \bullet x^{8"}\right] = \begin{bmatrix} -1.85 \\ 1.85 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ; \quad y^{8"} = y^8 \Rightarrow \quad Correcto$$

Como se pudo observar, los resultados alcanzados con el conjunto fundamental de patrones trasladados y su representación dimensionalmente menor, son equivalentes.

Cabe mencionar que, la representación dimensionalmente menor, es 50% más pequeña, en otras palabras, al aplicar este enfoque para la reducción dimensional de los datos, en este caso particular, es posible eliminar la mitad de la información presente en cada uno de los patrones que forman el conjunto fundamental, sin afectar el índice de clasificación.

4.2 Implementación en Hardware de la Operación Alfa

Retomando algunos conceptos presentados en la sección 3.1, tenemos que: para la definición de las operaciones binarias α y β se deben especificar los conjuntos A y B, los cuales son:

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

Los cuales para fines de implementación en hardware se tienen que expresar de la siguiente forma:

$$A = \{0, 1\} \quad \text{y} \quad B = \{00, 01, 10\}$$

La operación binaria α: A x $A \rightarrow B$, es utilizada para formar las asociaciones entre patrones de entrada y de salida (Tabla 4.2.1).

Tabla 4.2.1. Operación binaria α: A x $A \rightarrow B$

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

Por definición, esta operación recibe dos bits como parámetros de entrada y entrega otros dos bits a la salida, siendo a_out(1) el bit mas significativo a la salida (Figura 4.2.i).

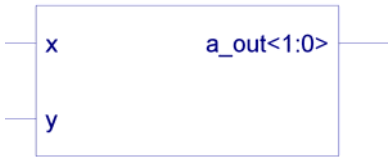


Figura 4.2.i *Bloque Alfa.*

La implementación de la operación binaria α, a nivel de compuertas lógicas se muestra en la Figura 4.2.ii.

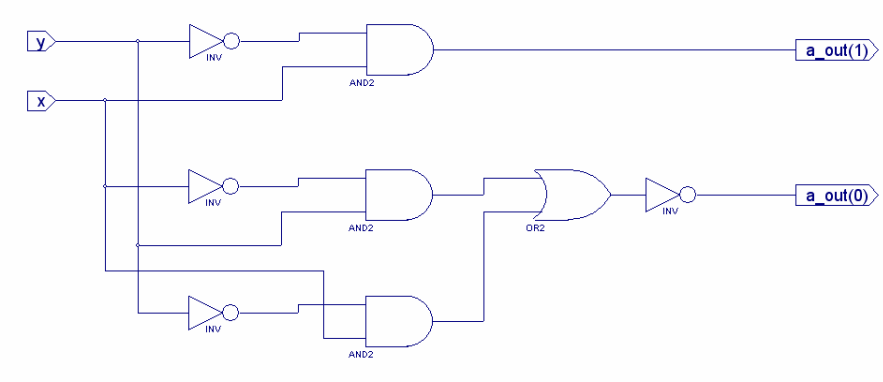


Figura 4.2.ii *Implementación de la Operación Alfa a nivel de compuertas lógicas.*

4.3 Implementación en Hardware de la Operación Beta

La operación binaria β: B x $A \rightarrow A$, es utilizada para recuperar los patrones de salida (Tabla 4.2.2).

Tabla 4.2.2. Operación binaria β: B x $A \rightarrow A$

x		y	$\beta(x, y)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1

Por definición, esta operación recibe tres bits como parámetros de entrada y entrega un solo bit a la salida, siendo x(1) el bit mas significativo a la entrada (Figura 4.3.i).

Figura 4.3.i *Bloque Beta.*

La implementación de la operación binaria β, a nivel de compuertas lógicas se muestra en la Figura 4.3.ii.

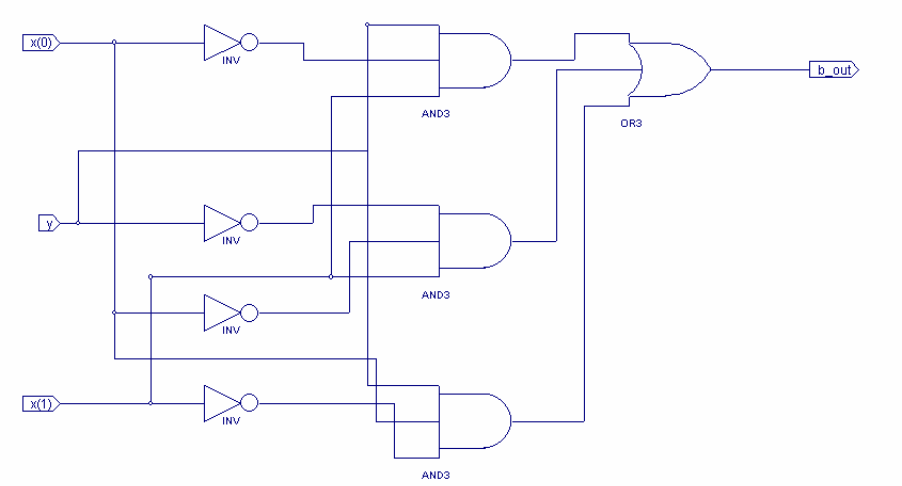


Figura 4.3.ii *Implementación de la Operación Beta a nivel de compuertas lógicas.*

4.4 Arquitectura Propuesta

El objetivo principal de toda memoria asociativa, es poder llevar a cabo tareas de aprendizaje y recuperación de patrones de manera eficaz, sin embargo, cuando la dimensionalidad de los datos es elevada, la fase de aprendizaje puede ser particularmente demandante. De ahí que, para lograr este tipo de tareas, es preciso contar con arquitecturas de cómputo especialmente diseñadas para ejecutar las operaciones necesarias de manera eficiente.

En esta sección se presentan por separado las dos fases, siempre presentes, en toda memoria asociativa: aprendizaje y recuperación.

4.4.1 Arquitectura de cómputo para la fase de aprendizaje

La arquitectura de cómputo propuesta para la fase de aprendizaje, se muestra en la Figura 4.4i. Consta de dos registros R_1 y R_2, los cuales almacenan un patrón de entrada y uno de salida respectivamente; es decir, R_1 contiene el patrón x^μ, mientras que R_2 contiene el patrón y^μ.

Un aspecto interesante de toda implementación en hardware, es la posibilidad de poder ejecutar operaciones en paralelo.

Particularmente, son dos aspectos fundamentales, los que hacen posible instanciar múltiples unidades Alfa funcionando en paralelo:

- El número de compuertas lógicas necesarias para la implementación en hardware de un bloque Alfa, es igual a ocho.

- El número de compuertas disponibles en un FPGA Spartan3 del fabricante Xilinx, puede ir desde 50 mil a 5 millones.

Los registros R_{3p} y R_{3m}, funcionan como argumentos de entrada del bloque de comparación MAX. El registro R_{3p} mantiene el dato proveniente de las unidades Alfa, mientras que el registro R_{3m} mantiene el dato proveniente de memoria. El registro R_{3f} funciona como recurso de almacenamiento del dato que será enviado a memoria.

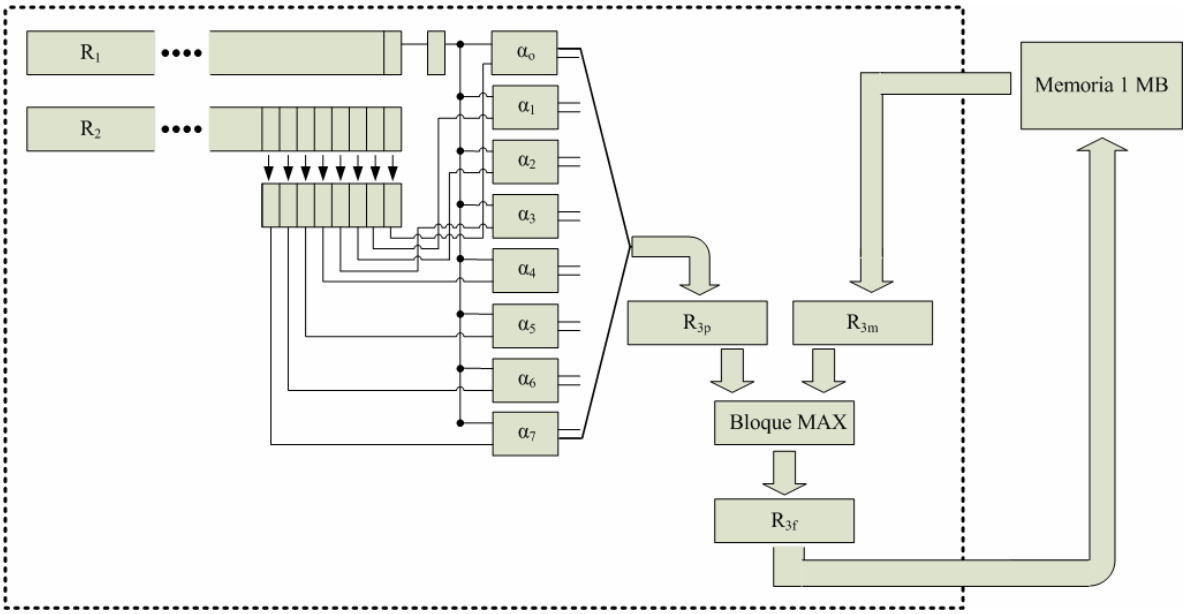


Figura 4.4.i *Arquitectura Propuesta para la Fase de Aprendizaje.*

La conclusión general es que, la arquitectura propuesta para la fase de aprendizaje, permite aprender *n* veces más rápido que el modelo propuesto en [20], siempre que sea posible instanciar *n* unidades Alfa operando en paralelo.

4.4.2 Arquitectura de cómputo para la fase de recuperación

La arquitectura de cómputo propuesta para la fase de recuperación, se muestra en la Figura 4.4ii. Consta de dos registros R_{3m} y R_1, los cuales almacenan una fila de la memoria asociativa y el patrón a recuperar respectivamente; es decir, R_1 contiene el patrón x^{μ}, mientras que R_{3m} contiene una línea de la memoria asociativa.

El registro R_3, funciona como recurso de almacenamiento para el argumento de entrada del bloque min. El registro R_2, almacena el patrón de salida recuperado y^{μ}.

Particularmente, son dos aspectos fundamentales, los que hacen posible instanciar múltiples unidades Beta funcionando en paralelo:

- El número de compuertas lógicas necesarias para la implementación en hardware de un bloque Beta, es igual a ocho.

- El número de compuertas disponibles en un FPGA Spartan3 del fabricante Xilinx, puede ir desde 50 mil a 5 millones.

