# INSTITUTO POLITECNICO NACIONAL

## CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

## CLASIFICADOR AUTOMÁTICO DE ALTO DESEMPEÑO

**T E S I S**

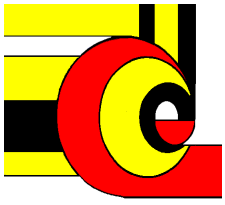
QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

**ITZAMÁ LÓPEZ YÁÑEZ**

DIRECTORES DE TESIS:

**DR. JESÚS FIGUEROA NAZUNO**
**DR. CORNELIO YÁÑEZ MÁRQUEZ**



MÉXICO, D.F. JUNIO DE 2007

# Dedicatoria

El presente trabajo de tesis ha sido el resultado de un gran esfuerzo, no solo de parte mía, sino de todas aquellas personas a mi alrededor que me han apoyado. En particular, esas mujeres y hombres que han sido piedras angulares en la formación del ser humano, el ciudadano, el profesionista, el hombre… que soy ahora.

Esta Tesis está dedicada a mi familia. En primer lugar, a mis padres: mis logros son, además de extensión de sus logros, también sus logros.

A Mamá Clarita: sus oraciones y su fe han sido un gran ejemplo para mi, así como una fuente de inspiración y confianza en el Señor. Dios la Bendiga.

A mis Tíos: Sin su apoyo, adaptarme a esta gran Ciudad de México no hubiera sido lo mismo.

Y así como le dediqué los concursos de sexto, esta tesis también está dedicada a ti, Papá Cornelio.

# Agradecimientos

Gracias Señor por darme vida, salud y discernimiento para terminar con éxito esta etapa de mi vida, así como por todas las experiencias que pusiste en mi camino durante la misma.

Gracias a *Margarita*, mi Madre, y a *Luis Gonzaga*, mi Padre, así como a *Mamá Clarita*, por su incondicional apoyo y comprensión.

Gracias a mis Tías, Tíos, Primas y Primos: ustedes son culpables, en gran medida, de que yo tenga mi casa fuera de casa.

A mis Amigos y Compañeros: ustedes son los culpables de que el CIC sea mi segunda casa.

A mis Asesores *Dr. Cornelio Yáñez Márquez* y *Dr. Jesús Figueroa Nazuno*, por su siempre acertada guía y por tenerme la paciencia que siempre me mostraron, además de ayudarme a alcanzar la meta por los medios que fueron necesarios.

A mis sinodales *Dr. Sergio Suárez Guerra*, *Dr. Oleksiy Pogrebnyak*, *Dra. María Elena Acevedo Mosqueda* y *M. en C. Marco Antonio Moreno Ibarra*, por sus excelentes aportaciones y crítica constructiva.

Al *Centro de Investigación en Computación del IPN*, por ser mi trinchera y mi segunda casa. Las largas horas pasadas ahí fueron inversión, y aquí están los frutos.Todo el personal con el que tuve la dicha de convivir durante mi estancia en el CIC merece mis más sinceras felicitaciones y agradecimiento.

Al *Instituto Politécnico Nacional* por ser mi Segunda Alma Mater. Siempre estarás en mi corazón.

Al *CONACyT* y a la *SIP*, por su apoyo económico, sin el cual el presente trabajo de tesis no sería posible.

# Índice General

# Resumen

En el presente trabajo de tesis se presenta un nuevo algoritmo de clasificación automática de patrones, el Clasificador Gama, mismo que se ubica dentro del Enfoque Asociativo de Reconocimiento de Patrones.

Se incluyen también las definiciones de dos operadores que resultan de gran importancia para la formulación del nuevo algoritmo: el operador $u_\beta$ y, sobre todo, el operador gama de similitud, tanto en su versión normal ($\gamma$) como en su versión generalizada ($\gamma_g$). Asimismo se incluye una versión algorítmica alternativa para el operador gama de similitud generalizado $\gamma_g$.

Además, se presenta un estudio experimental del desempeño del algoritmo propuesto, comparando su rendimiento de clasificación con el exhibido por otros clasificadores, al trabajar con diversas bases de datos de acceso público.

En dichos experimentos se muestra que el algoritmo propuesto es competitivo e, incluso, en algunos casos supera el desempeño mostrado por otros clasificadores de patrones, presentes en la literatura científica contemporánea.

Por otro lado, se presentan tres Teoremas y sus correspondientes demostraciones, que permiten caracterizar las relaciones de orden (tricotomía) de vectores binarios codificados con el código Jonson-Möbius modificado.

Con este trabajo de tesis se engrosan las filas del Enfoque Asociativo de Clasificación de Patrones, con un algoritmo de alto desempeño que, además de ofrecer una eficacia competitiva, ofrece también una eficiencia superior a varios de los mejores clasificadores actuales.

# Abstract

In the current document of thesis, a new algorithm for automatic pattern classification is presented. This novel algorithm, the Gamma Classifier, is a member of the Associative Approach of Pattern Recognition.

Also included are the definitions of two operators which are of great importance for the formulation of the algorithm. These two operators are the $u_\beta$ operator and the gamma similitude operator in both versions: the normal version ($\gamma$) as well as the generalized version ($\gamma_g$). An alternative algorithmic version of the generalized gamma similitude operator ($\gamma_g$) is included too.

An experimental study of the proposed algorithm performance is presented. In this study, the classification performance of the Gamma Classifier is compared to that exhibited by other classifiers, while working with different data bases of public domain.

It is shown with these experiments that the proposed algorithm is quite competitive, even surpassing the performance of other pattern classifiers, present in contemporary scientific literature.

Besides the former, three Theorems and their corresponding Proofs are presented. These Theorems allow characterizing the order relationships (trichotomy) between binary vectors coded with the modified Johnson-Möbius code.

With this work of thesis, the number of classifiers belonging to the Associative Approach of Pattern Recognition has been increased, with a high performance algorithm which, besides offering a competitive efficacy, offers also an efficiency superior to some of the best current classifiers.

# Capítulo 1

# Introducción

En esta tesis se presenta un nuevo clasificador de patrones, que viene a engrosar las filas del Enfoque Asociativo de Clasificación de Patrones. Dicho clasificador, el cual exhibe un desempeño experimental competitivo frente a los clasificadores existentes en la literatura actual, está basado en los operadores $\alpha$ y $\beta$, los cuales fundamentan a las memorias asociativas Alfa-Beta.