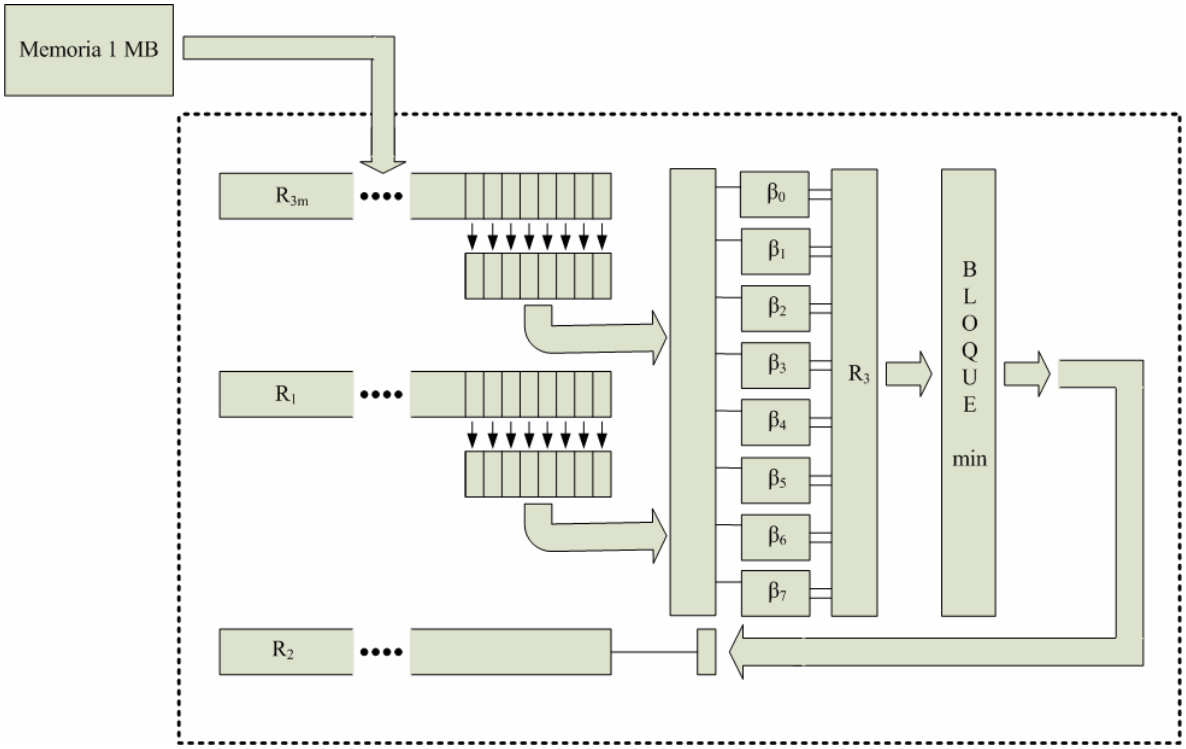


Figura 4.4.ii *Arquitectura Propuesta para la Fase de Recuperación.*

La conclusión general es que, la arquitectura propuesta para la fase de recuperación, permite recuperar *n* veces más rápido que el modelo propuesto en [20], siempre que sea posible instanciar *n* unidades Beta operando en paralelo.

CAPÍTULO 5

Resultados y Discusión

Este capítulo es de vital importancia en el presente trabajo de tesis, puesto que no solo se ilustran los conceptos descritos en los capítulos anteriores, también se expone la eficacia del enfoque propuesto para reducción dimensional de los datos; asimismo, se muestran los resultados de la implementación en hardware del modelo matemático de las memorias asociativas Alfa-Beta.

En la sección 5.1, se muestran tanto el índice de clasificación como el de optimización de rasgos, para cada una de las ocho bases de datos usadas durante la fase experimental.

En la sección 5.2, se presentan los resultados del desempeño de la arquitectura de cómputo propuesta; obtenidos mediante la herramienta Model*Sim* XE II/Starter 5.8c.

5.1 Aplicación en bases de datos

Para la realización de la fase experimental, fueron usadas ocho diferentes bases de datos; tomadas del repositorio de bases de datos de la Universidad de California en Irvine. Las principales características de cada una de ellas, se resumen en la Tabla 5.1.

Tabla 5.1. Características de los conjuntos de datos usados

	Iris	**Liver**	**Pima**	**Breast**	**CMC**	**Wine**	**Heart**	**Credit**
Número de Rasgos	4	6	8	9	9	13	13	14
Número de Clases	3	2	2	2	3	3	2	2
Número de Patrones	150	345	768	699	1473	178	270	690

Los experimentos se realizaron en una computadora personal (PC), con un procesador Intel Core2 Duo a 2.13 GHz, 1024 MBytes de memoria RAM y 73.2 GBytes de espacio en disco duro. Se usó el sistema operativo Windows XP Profesional de Microsoft y se utilizó la herramienta computacional HCM v1.0.

Cada uno 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 permite la obtención de un clasificador balanceado. Posteriormente, se aplicó el Algoritmo HCM [44], descrito en la sección 4.1, para conducir el proceso de selección de rasgos.

Para obtener una estimación confiable del índice de clasificación alcanzado por ambos conjuntos de datos (completo y optimizado), sobre instancias nunca antes vistas (conjunto de prueba), se aplicó una técnica de validación cruzada, a saber: *K*-Fold Cross Validation, con K=10.

Tabla 5.2. Resultados de la Selección de Rasgos

	Iris	Liver	Pima	Breast	CMC	Wine	Heart	Credit
Tamaño original del conjunto de datos	4	6	8	9	9	13	13	14
Rasgos seleccionados	2	2	2	4	5	7	4	2
Índice de reducción dimensional de los datos	50.0%	66.6%	75.0%	55.5%	44.4%	46.1%	69.2%	85.7%

Los resultados expuestos en la Tabla 5.2, muestran el índice de reducción dimensional de los datos para los ocho diferentes conjuntos de prueba; cabe mencionar que, el número de rasgos seleccionados para dos de las bases de datos (CMC, Wine) fue menor al 50%; sin embargo, para los demás conjuntos de prueba, el índice de reducción dimensional se mantuvo en el rango de 50% a 85%.

La representación gráfica de lo anteriormente dicho, se muestra en la Figura 5.1.i; ésta pone de manifiesto la eficacia del algoritmo HCM para obtener subconjuntos de datos (dimensionalmente menores) equivalentes para fines de clasificación.

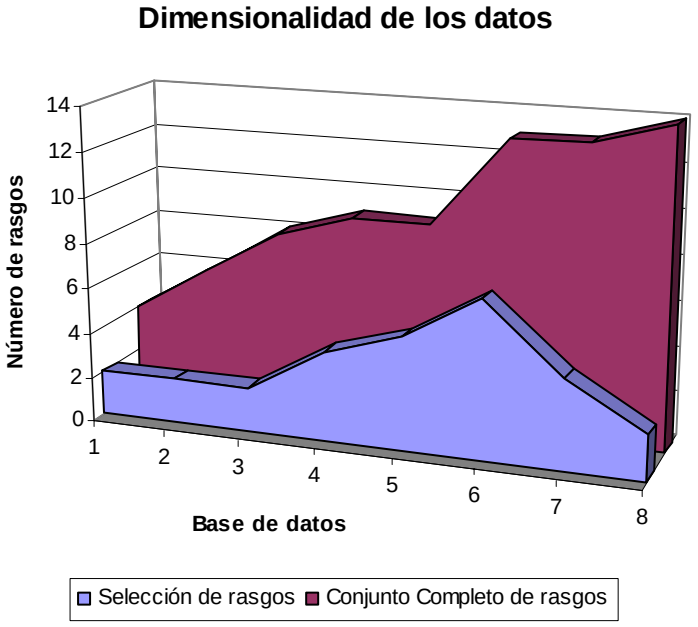


Figura 5.1.i *Comparación del número de rasgos utilizados para cada base de datos.*

Los resultados del índice de clasificación alcanzado por ambos conjuntos de rasgos (original y optimizado) se muestran en la Tabla 5.3 y en la Tabla 5.4; cabe mencionar que, en términos de este índice, para las primeras seis bases de datos, el desempeño de ambos conjuntos de rasgos es semejante.

Por otro lado, cuando las condiciones de las componentes de los patrones que conforman el conjunto original de rasgos permiten la eliminación de información irrelevante para fines de clasificación, el desempeño alcanzado por el conjunto optimizado de rasgos, es claramente superior.

Dicha situación puede observarse en dos de las ocho bases de datos empleadas (Heart y Credit), donde se alcanzan incrementos de 20% en el índice de clasificación.

Tabla 5.3. Índice de clasificación usando el conjunto original de rasgos

	Iris	Liver	Pima	Breast	CMC	Wine	Heart	Credit
Tamaño original del conjunto de datos	4	6	8	9	9	13	13	14
Índice de clasificación	95.6%	52.0%	62.0%	95.2%	55.2%	92.3%	63.0%	52.2%

Tabla 5.4. Índice de clasificación usando el conjunto optimizado de rasgos

	Iris	Liver	Pima	Breast	CMC	Wine	Heart	Credit
Rasgos seleccionados	2	2	2	4	5	7	4	2
Índice de clasificación	95.9%	54.0%	69.0%	97.0%	58.0%	95.2%	83.5%	85.5%

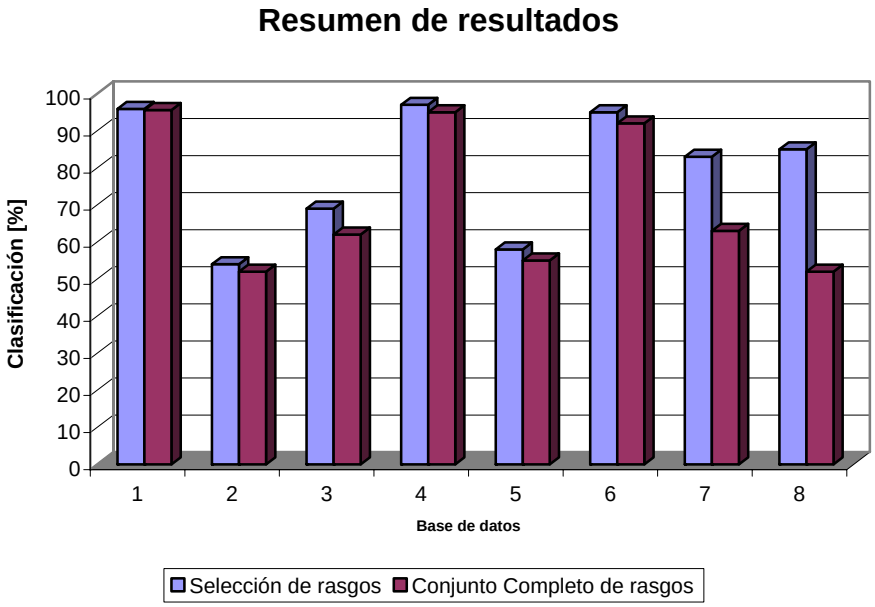


Figura 5.1.ii *Comparación del índice de clasificación para cada base de datos.*

La representación gráfica de lo anteriormente expresado, se muestra en la Figura 5.1.ii; ésta pone de manifiesto la eficacia del algoritmo HCM para eliminar información irrelevante para fines de clasificación; cabe mencionar que, al efectuarse dicha selección de rasgos, no solo se obtienen conjuntos de datos dimensionalmente menores, también se incrementa la precisión predictiva sobre instancias no conocidas.

En la Tabla 5.5 se muestran dos índices altamente relacionados que se traducen en importantes reducciones tanto en recursos de almacenamiento empleados, como en lógica sintetizada en un FPGA.

Tabla 5.5. Resultados de la Optimización en Hardware

	Iris	Liver	Pima	Breast	CMC	Wine	Heart	Credit
Tamaño original del conjunto de datos	4	6	8	9	9	13	13	14
Rasgos seleccionados	2	2	2	4	5	7	4	2
Índice de reducción dimensional de los datos	**50.0%**	**66.6%**	**75.0%**	**55.5%**	**44.4%**	**46.1%**	**69.2%**	**85.7%**
Tamaño original del registro en bits	50	105	153	100	130	252	224	360
Tamaño optimizado del registro en bits	30	45	51	50	78	144	80	72
Índice de optimización del tamaño del registro	**40%**	**57%**	**66%**	**50%**	**40%**	**42%**	**64%**	**80%**

Al aplicar este método para la reducción dimensional de los datos, previo al proceso de implementación de los registros y unidades funcionales requeridas para llevar a cabo tareas de aprendizaje y recuperación de patrones de manera eficiente, se puede observar que, es posible obtener arquitecturas de cómputo optimizadas (80% más pequeñas) sin demeritar el índice de clasificación. Asimismo, dado el índice de optimización de recursos alcanzado, es posible instanciar múltiples unidades funcionales operando en paralelo, lo cual reduce significativamente los tiempos de respuesta en ambas fases (aprendizaje y recuperación).

5.2 Desempeño de la Implementación en Hardware

Para mostrar el desempeño de la arquitectura de cómputo propuesta en la sección 4.4, se tiene el siguiente conjunto fundamental: $\left\{(x^{\mu}, y^{\mu}) \mid \mu = \ 1,2,\ldots,p\right\}$

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

$$y^1 = \begin{pmatrix}1\\1\\0\\1\end{pmatrix}, \quad y^2 = \begin{pmatrix}1\\0\\0\\1\end{pmatrix}, \quad y^3 = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}, \quad y^4 = \begin{pmatrix}1\\1\\0\\0\end{pmatrix}, \quad y^5 = \begin{pmatrix}1\\0\\1\\1\end{pmatrix}$$

Para obtener la memoria heteroasociativa Alfa-Beta MAX, tenemos:

$$y^1 \otimes \left(x^1\right)^t = \begin{pmatrix}1\\1\\0\\1\end{pmatrix} \otimes \begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 1 & 2 & 1\\1 & 1 & 2 & 1\\0 & 0 & 1 & 0\\1 & 1 & 2 & 1\end{pmatrix}$$

$$y^2 \otimes \left(x^2\right)^t = \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \otimes \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}2 & 2 & 2 & 1\\1 & 1 & 1 & 0\\1 & 1 & 1 & 0\\2 & 2 & 2 & 1\end{pmatrix}$$

$$y^3 \otimes \left(x^3\right)^t = \begin{pmatrix}1\\1\\1\\1\end{pmatrix} \otimes \begin{pmatrix}0 & 1 & 1 & 1\end{pmatrix} = \begin{pmatrix}2 & 1 & 1 & 1\\2 & 1 & 1 & 1\\2 & 1 & 1 & 1\\2 & 1 & 1 & 1\end{pmatrix}$$

$$y^4 \otimes \left(x^4\right)^t = \begin{pmatrix}1\\1\\0\\0\end{pmatrix} \otimes \begin{pmatrix}0 & 1 & 0 & 0\end{pmatrix} = \begin{pmatrix}2 & 1 & 2 & 2\\2 & 1 & 2 & 2\\1 & 0 & 1 & 1\\1 & 0 & 1 & 1\end{pmatrix}$$

$$y^5 \otimes \left(x^5\right)^t = \begin{pmatrix}1\\0\\1\\1\end{pmatrix} \otimes \begin{pmatrix}1 & 0 & 1 & 1\end{pmatrix} = \begin{pmatrix}1 & 2 & 1 & 1\\0 & 1 & 0 & 0\\1 & 2 & 1 & 1\\1 & 2 & 1 & 1\end{pmatrix}$$

Al aplicar el operador MAX ($\vee$), obtenemos la memoria heteroasociativa Alfa-Beta MAX:

$$\vee = \begin{pmatrix}2 & 2 & 2 & 2\\2 & 1 & 2 & 2\\2 & 2 & 1 & 1\\2 & 2 & 2 & 1\end{pmatrix}$$

Una vez obtenida la memoria heteroasociativa Alfa-Beta MAX, se da por concluida la fase de aprendizaje.

Para mostrar el desempeño de la fase de recuperación, presentaremos cada uno de los patrones que conforman el conjunto fundamental.

$$\vee \triangle_\beta x^1 = \begin{pmatrix}2 & 2 & 2 & 2\\2 & 1 & 2 & 2\\2 & 2 & 1 & 1\\2 & 2 & 2 & 1\end{pmatrix} \triangle_\beta \begin{pmatrix}1\\1\\0\\1\end{pmatrix} = \begin{pmatrix}1\\1\\0\\1\end{pmatrix} = y^1 \Rightarrow \quad Correcto$$

$$\vee \triangle_\beta x^2 = \begin{pmatrix}2 & 2 & 2 & 2\\2 & 1 & 2 & 2\\2 & 2 & 1 & 1\\2 & 2 & 2 & 1\end{pmatrix} \triangle_\beta \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}1\\0\\0\\1\end{pmatrix} = y^2 \Rightarrow \quad Correcto$$

$$\vee\triangle_\beta x^3 = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} \triangle_\beta \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = y^3 \Rightarrow \quad Correcto$$

$$\vee\triangle_\beta x^4 = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} \triangle_\beta \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = y^4 \Rightarrow \quad Correcto$$

$$\vee\triangle_\beta x^5 = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} \triangle_\beta \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = y^5 \Rightarrow \quad Correcto$$

Como se pudo observar, todo el conjunto fundamental fue correctamente recuperado; esto no debiera sorprender, ya que los teoremas en [20] aseguran la recuperación correcta de la información.

Toda vez que ha sido comprobado el comportamiento de las memorias asociativas Alfa-Beta, sobre un conjunto fundamental cualquiera, podemos analizar los resultados obtenidos mediante la arquitectura de cómputo propuesta en la sección 4.4.

En la figura 5.2i, después de 2.5μs, se observa que el dato en memoria es:

$$\begin{pmatrix} AA \\ 9A \\ A5 \\ A9 \end{pmatrix} \Rightarrow \begin{pmatrix} 10101010 \\ 10011010 \\ 10100101 \\ 10101001 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 1 \end{pmatrix} = \vee$$

Consecuentemente, se puede afirmar que, la fase de aprendizaje se llevó a cabo de manera exitosa.

Asimismo, se muestra que el tiempo requerido para aprender un bit es de 90ns; esto implica que, cuando se tengan patrones de entrada de dimensión *n* y siempre que sea posible instanciar *n* unidades funcionales trabajando en paralelo, el tiempo requerido para aprender un bit es conocido y constante.

Dicha afirmación no debiera sorprender, ya que, según el esquema mostrado en la Figura 4.4.i, el tiempo requerido para aprender *n* bits, depende completamente del número de unidades funcionales que puedan ser instanciadas.

Análogamente, para la fase de recuperación, se puede observar en la Figura 5.2.ii que el tiempo requerido para recuperar un patrón de *n* bits es de 280ns; esto implica que, cuando se tengan patrones de entrada de dimensión *n* y siempre que sea posible

instanciar *n* unidades funcionales trabajando en paralelo, el tiempo requerido para recuperar un patrón de *n* bits es conocido y constante.