## 1.1. Antecedentes

El proceso de reconocer cosas, fenómenos y conceptos, tales como un sonido, cierta comida, un depredador, una imagen, un amigo, un sabor, entre otras instancias, es algo que los seres vivos hacemos de forma inconsciente, pero que nos permite adaptarnos a nuestro entorno y puede ser clave en el momento de la supervivencia [1]. La gran importancia de las tareas de reconocimiento de patrones realizadas cotidianamente, ha conducido a que en ciertos equipos de investigación científica, sobre todo con el advenimiento de los sistemas computacionales modernos, surja la idea de crear, diseñar e implementar sistemas de reconocimiento automático de patrones [2].

Una de las primeras observaciones de quienes están dedicados a tratar de resolver este tipo de problemas en una computadora, es que la solución general es muy complicada, dado que son muchos los factores que están involucrados en el proceso de reconocer. Por ello, la identificación o selección de rasgos o características en los objetos, procesos, fenómenos y conceptos a reconocer de manera automática, es una actividad que se realiza de manera inductiva, de lo simple a lo complejo, de lo concreto a lo abstracto, y con una buena dosis de ensayo y error [3]. No obstante, es preciso aclarar que la selección de rasgos en los modernos sistemas automáticos de reconocimiento de patrones es una línea de investigación vigente [4].

Después de haber seleccionado los rasgos o características de las instancias en estudio, se crean vectores que representan a esas instancias, cada una de cuyas componentes es uno de dichos rasgos o características: a esos vectores se les conoce como patrones, aunque por convención en la literatura científica el término patrón se refiere de manera indistinta a la instancia o al vector que la representa. Acto seguido, se requiere del diseño e implementación de una estrategia que permita crear y operar un sistema computacional que automáticamente reconozca patrones. Esta es la etapa de

reconocimiento de patrones, la cual es posible sólo si existió previamente una etapa de aprendizaje o entrenamiento en el sistema automático [5], [6].

Dependiendo de la aplicación y del problema específico, la fase de reconocimiento de los patrones en el sistema automático puede operarse con al menos dos intenciones: recuperar o clasificar los patrones en estudio [2], [3]. Por ello, gran parte de la literatura relacionada con el área de reconocimiento de patrones menciona explícitamente la clasificación de patrones [7], [8].

Actualmente, entre los principales enfoques de Reconocimiento Automático de Patrones, se encuentran los siguientes:

- *Enfoque estadístico-probabilístico* [3], [5], [9]-[17].
- *Clasificadores basados en métricas* [18]-[29].
- *Enfoque sintáctico-estructural* [9], [12], [30], [31].
- *Enfoque neuronal* [9]-[13], [32]-[51].
- *Enfoque asociativo* [4], [7], [51]-[71]

Entre los algoritmos reconocedores de patrones existentes en la actualidad que pertenecen a los enfoques anteriores, hay dos que sobresalen porque comparten notables características deseables en un algoritmo de este tipo, a saber: su sencillez y su alta eficacia.

Uno es el clasificador $k$-NN ($k$-nearest neighbor, los $k$-vecinos más cercanos), el cual es un modelo de mínima distancia y se ubica en los clasificadores basados en métricas. Al utilizar el algoritmo $k$-NN se calcula la distancia de un patrón de prueba respecto a cada uno de los patrones de aprendizaje o entrenamiento, se ordenan las distancias de menor a mayor y se retiene la clase que se obtiene por mayoría entre los $k$ patrones más cercanos [18], [27], [72], [73]. En el clásico proyecto Statlog, desarrollado en Escocia por Michie y sus colaboradores a inicios de la década de los noventa, se muestra experimentalmente la superioridad del $k$-NN respecto de un buen número de clasificadores automáticos de patrones [13]. Sin embargo, la gran desventaja del $k$-NN es su poca eficiencia, puesto que cuando se trabaja con una conjunto grande de patrones de aprendizaje, el hecho de calcular las distancias de todos esos patrones con respecto al patrón de prueba y posteriormente ordenarlas, es un proceso computacionalmente caro [18], [74]-[76].

El otro enfoque de los dos anteriormente mencionados es el asociativo, mismo que se inició con un trabajo de tesis de maestría en ciencias de la computación [52], desarrollada en 2002 en el CIC-IPN. Este enfoque se basa en las memorias asociativas, cuyos modelos matemáticos pioneros [77] son contemporáneos a los primeros modelos de redes neuronales [33]. El estado del arte en las memorias asociativas que sirven de base a modelos asociativos de reconocimiento de patrones, está constituido por las memorias asociativas Alfa-Beta, cuyo modelo se basa en dos operaciones simples, Alfa y Beta, las cuales son equiparables en sencillez a las operaciones básicas de la lógica booleana [53]. En un número importante de investigaciones recientes, se ha mostrado teórica y experimentalmente que, a pesar de su sencillez, los modelos de reconocimeinto

automático de patrones basados en las memorias asociativas Alfa-Beta son altamente eficaces, así como competitivos con los modelos reportados en la literatura actual [4], [7], [8], [51]-[67], [70], [71], [78]-[86]. Además, estos modelos son más eficientes que el $k$-NN para conjuntos grandes de patrones de aprendizaje [84]. Cabe mencionar que las memorias asociativas Alfa-Beta, cuyos operadores sirven de fundamento al presente documento de tesis, trabajan solamente con patrones binarios [53], [54], [56], [85].

De esta manera, la presente tesis queda delimitada dentro del área de investigación del enfoque asociativo de reconocimiento automático de patrones, en sus tareas de recuperación y, particularmente, clasificación de patrones.

## 1.2. Justificación

Como fue mostrado por Michie *et al.* en los resultados arrojados por el proyecto Statlog [13], en términos generales se necesita sacrificar algo de eficiencia para obtener mejores resultados en eficacia con respecto a la tarea de clasificación de patrones, puesto que algunos de los mejores clasificadores presentan un alto costo computacional, sobre todo para conjuntos grandes de patrones de aprendizaje. Lo anterior sugiere que es deseable encontrar algoritmos clasificadores de patrones que sean capaces de competir en cuanto a eficacia con los mejores clasificadores, ofreciendo la ventaja de un menor costo computacional o una mayor sencillez: este hecho justifica el presente trabajo de tesis.

## 1.3. Objetivo

Diseñar e implementar un algoritmo de reconocimiento de patrones, que ofrezca un desempeño competitivo en la tarea de clasificación de patrones respecto de los clasificadores presentes en la literatura actual. Este nuevo algoritmo se basará en los operadores $\alpha$ y $\beta$ y sus propiedades, tomados de los modelos asociativos Alfa-Beta.