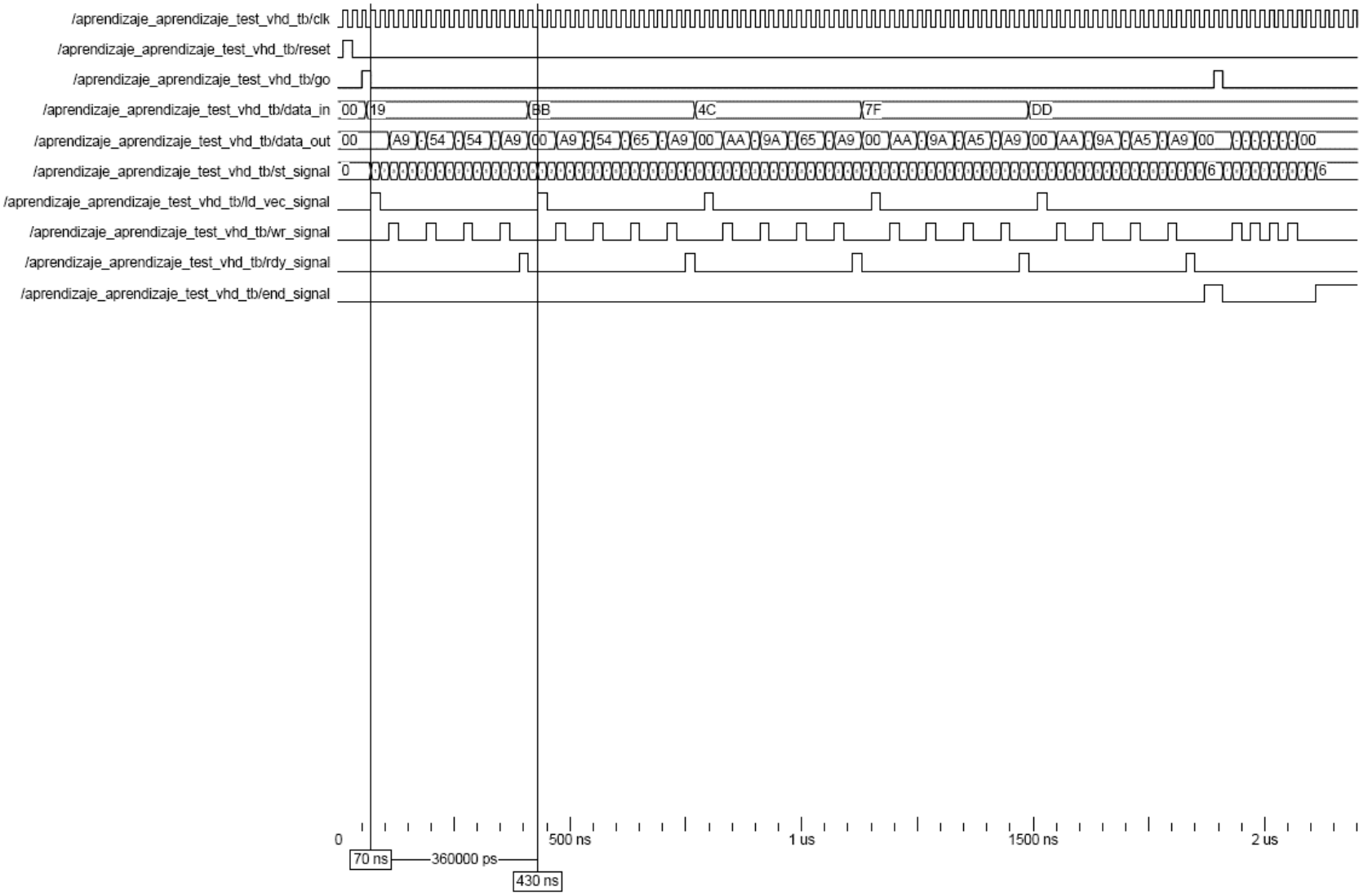


Figura 5.2.i *Ejecución de la fase de aprendizaje.*

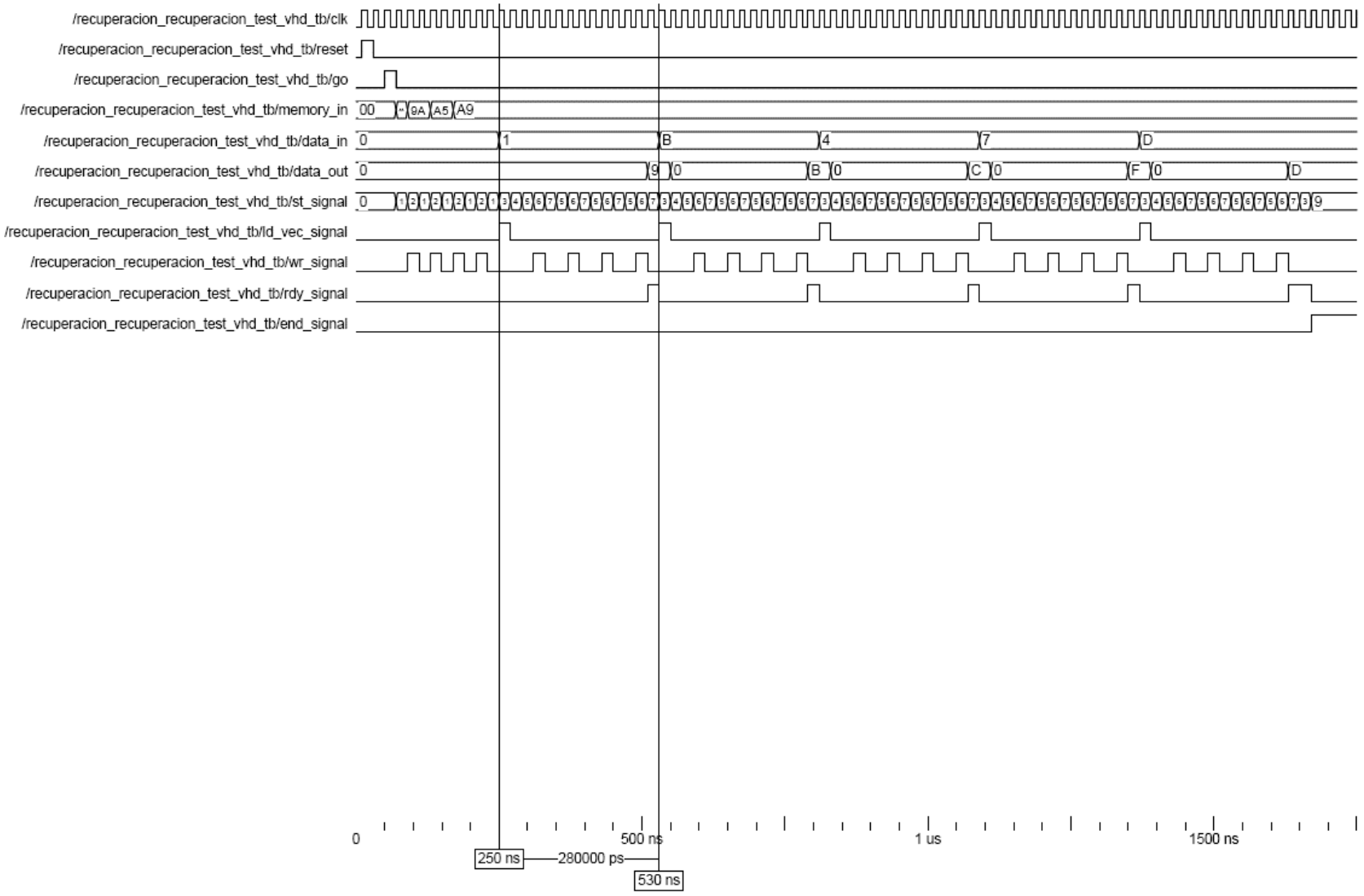


Figura 5.2.ii *Ejecución de la fase de recuperación.*

CAPÍTULO 6

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones derivadas de los resultados obtenidos durante la realización del presente trabajo de tesis; asimismo, se proponen algunos temas relacionados que pudieran ser afrontados con trabajos futuros de investigación.

6.1 Conclusiones

Durante la realización del presente trabajo de tesis, se observó que el rendimiento en la mayoría de los clasificadores, puede incrementarse cuando se elimina la información irrelevante del conjunto fundamental, sin embargo, la mayoría de los métodos de reducción dimensional, son altamente demandantes o requieren un elevado número de operaciones para alcanzar su cometido.

El algoritmo HCM para reducción dimensional de los datos, se centra en la obtención de una máscara que permite tanto la identificación como la eliminación de información irrelevante para fines de clasificación; cabe mencionar que, este enfoque no requiere construir clasificadores sucesivos para incrementar el índice de clasificación.

Un hecho que debe ser considerado seriamente es que, existe una relación directa entre la dimensionalidad de los datos y el tamaño de las unidades funcionales requeridas para su procesamiento, consecuentemente, un conjunto fundamental de patrones dimensionalmente menores, permite importantes reducciones en la cantidad de lógica sintetizable, así como de recursos de almacenamiento.

Basado en los resultados alcanzados durante la fase experimental, se puede afirmar que, la aplicación del algoritmo HCM para la reducción dimensional de los datos, es una alternativa eficaz para obtener arquitecturas de cómputo optimizadas, capaces de llevar a cabo tareas de aprendizaje y recuperación de patrones, de manera eficiente.

Otro aspecto igualmente importante es que, al aplicar esquemas de unidades funcionales operando en paralelo, es posible estimar la duración total de la fase de aprendizaje, aun en presencia de datos altamente dimensionales, ya que al instanciar múltiples unidades funcionales operando en paralelo, la tasa de aprendizaje es conocida y constante.

6.2 Trabajo futuro

1. Tomar como base el trabajo de investigación de M.E. Acevedo-Mosqueda, *et al.* [22], para desarrollar una arquitectura de cómputo que aprenda y recupere patrones de manera bidireccional.

2. Tomar como base el trabajo de investigación de T. Godhavari, *et al.* [47] sobre la aplicación de las redes neuronales en criptografía, para desarrollar ASICs dedicados a la encriptación de información.

3. Tomar como base el trabajo de investigación de E. Guzmán, *et al.* [57] sobre la aplicación de las memorias asociativas en compresión de imágenes, para desarrollar un sistema dedicado para compresión de imágenes.

4. Tomar como base el trabajo de investigación de C. Yáñez-Márquez, *et al.* [58] sobre la aplicación de las memorias asociativas en igualación industrial de colores, para desarrollar sistemas de igualación de colores, usando dispositivos basados en lógica reconfigurable.

5. Tomar como base el trabajo de investigación de B. Svensson, *et al.* [59] sobre la ejecución de algoritmos en redes neuronales implementadas con arreglos de procesadores seriales, para desarrollar arquitecturas especiales de computo basadas en los operadores Alfa y Beta.

Referencias

[1] Hassoun, M. H. (1993). *Associative Neural Memories: Theory and Implementation*. New York: Oxford University Press.

[2] Kohonen, T. (1989). *Self-Organization and Associative Memory*. (3rd edition ed.) Springer-Verlag New York, Inc.

[3] McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115-133.

[4] Ritter , G. X. & Sussner, P. (1996). An Introduction to Morphological Neural Networks. In *Proceedings of the 13th International Conference on Pattern Recognition* (pp. 709-717). IEEE Computer Society.

[5] Rosenblatt, F. (1958). The Perceptron: A Probabilistic model for Information Storage and Organization in the Brain. *Psychological Review, 65*, 386-408.

[6] Widrow, B. & Lehr, M. A. (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proceedings of the IEEE, 78*, 1415-1442.

[7] Werbos, P. J. (1990). Backpropagation Through Time: What it does and how to do it. *Proceedings of the IEEE, 78*, 1550-1560.

[8] Hopfield , J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences, 79*, 2554-2558.

[9] Minsky , M. L. & Papert, S. A. (1969). *Perceptrons*. MIT Press.

[10] Steinbuch, K. (1961). Die Lernmatrix. *Biological Cybernetics, 1*, 36-45.

[11] Willshaw, D. J., Buneman, O. P., & Longuet-Higgins, H. C. (1969). Non-holographic associative memory. *Nature, 222*, 960-962.

[12] Anderson , J. A. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences, 14*, 197-220.

[13] Kohonen, T. (1972). Correlation Matrix Memories. *IEEE Transactions on Computers, C-21*, 353-359.

[14] Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, Man and Cybernetics, 18*, 49-60.

[15] Haines, K. & Hecht-Nielsen, R. (1988). A BAM with increased information storage capacity. In *IEEE International Conference on Neural Networks* (pp. 181-190).

[16] Leung, C. S. & Cheung, K. F. (1991). Householder encoding for discrete bidirectional associative memory. In *IEEE International Joint Conference on Neural Networks* (pp. 237-241).

[17] Wang, Y.-F.; Cruz, J.B., Jr.; Mulligan, J.H., Jr. (1991). Guaranteed recall of all training pairs for bidirectional associative memory. *IEEE Transactions on Neural Networks, 2*, 559-567.

[18] Shen, D. & Cruz, J.B., Jr. (2005). Encoding strategy for maximum noise tolerance bidirectional associative memory. *IEEE Transactions on Neural Networks, 16*, 293-300.

[19] Ritter , G. X., Sussner, P., & Diaz de Leon, J. L. (1998). Morphological associative memories. *IEEE Transactions on Neural Networks, 9*, 281-293.

[20] Yáñez-Márquez, C. (2002). *Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios*. Tesis de Doctorado en Ciencias de la Computación, Centro de Investigación en Computación, México.

[21] Acevedo-Mosqueda, M.E. (2006). *Memorias Asociativas Bidireccionales Alfa-Beta*. Tesis de Doctorado en Ciencias de la Computación, Centro de Investigación en Computación, México.

[22] Acevedo-Mosqueda, M. E., Yáñez-Márquez, C., & López-Yáñez, I. (2006). A New Model of BAM: Alpha-Beta Bidirectional Associative Memories. *Lecture Notes in Computer Science, 4263*, 286-295.

[23] Acevedo-Mosqueda, M. E., Yáñez-Márquez, C., & López-Yáñez, I. (2006). Alpha-Beta Bidirectional Associative Memories Based Translator. *International Journal of Computer Science and Network Security, 6*, 190-194.

[24] Yáñez-Márquez, C., Sánchez-Fernández, L. P., & López-Yáñez, I. (2006). Alpha-Beta Associative Memories for Gray Level Patterns. *Lecture Notes in Computer Science, 3971*, 818-823.

[25] Yáñez Márquez, C., Díaz de León Santiago, J.L. & Salgado Ramírez, J.C. (2005), Applying the New V-Alpha-Beta Associative Memories to Gray Level Images, IT-209, Serie Azul, ISBN 970-36-0282-7, CIC-IPN, México.

[26] Yáñez Márquez, C., Díaz de León Santiago, J.L. & Salgado Ramírez, J.C. (2005). New V-Alpha-Beta Associative Memories able to Learn and Recall Patterns with Integer Components, IT-210, Serie Azul, ISBN 970-36-0283-5, CIC-IPN, México.

[27] Sossa, H., Barrón, R., & Vázquez, R. (2004). New Associative Memories to Recall Real-Valued Patterns. *Lecture Notes in Computer Science, 3287*, 195-202.

[28] Simpson, P. K. (1990). *Artificial Neural Systems*. Pergamon Press.

[29] Steinbuch, K. & Frank, H. (1961). Nichtdigitale lernmatrizen als perzeptoren. *Biological Cybernetics, 1*, 117-124.

[30] Suresh, D. C., Agrawal, B., Yang, J., Najjar, W. A., & Bhuyan, L. N. (2003). Power efficient encoding techniques for off-chip data buses. In *Proceedings of International Conference on Compilers, Architecture and Synthesis for Embedded Systems* (pp. 267-275). San Jose, California, USA: ACM Press.

[31] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001), Lernmatrix de Steinbuch, IT-48, Serie Verde, ISBN 970-18-6688-6, CIC-IPN, México

[32] Abu-Mostafa, Y. S. & St Jacques, J.-M. (1985). Information capacity of the Hopfield model. *IEEE Transactions on Information Theory, 31*, 461-464.

[33] Austin, J. & Stonham, T. J. (1987). Distributed associative memory for use in scene analysis. *Image and Vision Computing, 5*, 251-260.

[34] Yáñez Márquez, C. & Díaz de León Santiago, J.L. (2001), Memorias Morfológicas Autoasociativas, IT-58, Serie Verde, ISBN 970-18-6698-3, CIC-IPN, México.

[35] Díaz de León Santiago, J.L. & Yáñez Márquez, C. (2001), Memorias Morfológicas Heteroasociativas, IT-57, Serie Verde, ISBN 970-18-6697-5, CIC-IPN, México.

[36] Anderson, J. A. & Rosenfeld, E. (1988). *Neurocomputing: foundations of research*. MIT Press.

[37] Rosen, K. H. (1996). *Discrete Mathematics and Its Applications*. McGraw-Hill, Inc.

[38] Minsky, M. L. (1954). *Neural Nets and the Brain Model problem*. Ph. D. dissertation, Princeton University.

[39] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*, 386-408.

[40] Steinbuch, K. (1965). *Automat und Mensch; Kybernetische Tatsachen und Hypothesen*. Berlin: Springer.

[41] Fukushima, K. (1975). Cognitron: a self-organizing multilayered neural network. *Biological Cybernetics, 20*, 121-136.

[42] Zurada, J. (1992). *Introduction to Artificial Neural Systems*. St. Paul, MN: West Publishing Company.

[43] Hecht-Nielsen, R. (1990). *Neurocomputing*. Reading, MA: Addison-Wesley.

[44] Aldape-Pérez, M., Yáñez-Márquez, C., & López-Leyva, L. O. (2006). Feature Selection Using a Hybrid Associative Classifier with Masking Techniques. In *Fifth Mexican International Conference on Artificial Intelligence* (pp. 151-160). IEEE Computer Society.

[45] Aldape-Pérez, M., Yáñez-Márquez, C., & Argüelles-Cruz, A. J. (2007). Optimized Associative Memories for Feature Selection. *Lecture Notes in Computer Science, 4477*, 435-442.

[46] Yáñez-Márquez, C., Felipe-Riverón, E. M., López-Yáñez, I., & Flores-Carapia, R. (2006). A Novel Approach to Automatic Color Matching. *Lecture Notes in Computer Science, 4225*, 529-538.

[47] Godhavari, T., Alamelu, N. R., & Soundararajan, R. (2005). Cryptography Using Neural Network. In *IEEE Annual India Conference* (pp. 258-261).

[48] Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157-1182.

[49] Blum, A. & Langley, P. (1997). Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence, 97*, 245-271.

[50] Kohavi, R. & John, G. H. (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence, 97*, 273-324.

[51] Last, M., Kandel, A., & Maimon, O. (2001). Information-theoretic algorithm for feature selection. *Pattern Recognition Letters, 22*, 799-811.

[52] Kira, K. & Rendell, L.A. (1992). The feature selection problem: traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 129-134). MIT Press.

[53] Almuallim, H. & Dietterich, T. G. (1994). Learning Boolean Concepts in the Presence of Many Irrelevant Features. *Artificial Intelligence, 69*, 279-305.

[54] Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth , M. K. (1987). Occam's razor. *Information Processing Letters, 24*, 377-380.

[55] Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 1995 International Joint Conference on AI* (pp. 1137-1145).

[56] Reunanen, J. (2003). Overfitting in Making Comparisons Between Variable Selection Methods. *Journal of Machine Learning Research, 3*, 1371-1382.

[57] Guzmán, E., Pogrebnyak, O., Yáñez-Márquez, C., & Moreno, J. A. (2006). Image Compression Algorithm Based on Morphological Associative Memories. *Lecture Notes in Computer Science, 4225*, 519-528.

[58] Yáñez-Márquez, C., Felipe-Riverón, E.M., López-Yáñez, I. & Flores-Carapia, R. (2006). A Novel Approach to Automatic Color Matching. *Lecture Notes in Computer Science, 4225*, 529-538.

[59] Svensson, B. & Nordström, T. (1990). Execution of neural network algorithms on an array of bit-serial processors. In *Proceedings of the 10th International Conference on Pattern Recognition, Computer Architectures for Vision and Pattern Recognition* (pp. 501-505).

[60] Piramuthu, S. (1998). Evaluating Feature Selection Methods for Learning in Data Mining Applications. In *Proceedings of The Thirty-First Annual Hawaii International Conference on System Sciences* (pp. 294-301).

[61] Jiang, L. & Zhang, H. (2006). Learning Naive Bayes for Probability Estimation by Feature Selection. *Lecture Notes in Computer Science*, *4013*, 503-514.

[62] Law, M.H.C., Figueiredo, M.A.T., Jain, A.K. (2004). Simultaneous Feature Selection and Clustering Using Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9)*, 1154-1166.

[63] Last, M., Kandel, A. & Maimon, O. (2001). Information-theoretic algorithm for feature selection. *Pattern Recognition Letters, 22(6-7)*, 799-811.

Apéndice A

Simbología

$\mathbf{M}$	memoria asociativa
$\mathbf{x}$	vector columna que corresponde a un patrón de entrada
$\mathbf{y}$	vector columna que corresponde a un patrón de salida
$(\mathbf{x}, \mathbf{y})$	asociación de un patrón de entrada con uno de salida
$(\mathbf{x}^k, \mathbf{y}^k)$	asociación de la k-ésima pareja de patrones
$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\}$	conjunto fundamental
A	conjunto al que pertenecen los vectores $\mathbf{x}$ y $\mathbf{y}$
B	conjunto al que pertenecen las entradas de la matriz $\mathbf{M}$
p	número de parejas del conjunto fundamental
n	dimensión de los patrones de entrada
m	dimensión de los patrones de salida
$\forall$	cuantificador universal
$\in$	pertenencia de un elemento a un conjunto
$\exists$	cuantificador existencial
y_k^μ	k-ésima componente del vector columna $\mathbf{y}^\mu$
$\times$	producto cruz (entre conjuntos)
m_{ij}	ij-ésima componente de la matriz $\mathbf{M}$
Δm_{ij}	incremento en m_{ij}
$\cdot$	producto usual entre vectores o matrices
$\vee$	operador máximo
$(\mathbf{x}^\mu)^t$	Transpuesto del vector $\mathbf{x}^\mu$
δ_{ij}	delta de Kronecker (afecta a los índices i y j)
$\mathbf{I}$	matriz identidad
$\tilde{\mathbf{x}}$	versión alterada del patrón fundamental $\mathbf{x}$
$\mathbf{W}$	memoria asociativa morfológica *min*
∇	producto máximo
Δ	producto mínimo
$\wedge$	operador mínimo
α y β	operadores en que se basan las memorias Alfa-Beta
$\alpha(x, y)$	operación binaria α con argumentos x y y
$\beta(x, y)$	operación binaria β con argumentos x y y
A y B	operadores en que se basan las memorias Media
med	operador media
∇_α	operador α *max*
∇_β	operador β *max*
Δ_α	operador α *min*
Δ_β	operador β *min*
$\otimes$	símbolo que representa a las dos operaciones ∇_α y Δ_α
$\mathbf{V}$	memorias asociativas Alfa-Beta tipo *max*
$\mathbf{\Lambda}$	memorias asociativas Alfa-Beta tipo *min*
$\mathbf{Z}^+$	conjunto de los números enteros positivos
$\mathbf{h}^k$	k-ésimo vector *one-hot*

Apéndice B

Resultados obtenidos usando el paquete computacional HCM v1.0

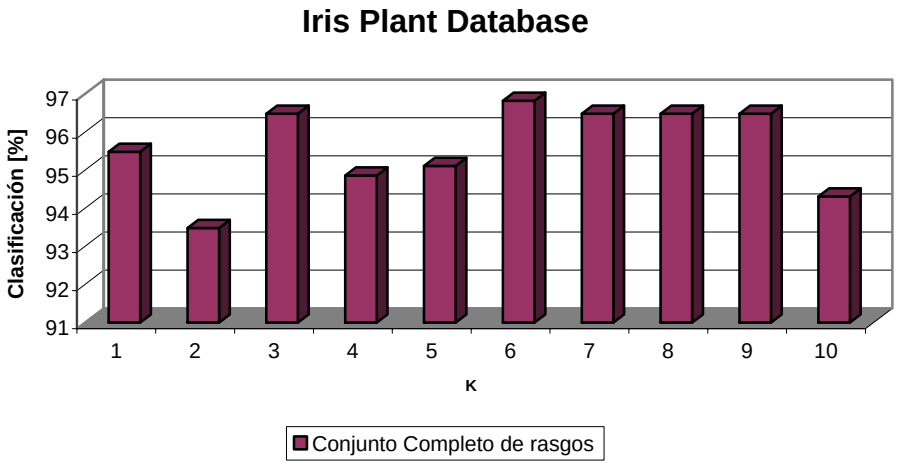


Figura Ap.B.i *Índice de clasificación usando el conjunto completo de rasgos.*

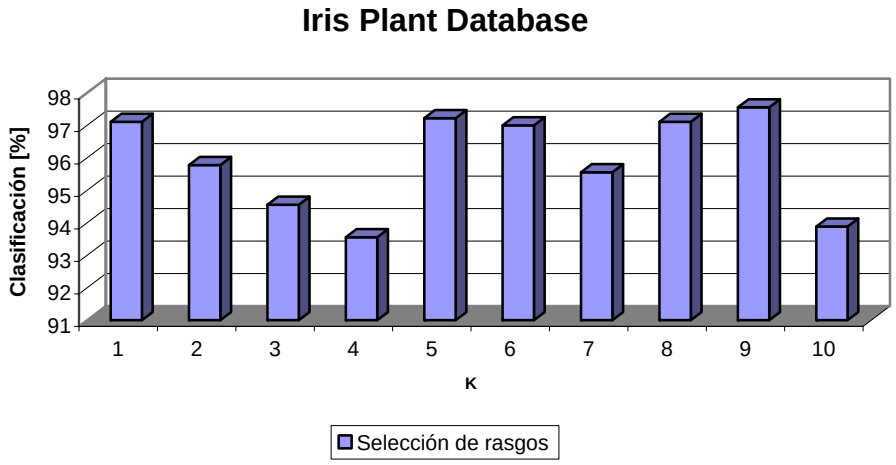


Figura Ap.B.ii *Índice de clasificación usando el conjunto optimizado de rasgos.*