## 1.4. Contribuciones

- Un algoritmo de clasificación de patrones, que es competitivo con respecto de los clasificadores de patrones actuales.
- Un operador unario, el operador $u_\beta$.
- Dos operadores vectoriales, el operador gama de similitud $\gamma(\mathbf{x}, \mathbf{y}, \theta)$ y el operador gama de similitud generalizado $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$.
- Un algoritmo alternativo para calcular el resultado del operador gama de similitud generalizado $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$.
- Tres teoremas acerca de las relaciones de orden (tricotomía) entre vectores binarios codificados con el código Johnson-Möbius modificado.
- Aplicaciones.

## 1.5. Organización del documento

En este Capítulo se han presentado: los antecedentes, la justificación, el objetivo y las contribuciones de este trabajo de tesis . El resto del documento está organizado como se describe a continuación.

En el Capítulo 2 se presenta el estado del arte del área de investigación dentro de la cual se encuentra circunscrita la presente tesis, que es la clasificación de patrones en general y el enfoque asociativo de clasificación de patrones en particular. Por otro lado, aquellas herramientas matemáticas, materiales y métodos necesarios para la presentación del algoritmo propuesto, tanto en el cuerpo del documento como en el Apéndice B, serán expuestos en el Capítulo 3.

El Capítulo 4 es la parte más relevante de este documento. En el mismo, se introduce el algoritmo propuesto, junto con las operaciones y herramientas matemáticas desarrolladas en este trabajo de tesis, que le dan sustento. El contenido del Capítulo incluye el desarrollo del algoritmo, su fundamentación y ejemplos ilustrativos de clasificación.

Los resultados experimentales, así como la discusión de los mismos, se presentan en el Capítulo 5; y en el Capítulo final, el 6, se exponen las conclusiones y recomendaciones para trabajo futuro.

Cuatro Apéndices complementan el documento de tesis: en el Apéndice A se presenta la simbología utilizada a lo largo del documento, mientras que en el B se incluye una versión algorítmica alternativa al operador central de la propuesta. El Apéndice C, por su parte, está integrado por algunos Teoremas desarrollados durante el presente trabajos de tesis, relacionados con vectores binarios codificados con el código Johnson-Möbius modificado. El Apéndice último, el D, consta del manual de operación del software diseñado e implementado *ex-profeso* para este trabajo de tesis.

Finalmente, se incluyen las referencias bibliográficas.

# Capítulo 2

# Estado del Arte

En el presente Capítulo se abordan algunos de los principales modelos de clasificación de patrones, mismos que integran el estado del arte del reconocimiento de patrones en su tarea de clasificación.

Anteriormente se mencionó que existen varios enfoques que atacan los problemas de reconocimiento de patrones de diversas formas; cada uno de ellos tiene ventajas y desventajas que les permiten funcionar mejor en algunos problemas que en otros. Actualmente, entre los principales enfoques de Reconocimiento Automático de Patrones, se encuentran los siguientes:

- *Enfoque estadístico-probabilístico.* Su principal clasificador está basado en la teoría de la probabilidad, y específicamente en el teorema de Bayes. Es históricamente el primer enfoque que existió y probablemente el más desarrollado [3], [5], [9]-[17].
- *Clasificadores basados en métricas.* Usualmente se ubican dentro del enfoque estadístico y se basan en el concepto de métrica y en las propiedades de los espacios métricos para hacer la clasificación [18]-[29].
- *Enfoque sintáctico-estructural.* Se basa en la teoría de autómatas y lenguajes formales para hacer la clasificación. Se enfoca más en la estructura de las cosas a clasificar que en mediciones numéricas [9], [12], [30], [31].
- *Enfoque neuronal.* Se basa en modelos matemáticos de las neuronas del cerebro humano y, a diferencia del enfoque estadístico-probabilístico, los sistemas automáticos de reconocimiento de patrones basados en en el enfoque neuronal, además de clasificar patrones, también son capaces de recuperarlos [9]-[13], [32]-[51].
- *Enfoque asociativo.* Este enfoque fue creado en el Centro de Investigación en Computación del IPN en 2002, y utiliza los modelos de memorias asociativas para diseñar e implementar reconocedores de patrones robustos ante la presencia de patrones ruidosos. Los sistemas automáticos de reconocimiento de patrones basados en en el enfoque asociativo, son capaces de realizar la tarea de clasificación de patrones, como un caso particular de la tarea principal que realizan de manera eficiente: recuperar patrones [4], [7], [51]-[71].

En cada uno de los enfoques se han creado algoritmos para diseñar reconocedores de patrones. Por ejemplo, el reconocedor de patrones más famoso en el enfoque estadístico-probabilístico es el clasificador bayesiano [3], y el clasificador euclideano se identifica con el enfoque basado en métricas [87]. Sin embargo, la mayoría de los algoritmos de reconocimiento de patrones comparten características de más de un enfoque y elementos de otras áreas de la ciencia, especialmente de disciplinas matemáticas. Esto es particularmente notorio en el clasificador euclideano dado que, no obstante que es un clasificador basado en métricas, también utiliza de manera importante la teoría de las funciones discriminantes, misma que no interviene en otros clasificadores basados en métricas. Esta situación de tomar elementos de un enfoque específico de reconocimiento de patrones y algunas disciplinas matemáticas sucede también con las Máquinas de Soporte Vectorial (en la literatura científica actual, se refiere a este modelo por sus siglas en inglés, SVM, por lo que en el presente documento continuaremos con esta tendencia), técnica desarrollada por Vapnik que utiliza un algoritmo de entrenamiento, el cual maximiza el margen entre los patrones en el límite de clases, y estos patrones son llamados por Vapnik *vectores de soporte* (support vectors) [88], [89]. Cabe mencionar que esta técnica ha sido aplicada exitosamente en diversas áreas de la actividad humana [90]-[93].