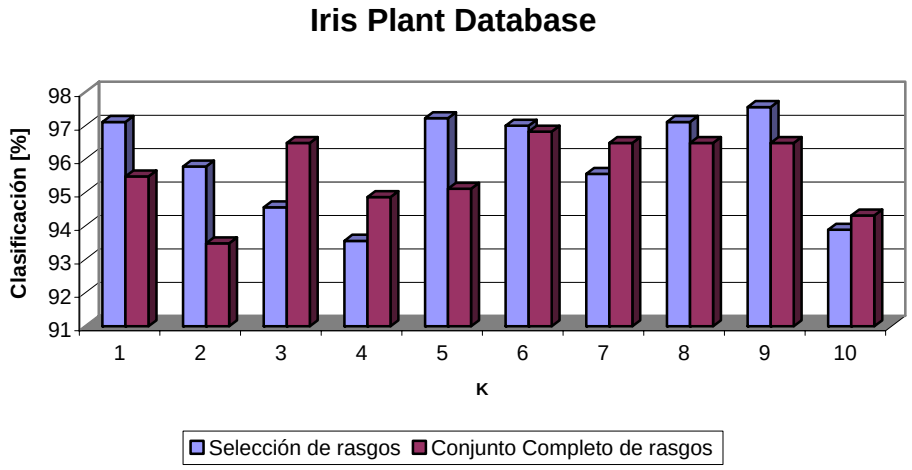


Figura Ap.B.iii *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

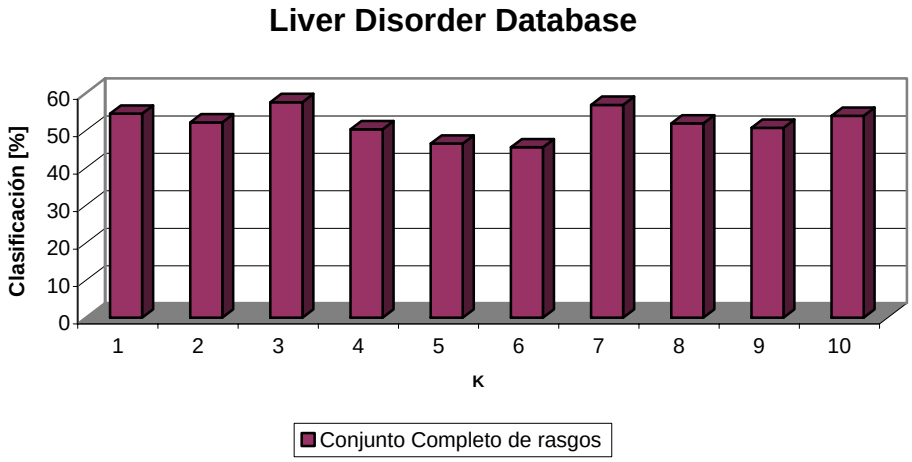


Figura Ap.B.iv *Índice de clasificación usando el conjunto completo de rasgos.*

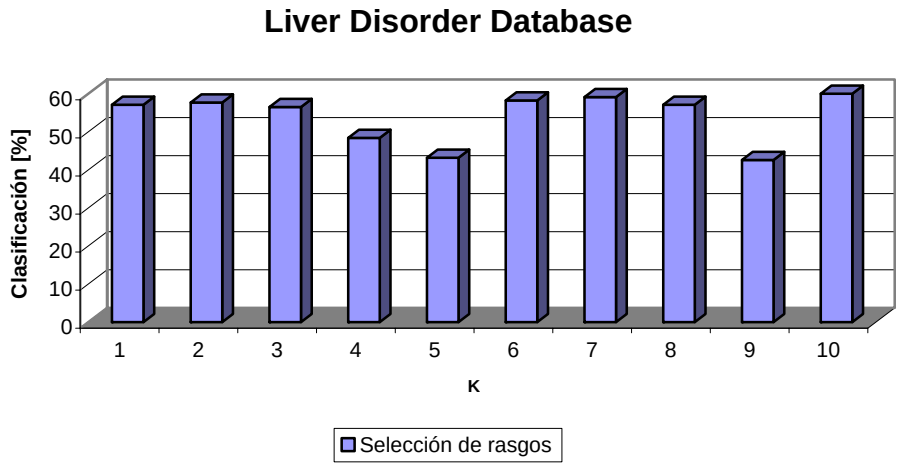


Figura Ap.B.v *Índice de clasificación usando el conjunto optimizado de rasgos.*

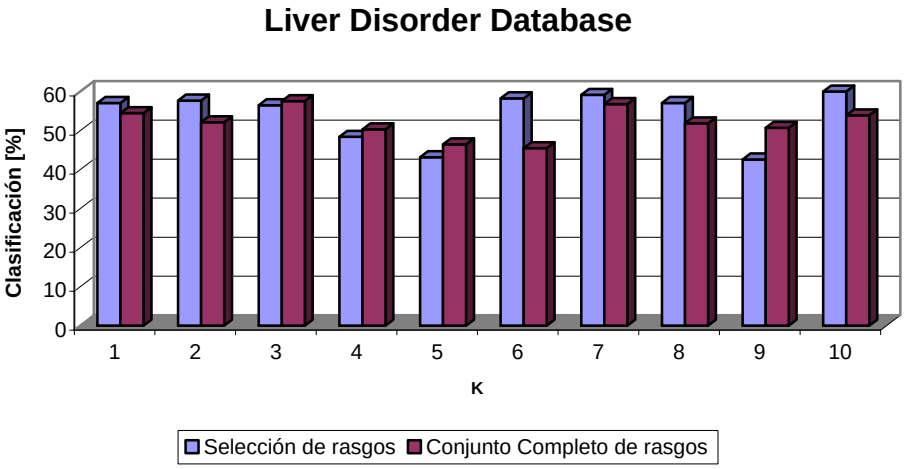


Figura Ap.B.vi *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

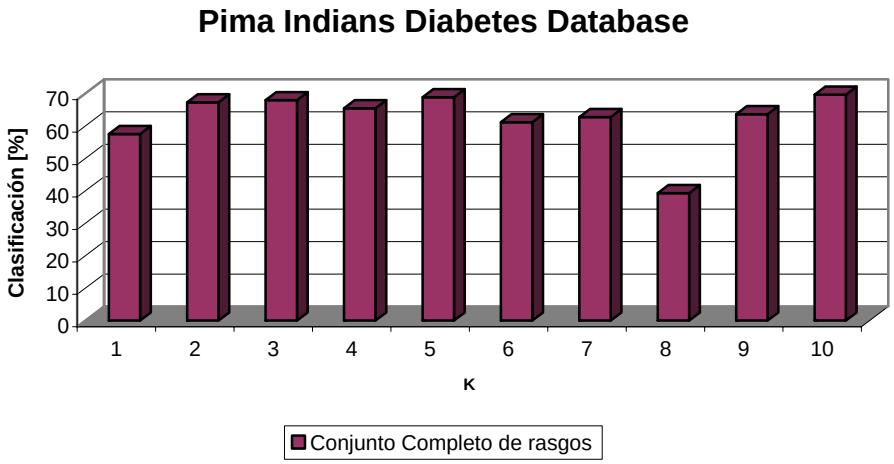


Figura Ap.B.vii *Índice de clasificación usando el conjunto completo de rasgos.*

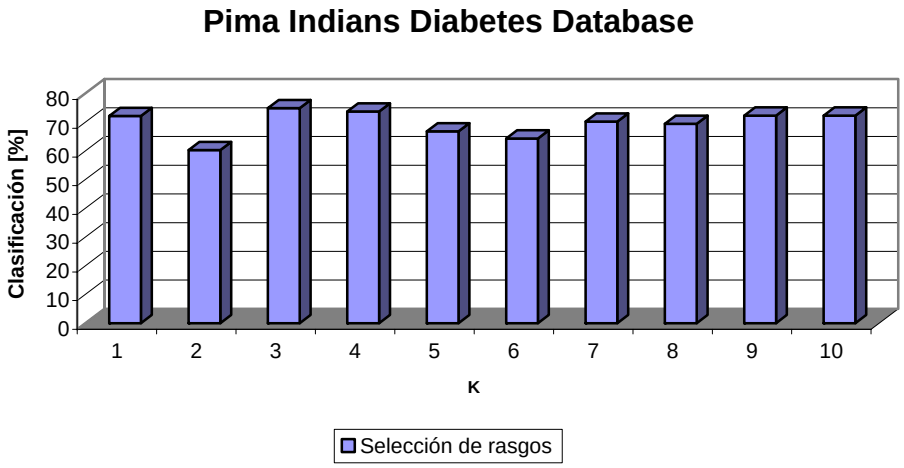


Figura Ap.B.viii *Índice de clasificación usando el conjunto optimizado de rasgos.*

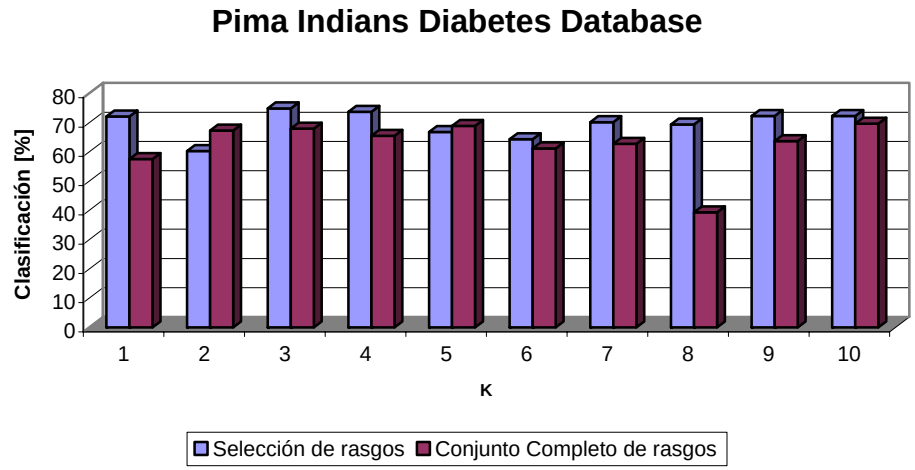


Figura Ap.B.ix *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

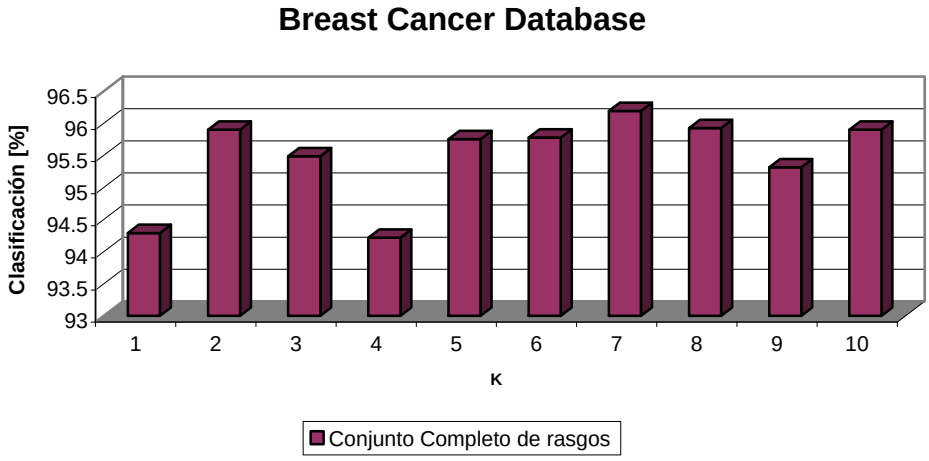


Figura Ap.B.x *Índice de clasificación usando el conjunto completo de rasgos.*

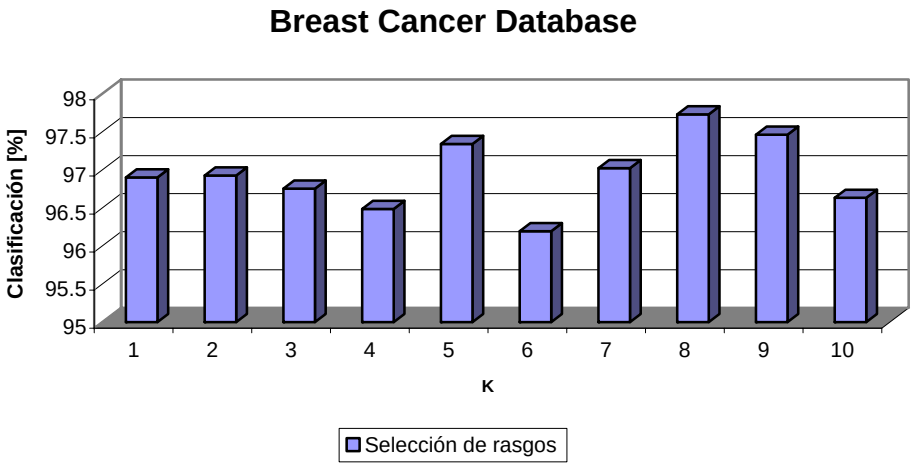


Figura Ap.B.xi *Índice de clasificación usando el conjunto optimizado de rasgos.*

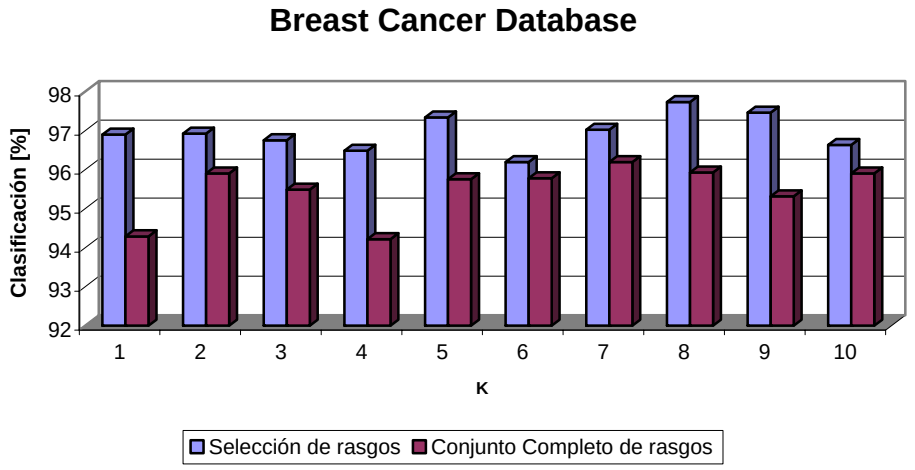


Figura Ap.B.xii *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

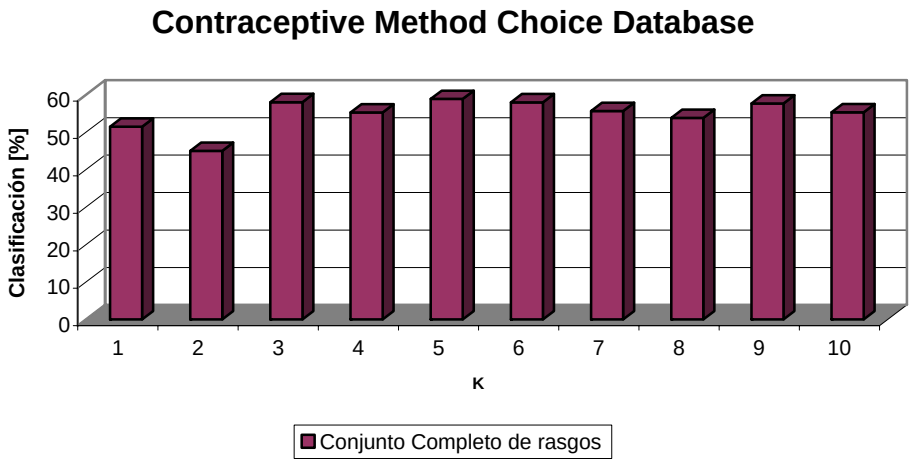


Figura Ap.B.xiii *Índice de clasificación usando el conjunto completo de rasgos.*

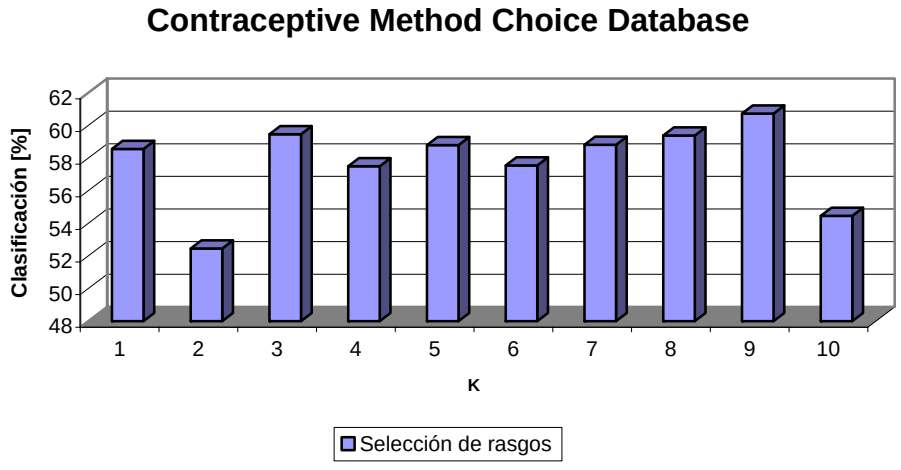


Figura Ap.B.xiv *Índice de clasificación usando el conjunto optimizado de rasgos.*

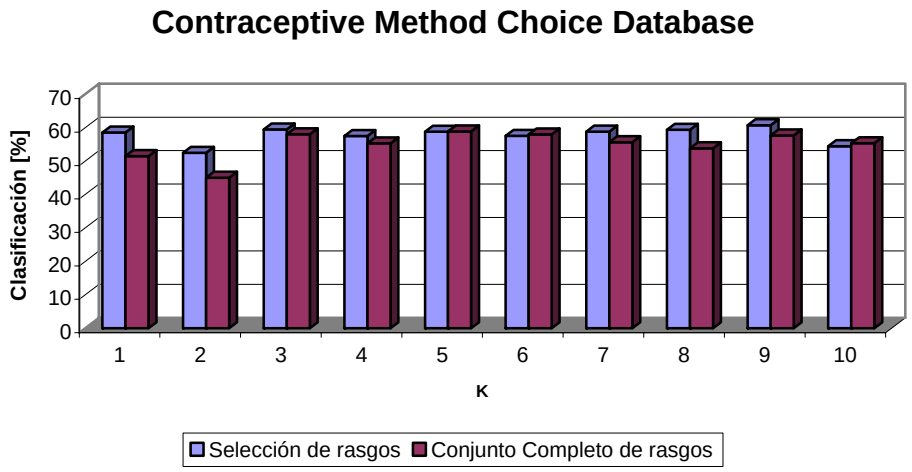


Figura Ap.B.xv *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

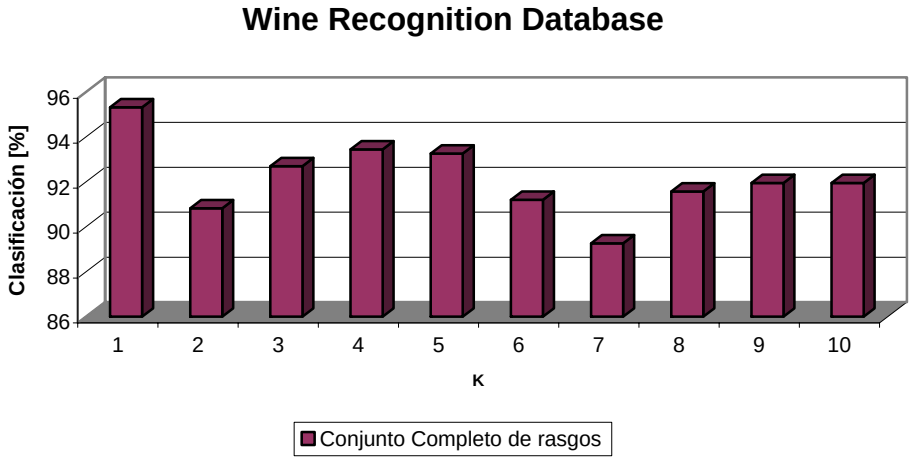


Figura Ap.B.xvi *Índice de clasificación usando el conjunto completo de rasgos.*

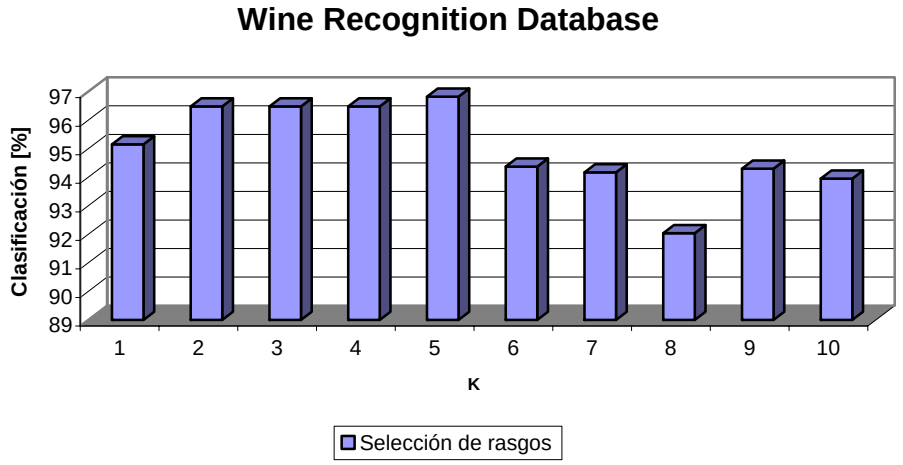


Figura Ap.B.xvii *Índice de clasificación usando el conjunto optimizado de rasgos.*