A continuación se hará hincapié en lo ya mencionado en el Capítulo 1 respecto de dos algoritmos clasificadores de patrones, relevantes por su sencillez y su alta eficacia. Uno es el clasificador $k$-NN, el cual es un modelo de mínima distancia y se ubica en los clasificadores basados en métricas. En el clásico proyecto Statlog se muestra experimentalmente la superioridad del $k$-NN respecto de un buen número de clasificadores automáticos de patrones [13]. Sin embargo, la gran desventaja del $k$-NN es su poca eficiencia al trabajar con una conjunto amplio de patrones de aprendizaje, puesto que calcular las distancias entre todos los patrones y ordenarlas es un proceso computacionalmente caro [18], [74]-[76]. El enfoque asociativo, por su parte, se inició con el CHAT (Clasificador Híbrido Asociativo con Traslación) [52], desarrollado en 2002 por Raúl Santiago Montero en el CIC-IPN, como tema central de su tesis de maestría. Este enfoque se basa en las memorias asociativas, cuyo estado del arte incluye a las memorias asociativas Alfa-Beta. En un número importante de investigaciones recientes, se ha mostrado teórica y experimentalmente que, a pesar de su sencillez, los modelos de reconocimeinto automático de patrones basados en las memorias asociativas Alfa-Beta son altamente eficaces, así como competitivos con los modelos reportados en la literatura actual [4], [7], [8], [51]-[67], [70], [71], [78]-[86]. Adicionalmente, es preciso hacer notar que los modelos asociativos son más eficientes que el $k$-NN para conjuntos grandes de patrones de aprendizaje [84], pues se basan en operaciones equiparables en sencillez a las operaciones básicas de la lógica booleana (estas operaciones y sus propiedades serán descritas en el Capítulo 3) [53], [54], [56], [85].

## 2.1. Clasificador Bayesiano

El contenido de la presente sección está basado en las referencias [3], [17], [94]-[102], en algunas de las cuales se especifica que de la definición de probabilidad condicional, surge uno de los teoremas más importantes en la teoría de la probabilidad, y en particu-

lar, en el enfoque probabilístico-estadístico de reconocimiento de patrones: el Teorema de Bayes. La importancia de éste radica en que es la base del clasificador bayesiano. Es notable la afirmación debida a Duda y Hart, autores de uno de los textos más utilizados a nivel mundial en los cursos de reconocimiento de patrones, respecto del clasificador bayesiano al presentar la Teoría de la Decisión Beyesiana:

> "Empezamos [con la Teoría de la Decisión Beyesiana] considerando el caso ideal en el cual la estructura estadística inherente a las categorías es perfectamente conocida. Mientras que este tipo de situación raramente se presenta en la práctica, [esta teoría] nos permite determinar el clasificador óptimo (Bayesiano) contra el cual podemos comparar todos los demás clasificadores"... ([3], pp. 17).

**Teorema 2.1** ***Teorema de Bayes.*** *Sean los eventos $A_1, A_2, ..., A_n$ una partición del espacio muestral $X$ y sea $B$ un evento dentro del mismo espacio, entonces:*

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{P(B)} = \frac{P(A_i)P(B|A_i)}{\sum_{k=1}^{n} P(A_k)P(B|A_k)}$$

■

El teorema de Bayes es muy importante porque permite cambiar el sentido de la probabilidad condicional; es especialmente útil cuando es más fácil calcular la probabilidad de $B$ dado $A_i$ que de $A_i$ dado $B$, y se puede explicar con palabras de la siguiente forma:

$$posterior = \frac{apriori \text{ X } verosimilitud}{evidencia}$$

donde $P(A_i)$ define el conocimiento *a priori* del problema, es decir, la probabilidad absoluta de que suceda $A_i$ antes de saber cualquier cosa sobre $B$; la probablidad posterior $P(B|A_i)$ define la *verosimilitud*, es decir, qué tan probable es que suceda el evento $B$ dentro del espacio de trabajo reducido de $A_i$; luego, $P(B) = \sum_{k=1}^{n} P(A_k)P(B|A_k)$ es la *evidencia*, lo que indica cuál es la probabilidad de que ocurra $B$ si se tiene todo el conocimiento *a priori* de toda la partición y la verosimilitud de $B$ dentro de cada espacio de trabajo $A_k$ (usualmente sólo se ve como un factor de normalización); finalmente, $P(A_i|B)$ es el conocimiento posterior o consecuente de que suceda el evento $A_i$ dado que ocurrió $B$.

Usualmente, los eventos en el teorema de Bayes están expresados en términos de variables aleatorias y distribuciones de probabilidad, por lo que normalmente en la práctica el teorema toma la siguiente forma:

$$P(A_i|B) = \frac{P(A_i)p(B|A_i)}{P(B)} = \frac{P(A_i)p(B|A_i)}{\sum_{k=1}^{n} P(A_k)p(B|A_k)} \tag{2.1}$$

luego, para conocer la probabilidad absoluta $P(A_i|B)$ es necesario conocer todas las distribuciones de probabilidad asociadas al problema a resolver y todas las probabilidades *a priori*.

Lo anterior puede ser útil para reconocer patrones si las clases y los patrones se modelan como eventos o variables aleatorias. La idea general es la siguiente: un patrón (evento representado por una variable aleatoria vectorial $X$) pertenece a la clase $i$ (evento representado por la variable aleatoria $C_i$) si su probabilidad de pertenecer a esa clase es más grande que la probabilidad de pertenecer a las demás clases.

El clasificador Bayesiano toma ventaja de las probabilidades condicionales al sustituir, en el proceso de clasificación, el teorema de Bayes en la siguiente regla:

$$X \in C_i, \text{ si } P(C_i \mid X) > P(C_j \mid X)\ \forall i \neq j \tag{2.2}$$

cuya interpretación lógica es la siguiente: si se conoce que el patrón $X$ ocurrió (es decir que fue presentado al sistema), se calcula la probabilidad de que ocurra $C_k$ $\forall k = 1, 2, ..., n$ y se clasifica en la clase $C_i$ si dicha probabilidad es la mayor de todas, es decir, si la probabilidad de pertenencia de $X$ a $C_i$ es mayor a cualquier otra $C_j$.

Al usar el teorema de Bayes en 2.2, y tomando en cuenta que las probabilidades siempre son positivas, queda lo siguiente:

$$\begin{aligned} P(C_i \mid X) &> P(C_j \mid X) \\ \frac{P(C_i)\ p(X \mid C_i)}{P(X)} &> \frac{P(C_j)\ p(X \mid C_j)}{P(X)} \\ P(C_i)\ p(X \mid C_i) &> P(C_j)\ p(X \mid C_j) \end{aligned} \tag{2.3}$$

Ahora bien, dado que ln (la función logaritmo natural) es una función monótona creciente, es decir del tipo: $f(x) < f(y) \Leftrightarrow x < y$, se puede hacer la siguiente sustitución en la expresión 2.3:

$$\begin{aligned} \ln\left(P(C_i)\ p(X \mid C_i)\right) &> \ln\left(P(C_j)\ p(X \mid C_j)\right) \\ \ln(P(C_i)) + \ln(p(X \mid C_i)) &> \ln(P(C_j)) + \ln(p(X \mid C_j)) \\ d_i &> d_j, \text{ con } d_k = \ln(P(C_k)) + \ln(p(X \mid C_k)) \end{aligned} \tag{2.4}$$

donde $d_k$ define una función discriminante para el clasificador.

Así pues, que un patrón desconocido sea clasificado en una clase $C_i$ en particular, implica tener todo el conocimiento *a priori* de cada clase $C_i$ y su distribución de probabilidad correspondiente, lo que raramente sucede en la práctica [3].

Tomando en cuenta lo anterior, el algoritmo para diseñar un clasificador Bayesiano queda como sigue:

**Algoritmo 2.2** *Algoritmo del clasificador Bayesiano.*

1. Obtener una muestra representativa $S$ de los objetos a clasificar.
2. Determinar cada una de las clases $C_k$ que formarán parte del sistema.

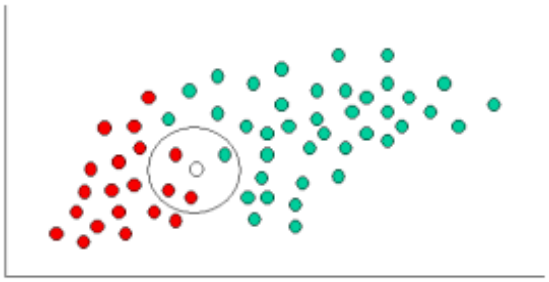


Figura 2.1: Clasificador Bayesiano. Ejemplo de clasificación del elemento blanco, dada la probabilidad de verde y rojo en su vecindad

3. Determinar, con base en la muestra y en la cardinalidad de cada clase, las probabilidades $P(C_k)$.

4. Determinar los rasgos útiles que se van a utilizar para clasificar, y elaborar cada distribución de probabilidad $p(X \mid C_k)$ la cual va a ser dependiente del número y naturaleza de cada rasgo de la variable aleatoria vectorial $X$.

5. Aplicar la siguiente regla para clasificar un patrón desconocido de entrada $X$, :

$$X \in C_i, \text{ si } d_i > d_j \ \forall i \neq j, \text{ con } d_k = \ln(P(C_k)) + \ln(p(X \mid C_k)) \tag{2.5}$$

■

En la figura 2.1 se puede ver un ejemplo de clasificación usando el clasificador bayesiano, tomada de [103].

Como se puede apreciar en los párrafos anteriores, este clasificador, si bien es muy robusto, también tiene la desventaja de que se debe tener una estadística muy amplia y completa sobre todas las variables aleatorias que forman parte del sistema. Entre más grande sea la muestra, y por tanto las mediciones estadísticas sobre ella, más confiable serán los resultados del clasificador, lo cual de alguna manera significa haber hecho el proceso de clasificación a mano durante mucho tiempo para poder tener una buena respuesta. Dado que esta situación pocas veces se presenta en la práctica, el uso del clasificador bayesiano se ve limitado, forzando a los investigadores en este campo a establecer condicines artificiales a las probabilidades condicionales de modo que sea funcional su uso.

## 2.2. Clasificador Euclideano

El funcionamiento básico de los clasificadores basados en métricas, entre los que se encuentra el Clasificador Euclideano [87], [25], [28], [104], [105] se muestra en el siguiente algoritmo:

**Algoritmo 2.3** *Algoritmo de clasificación de patrones basados en una métrica.*

1. Escoger una muestra de patrones clasificada de antemano en $n$ clases $\{C_1, C_2, .., C_n\}$ y una métrica $d$.
2. Con base en la muestra y para cada clase $C_i$, encontrar un patrón $v_i$ que la represente mejor.
3. Si $x$ es un patrón de dimensión $n$ cuya clase se desconoce, este patrón será clasificado en la clase $C_i$, si se cumple lo siguiente:

$$\forall j, j \neq i, d(x, v_i) \leq d(x, v_j)$$

■

Aquí aparece una primera debilidad de este enfoque, ya que el método que se utilice para decidir qué patrón es el mejor representante para una clase, y la métrica elegida, influirán en gran medida en el desempeño del clasificador.

Si en el algortimo anterior se especifica que la métrica a utilizar es la distancia euclideana, entonces se tiene el clasificador euclideano. Esta métrica es intuitivamente la que se utiliza para medir distancias, ya que equivale a medir el tamaño del segmento de recta que une a dos puntos y es la que normalmente se utiliza en geometría analítica y en análisis vectorial. La *distancia euclideana* entre dos vectores $\mathbf{x}$ y $\mathbf{y}$ de dimensión $n \in \mathbb{Z}^+$, con componentes $x_i$ y $y_i$ respectivamente, donde $i \in \{1, 2, \ldots, n\}$, se denota por $d_2(\mathbf{x}, \mathbf{y})$ y se calcula de la siguiente forma:

$$d_2(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^{n} (x_i - y_i)^2\right]^{\frac{1}{2}} \tag{2.6}$$

o en forma vectorial:

$$d_2(\mathbf{x}, \mathbf{y}) = \left[(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})\right]^{\frac{1}{2}} \tag{2.7}$$

A continuación se enuncia un primer algoritmo para el clasificador euclideano el cual es, en escencia, equivalente al algoritmo 2.3.

**Algoritmo 2.4** *Primer algoritmo del clasificador euclideano.*

1. Escoger una muestra de patrones clasificada de antemano en $p$ clases $\{C_1, C_2, .., C_p\}$.
2. Con base en la muestra y para cada clase $C_i$ , calcular el patrón representante $\mu_i = \frac{1}{k} \sum_{x_j \in C_i} x_j$, donde $k$ es el número de elementos en la muestra que pertenecen a $C_i$.

3. Sea $x$ un patrón de dimensión $n$ cuya clase se desconoce. Utilizando la distancia euclideana $d_2$, este patrón será clasificado en la clase $C_i$, si se cumple lo siguiente:

$$\forall j, j \neq i, d_2(x, \mu_i) \leq d_2(x, \mu_j)$$

■

Ahora bien, se puede reescribir el algoritmo del clasificador euclideano con base en el concepto de función discriminante, como sigue.