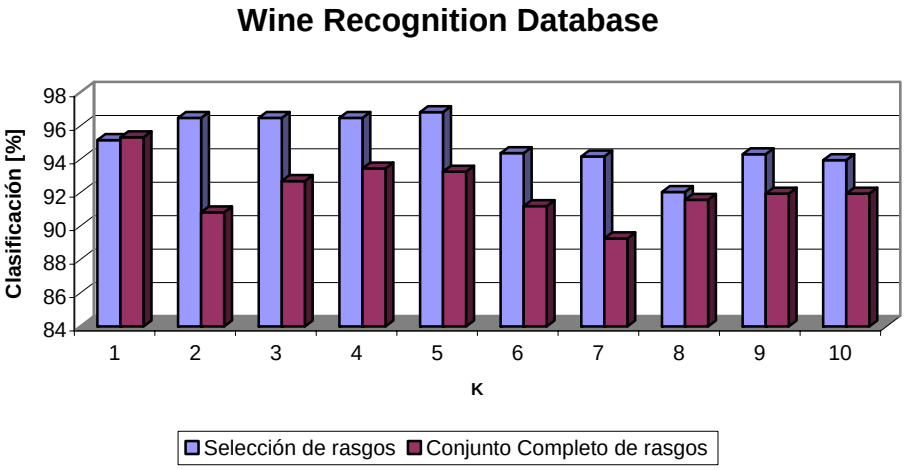


Figura Ap.B.xviii *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

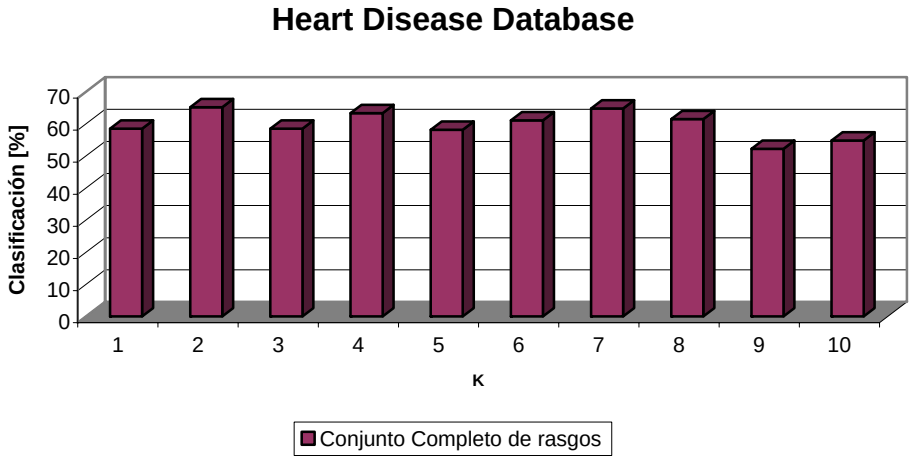


Figura Ap.B.xix *Índice de clasificación usando el conjunto completo de rasgos.*

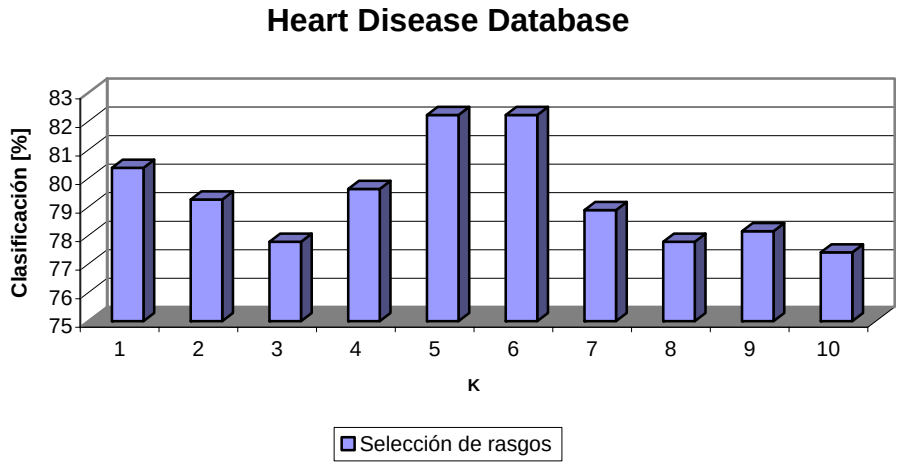


Figura Ap.B.xx *Índice de clasificación usando el conjunto optimizado de rasgos.*

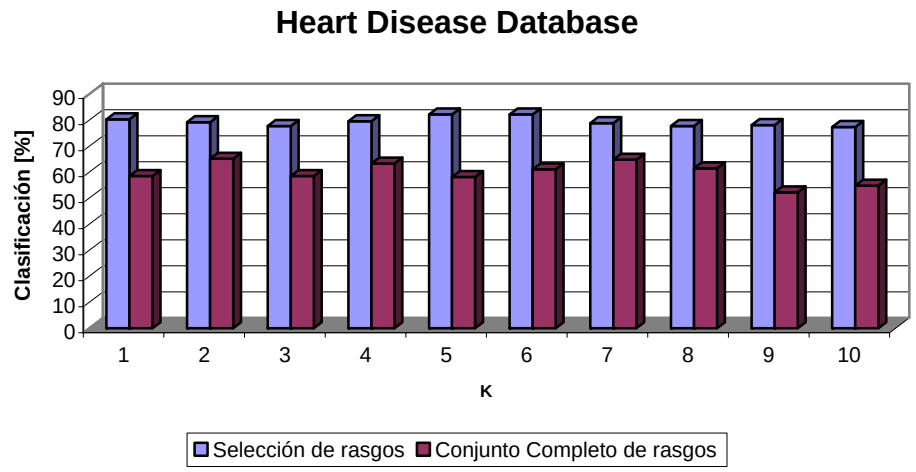


Figura Ap.B.xxi *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

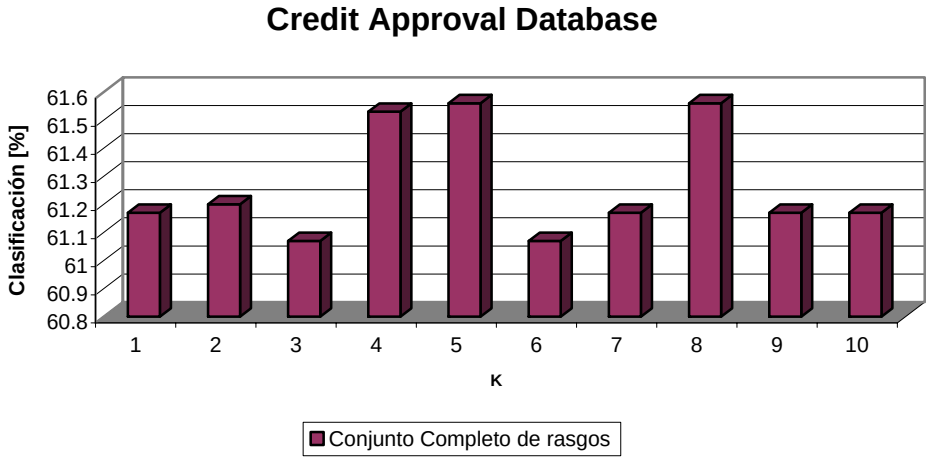


Figura Ap.B.xxii *Índice de clasificación usando el conjunto completo de rasgos.*

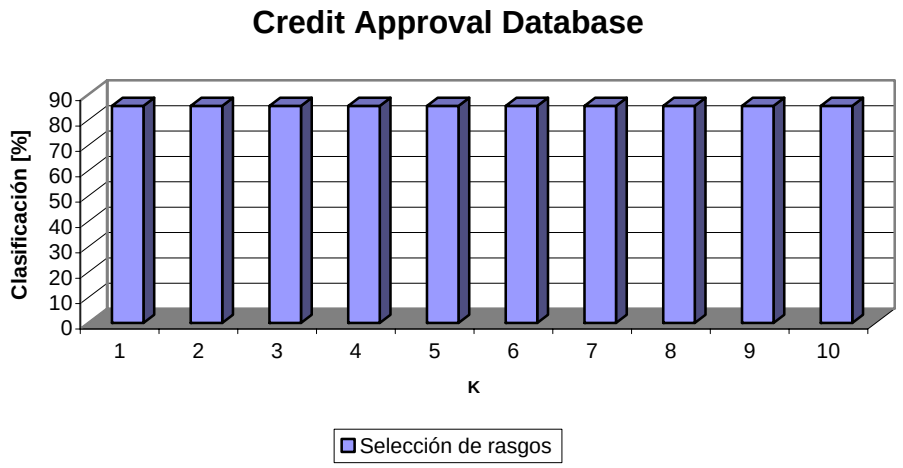


Figura Ap.B.xxiii *Índice de clasificación usando el conjunto optimizado de rasgos.*

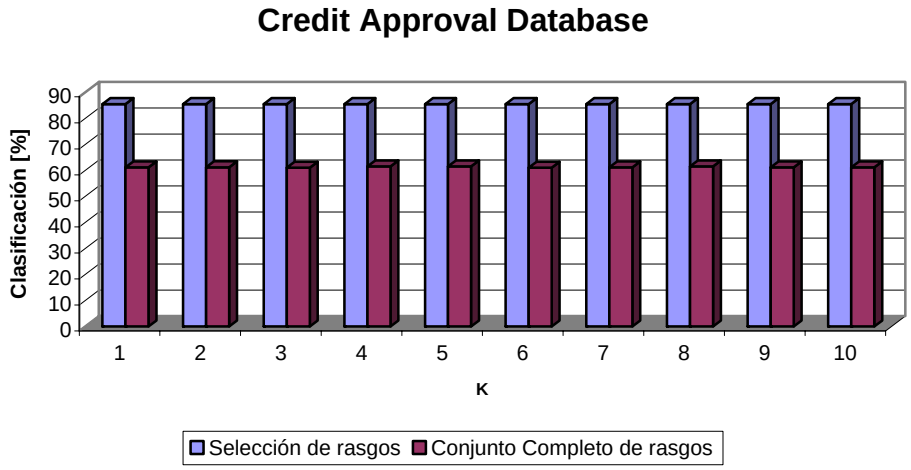


Figura Ap.B.xxiv *Comparación de los índices de clasificación para ambos conjuntos de rasgos.*

$\bar{\mathbf{h}}^k$	*k*-ésimo vector *zero-hot*
τ^e	transformada vectorial de expansión
τ^c	transformada vectorial de contracción
$\bar{s}$	Vector negado del vector **s**
LAy	*Linear Associator* modificado para **y**
$k(r)$	operador de cadena binaria mínima
$\mathcal{E}(r,k)$	operador k-binario de expansión
$\mathcal{E}_k^{-1}(\mathbf{b})$	operador k-binario de expansión inverso
$\mathcal{C}_i^{\mu}$	concatenación ordenada

Apéndice C

Análisis comparativo de la técnica de reducción dimensional de los datos

En esta sección se compara el desempeño del método propuesto en el presente trabajo de tesis contra otras tres técnicas frecuentemente utilizadas para reducir dimensionalmente los datos de un problema. La primera técnica busca obtener representaciones equivalentes de los datos, dimensionalmente menores, mediante la aplicación de reglas de lógica difusa y la obtención de un clasificador basado en interconexiones [63]. El método Relief fue introducido en 1992 y demostró claras ventajas sobre los algoritmos de filtrado, sin embargo, su desempeño disminuye en presencia de información redundante [60]. El tercer método generalmente es utilizado para identificar información redundante, no obstante que únicamente puede trabajar con valores discretos, consecuentemente, puede presentar problemas de convergencia [61,62]. El cuarto método, HCM, corresponde al presente trabajo de tesis [45]. Este se basa en la obtención de una máscara que permite la identificación y eliminación de información irrelevante para fines de clasificación; cabe mencionar que el índice de clasificación alcanzado únicamente depende de la cantidad de información relevante seleccionada, además, el proceso de búsqueda del subconjunto de rasgos óptimo depende solamente del valor máximo que pueda tomar la máscara.

La siguiente tabla muestra tanto el número de rasgos seleccionados por cada una de las técnicas, así como el índice de clasificación alcanzado.

Tabla Ap.C.1. Análisis comparativo

	Tamaño original	Selección de rasgos				Índice de clasificación			
		IFN	Relief	ABB	HCM	IFN	Relief	ABB	HCM
Iris	4	1	2	1	2	94.0%	96.0%	90.0%	**95.9%**
Liver	6	5	2	4	2	60.8%	54.9%	54.9%	55.0%
Pima	8	4	1	2	2	72.3%	77.7%	77.3%	69.0%
Breast	9	3	2	3	4	94.0%	93.6%	93.6%	**97.0%**
Wine	13	3	3	2	7	91.7%	95.0%	78.3%	**95.2%**
Heart	13	3	2	4	4	78.0%	74.0%	69.0%	**83.5%**
Credit	14	4	2	5	2	84.9%	86.6%	84.9%	85.5%