**Algoritmo 2.5** *Algoritmo del clasificador euclideano que usa funciones discriminantes.*

1. Escoger una muestra de patrones clasificada de antemano en $p$ clases $\{C_1, C_2, .., C_p\}$.
2. Con base en la muestra y para cada clase $C_i$ , calcular el patrón representante $\mu_i = \frac{1}{k} \sum_{x_j \in C_i} x_j$, donde $k$ es el número de elementos en la muestra que pertenecen a $C_i$ .
3. Generar funciones discriminantes $d_{ij}(x)$ para cada par de clases $C_i, C_j$, de forma que $d_{ij}(x) = (\mu_i - \mu_j)^T x - \frac{(\mu_i - \mu_j)^T(\mu_i + \mu_j)}{2}$.
4. Si $x$ es un patrón de dimensión $n$ cuya clase se desconoce, este patrón será clasificado en la clase $C_i$, si se cumple lo siguiente:

$$\forall j, j \neq i, d_{ij}(x) \geq 0$$

■

Así, el clasificador euclideano puede ser visto como un conjunto de funciones discriminantes, que ayudan a decidir si un patrón pertenece a una clase dada. Los patrones que hacen $d_{ij}$ igual a cero definen una *frontera* entre las dos clases $C_i$ y $C_j$. Un ejemplo de lo anterior está representado en la figura 2.2, tomada de [25].

**Nota 2.6** *El clasificador euclideano clasifica correctamente si las clases $C_i$ y $C_j$ son linealmente separables.*

El clasificador euclideano, aunque simple de implementar, posee limitaciones fuertes, como el hecho de que para se presente clasificación correcta, las clases tienen que ser linealmente separables, por lo que la aplicación de este modelo a problemas prácticos resulta muy reducida, y los investigadores se han visto en la necesidad de modificar sustancialmente el modelo para que sea funcional.

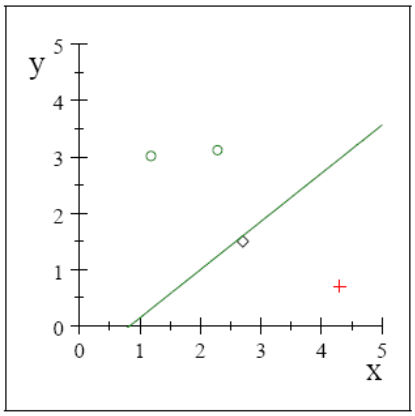



Figura 2.2: Clasificador Euclideano

## 2.3. Support Vector Machines (SVM)

El algoritmo utilizado por las SVMs funciona como un clasificador biclase que minimiza simultáneamente el error empírico de clasificación y maximiza algunas características de las métricas involucradas [106], [107]. Como tal, este algoritmo está fuertemente basado en la teoría de aprendizaje estadístico desarrollado por Vapnik, Chervonenkis y otros, que dio lugar a la implementación de las SVM durante la década de los noventa, en los Bell Laboratories de AT&T por Vapnik y sus colaboradores [88], [89].

El problema básico que ataca este modelo es el de de separar un hiperplano $n$-dimensional en dos clases, por medio de un hiperplano $n-1$-dimensional. Sin embargo, existen más de un hiperplano que logra este cometido. El objetivo de las SVM es encontrar el hiperplano óptimo que mejor generalice la clasificación. Para ello, se utiliza el concepto de *vector de soporte*, que se refiere a los patrones más cercanos al hiperplano buscado; dichos patrones se encuentran en la frontera de las clases. Para encontrar ese hiperplano óptimo, se maximiza la distancia a los vectores de soporte. De manera informal, se puede afirmar que los vectores de soporte son los patrones que proporcionan más información para la tarea de clasificación [3].

Ahora bien, puede darse el caso de que existan patrones, cercanos a la frontera de las clases, que se comportan más como excepciones. Si dichos patrones se tomaran como vectores de soporte, degradarían la generalización de la clasificación. Para solucionar este problema, se incluye cierto margen de error, con lo cual se admite que ciertos patrones cercanos a la frontera no sean tomados como vectores de soporte, siempre y cuando se mantengan a una distancia menor o igual al margen de error establecido. Entonces, el problema de encontrar el hiperplano que divide las dos clases de manera óptima, se resuelve maximizando la distancia entre dicho hiperplano y los patrones más cercanos a la frontera de las clases (vectores de soporte), a la vez que se minimiza el error [89], [108], [109]. Un ejemplo de SVM se puede apreciar en la figura 2.3.

En caso de que el problema no sea linealmente separable en el espacio original, se utiliza una transformación por medio de una función *kernel*, para llevar los patrones

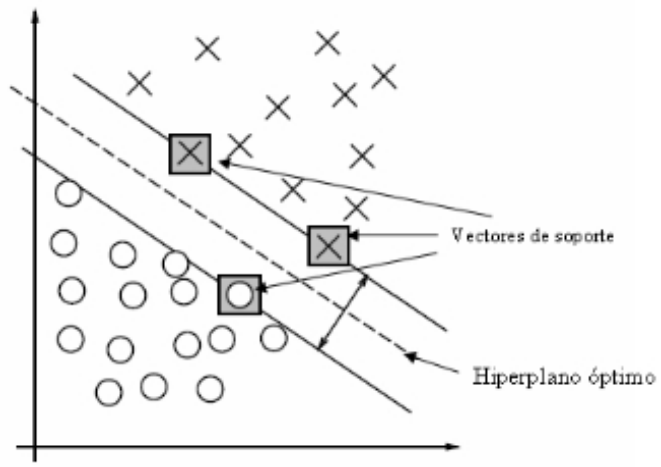



Figura 2.3: Support Vector Machine. Los vectores de soporte están indicados con un recuadro mientras que el hiperplano óptimo aparece como una línea punteada

a un espacio en una dimensión mayor, donde el problema sea linealmente separable [109].

## 2.4. k-Nearest Neighbor (k-NN)

Uno de los algoritmos de clasificación de patrones más notables es sin duda el del *k-Nearest Neighbor* [18]-[24], [26], [27], [72], [73], [110], [111]; a pesar de contar con cuatro décadas de existencia, sigue siendo uno de los más eficaces, como lo mostró Michie con los resultados del proyecto Statlog [13]. En la Figura 2.4 se puede ver un ejemplo de 5-NN, tomada de [3]. A continuación se describe el algoritmo $k$-NN, tanto para el caso de $k = 1$ como para $k > 1$.

**Algoritmo 2.7** *Algoritmo del k-NN para k = 1:*

1. Seleccionar la métrica a utilizar.
2. Calcular la distancia de un patrón $x$ por clasificar, a cada uno de los patrones del conjunto fundamental.
3. Obtener la distancia mínima.
4. Asignar a $x$ la clase del patrón con la mínima distancia.

■

**Algoritmo 2.8** *Algoritmo del k-NN para k > 1:*

1. Seleccionar la métrica a utilizar.

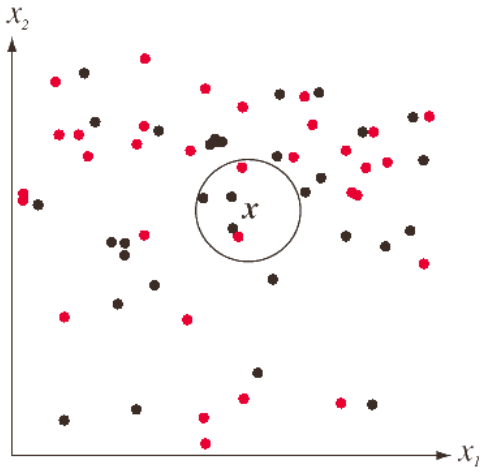



Figura 2.4: *k-Nearest Neighbor*. Para $k = 5$, el punto $x$ sería clasificado con la clase de los puntos negros (3 puntos negros contra 2 puntos rojos)

2. Calcular la distancia de un patrón $x$ por clasificar, a cada uno de los patrones del conjunto fundamental.

3. Ordenar los datos en orden ascendente.

4. Obtener los $k$ menores valores de distancia.

5. Usar la regla de *majority* para asignar la clase al patrón $x$.
   ■

No obstante que el $k$-NN es altamente eficaz, su gran desventaja es la baja eficiencia mostrada cuando se trabaja con un conjunto grande de patrones [18], [74]-[76].

## 2.5. Clasificador Híbrido Asociativo con Traslación (CHAT)

Este clasificador es históricamente el primer representante del enfoque asociativo, pues toma conceptos del área de memorias asociativas para llevar a cabo la tarea de clasificación de patrones [7], [52]. En particular, el CHAT toma principios de la *Lernmatrix* de Steinbuch y del *Linear Associator* de Anderson-Kohonen para realizar la tarea de clasificación, a pesar de que ambos modelos por separado tienen varias desventajas.

Por un lado, la Lernmatrix puede aceptar sólo patrones binarios como entradas, y se llega rápidamente a la saturación, fenómeno que impide la clasificación correcta. Por otro lado, no obstante que el Linear Associator elimina la restricción de patrones binarios a la entrada, puesto que puede aceptar patrones con valores reales en sus componentes, surge una restricción bastante fuerte: se requiere la ortonormalidad de los patrones de entrada para que la clasificación sea correcta.

La combinación de la fase de aprendizaje del Linear Associator y la de recuperación de la Lernmatrix dan como resultado el Clasificador Híbrido Asociativo (CHA), que presenta algunos problemas cuando la magnitud de los patrones pertecientes a una clase es considerablemente mayor a la magnitud de los patrones de las otras clases. Para subsanar este problema, el CHAT realiza una traslación de ejes previa al procesamiento de los patrones.

El algoritmo del CHAT es como sigue:

**Algoritmo 2.9** *Algoritmo del clasificador híbrido asociativo con traslación:*

1. Sea un conjunto fundamental de patrones de entrada de dimensión $n$ con valores reales en sus componentes (a la manera del Linear Associator), que se aglutinan en $m$ clases diferentes.
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 coordenada $k$-ésima, donde el valor es uno (a la manera de la Lernmatrix).
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 ejes coordenados.
5. Se realiza la traslación de todos los patrones del conjunto fundamental.
6. La fase de aprendizaje es similar a la del Linear Associator.
7. La fase de recuperación es similar a la que usa la Lernmatrix.
8. Se traslada todo patrón a clasificar a los nuevos ejes.
9. Se procede a clasificar los patrones desconocidos.

■

El autor del CHAT describe en su trabajo de tesis [52] una gran cantidad de experimentos donde se exhibe la superioridad del enfoque asociativo de clasificación de patrones, respecto de algunos clasificadores de actualidad.

En el Capítulo 5 del presente trabajo de tesis se presentarán los resultados de estudios experimentales realizados con bases de datos de prestigio, con objeto de comparar el desempeño de los clasificadores mostrados en diferentes publicaciones actuales, respecto del algoritmo de clasificador asociativo, tema central de esta tesis.

# Capítulo 2

# Estado del Arte

En el presente Capítulo se abordan algunos de los principales modelos de clasificación de patrones, mismos que integran el estado del arte del reconocimiento de patrones en su tarea de clasificación.

Anteriormente se mencionó que existen varios enfoques que atacan los problemas de reconocimiento de patrones de diversas formas; cada uno de ellos tiene ventajas y desventajas que les permiten funcionar mejor en algunos problemas que en otros. Actualmente, entre los principales enfoques de Reconocimiento Automático de Patrones, se encuentran los siguientes:

- *Enfoque estadístico-probabilístico.* Su principal clasificador está basado en la teoría de la probabilidad, y específicamente en el teorema de Bayes. Es históricamente el primer enfoque que existió y probablemente el más desarrollado [3], [5], [9]-[17].
- *Clasificadores basados en métricas.* Usualmente se ubican dentro del enfoque estadístico y se basan en el concepto de métrica y en las propiedades de los espacios métricos para hacer la clasificación [18]-[29].
- *Enfoque sintáctico-estructural.* Se basa en la teoría de autómatas y lenguajes formales para hacer la clasificación. Se enfoca más en la estructura de las cosas a clasificar que en mediciones numéricas [9], [12], [30], [31].
- *Enfoque neuronal.* Se basa en modelos matemáticos de las neuronas del cerebro humano y, a diferencia del enfoque estadístico-probabilístico, los sistemas automáticos de reconocimiento de patrones basados en en el enfoque neuronal, además de clasificar patrones, también son capaces de recuperarlos [9]-[13], [32]-[51].
- *Enfoque asociativo.* Este enfoque fue creado en el Centro de Investigación en Computación del IPN en 2002, y utiliza los modelos de memorias asociativas para diseñar e implementar reconocedores de patrones robustos ante la presencia de patrones ruidosos. Los sistemas automáticos de reconocimiento de patrones basados en en el enfoque asociativo, son capaces de realizar la tarea de clasificación de patrones, como un caso particular de la tarea principal que realizan de manera eficiente: recuperar patrones [4], [7], [51]-[71].

En cada uno de los enfoques se han creado algoritmos para diseñar reconocedores de patrones. Por ejemplo, el reconocedor de patrones más famoso en el enfoque estadístico-probabilístico es el clasificador bayesiano [3], y el clasificador euclideano se identifica con el enfoque basado en métricas [87]. Sin embargo, la mayoría de los algoritmos de reconocimiento de patrones comparten características de más de un enfoque y elementos de otras áreas de la ciencia, especialmente de disciplinas matemáticas. Esto es particularmente notorio en el clasificador euclideano dado que, no obstante que es un clasificador basado en métricas, también utiliza de manera importante la teoría de las funciones discriminantes, misma que no interviene en otros clasificadores basados en métricas. Esta situación de tomar elementos de un enfoque específico de reconocimiento de patrones y algunas disciplinas matemáticas sucede también con las Máquinas de Soporte Vectorial (en la literatura científica actual, se refiere a este modelo por sus siglas en inglés, SVM, por lo que en el presente documento continuaremos con esta tendencia), técnica desarrollada por Vapnik que utiliza un algoritmo de entrenamiento, el cual maximiza el margen entre los patrones en el límite de clases, y estos patrones son llamados por Vapnik *vectores de soporte* (support vectors) [88], [89]. Cabe mencionar que esta técnica ha sido aplicada exitosamente en diversas áreas de la actividad humana [90]-[93].

A continuación se hará hincapié en lo ya mencionado en el Capítulo 1 respecto de dos algoritmos clasificadores de patrones, relevantes por su sencillez y su alta eficacia. Uno es el clasificador $k$-NN, el cual es un modelo de mínima distancia y se ubica en los clasificadores basados en métricas. En el clásico proyecto Statlog se muestra experimentalmente la superioridad del $k$-NN respecto de un buen número de clasificadores automáticos de patrones [13]. Sin embargo, la gran desventaja del $k$-NN es su poca eficiencia al trabajar con una conjunto amplio de patrones de aprendizaje, puesto que calcular las distancias entre todos los patrones y ordenarlas es un proceso computacionalmente caro [18], [74]-[76]. El enfoque asociativo, por su parte, se inició con el CHAT (Clasificador Híbrido Asociativo con Traslación) [52], desarrollado en 2002 por Raúl Santiago Montero en el CIC-IPN, como tema central de su tesis de maestría. Este enfoque se basa en las memorias asociativas, cuyo estado del arte incluye a las memorias asociativas Alfa-Beta. En un número importante de investigaciones recientes, se ha mostrado teórica y experimentalmente que, a pesar de su sencillez, los modelos de reconocimeinto automático de patrones basados en las memorias asociativas Alfa-Beta son altamente eficaces, así como competitivos con los modelos reportados en la literatura actual [4], [7], [8], [51]-[67], [70], [71], [78]-[86]. Adicionalmente, es preciso hacer notar que los modelos asociativos son más eficientes que el $k$-NN para conjuntos grandes de patrones de aprendizaje [84], pues se basan en operaciones equiparables en sencillez a las operaciones básicas de la lógica booleana (estas operaciones y sus propiedades serán descritas en el Capítulo 3) [53], [54], [56], [85].

## 2.1. Clasificador Bayesiano

El contenido de la presente sección está basado en las referencias [3], [17], [94]-[102], en algunas de las cuales se especifica que de la definición de probabilidad condicional, surge uno de los teoremas más importantes en la teoría de la probabilidad, y en particu-

lar, en el enfoque probabilístico-estadístico de reconocimiento de patrones: el Teorema de Bayes. La importancia de éste radica en que es la base del clasificador bayesiano. Es notable la afirmación debida a Duda y Hart, autores de uno de los textos más utilizados a nivel mundial en los cursos de reconocimiento de patrones, respecto del clasificador bayesiano al presentar la Teoría de la Decisión Beyesiana:

> "Empezamos [con la Teoría de la Decisión Beyesiana] considerando el caso ideal en el cual la estructura estadística inherente a las categorías es perfectamente conocida. Mientras que este tipo de situación raramente se presenta en la práctica, [esta teoría] nos permite determinar el clasificador óptimo (Bayesiano) contra el cual podemos comparar todos los demás clasificadores"... ([3], pp. 17).

**Teorema 2.1** ***Teorema de Bayes.*** *Sean los eventos $A_1, A_2, ..., A_n$ una partición del espacio muestral $X$ y sea $B$ un evento dentro del mismo espacio, entonces:*

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{P(B)} = \frac{P(A_i)P(B|A_i)}{\sum_{k=1}^{n} P(A_k)P(B|A_k)}$$

■

El teorema de Bayes es muy importante porque permite cambiar el sentido de la probabilidad condicional; es especialmente útil cuando es más fácil calcular la probabilidad de $B$ dado $A_i$ que de $A_i$ dado $B$, y se puede explicar con palabras de la siguiente forma:

$$posterior = \frac{apriori \text{ X } verosimilitud}{evidencia}$$

donde $P(A_i)$ define el conocimiento *a priori* del problema, es decir, la probabilidad absoluta de que suceda $A_i$ antes de saber cualquier cosa sobre $B$; la probablidad posterior $P(B|A_i)$ define la *verosimilitud*, es decir, qué tan probable es que suceda el evento $B$ dentro del espacio de trabajo reducido de $A_i$; luego, $P(B) = \sum_{k=1}^{n} P(A_k)P(B|A_k)$ es la *evidencia*, lo que indica cuál es la probabilidad de que ocurra $B$ si se tiene todo el conocimiento *a priori* de toda la partición y la verosimilitud de $B$ dentro de cada espacio de trabajo $A_k$ (usualmente sólo se ve como un factor de normalización); finalmente, $P(A_i|B)$ es el conocimiento posterior o consecuente de que suceda el evento $A_i$ dado que ocurrió $B$.

Usualmente, los eventos en el teorema de Bayes están expresados en términos de variables aleatorias y distribuciones de probabilidad, por lo que normalmente en la práctica el teorema toma la siguiente forma:

$$P(A_i|B) = \frac{P(A_i)p(B|A_i)}{P(B)} = \frac{P(A_i)p(B|A_i)}{\sum_{k=1}^{n} P(A_k)p(B|A_k)} \tag{2.1}$$

luego, para conocer la probabilidad absoluta $P(A_i|B)$ es necesario conocer todas las distribuciones de probabilidad asociadas al problema a resolver y todas las probabilidades *a priori*.

Lo anterior puede ser útil para reconocer patrones si las clases y los patrones se modelan como eventos o variables aleatorias. La idea general es la siguiente: un patrón (evento representado por una variable aleatoria vectorial $X$) pertenece a la clase $i$ (evento representado por la variable aleatoria $C_i$) si su probabilidad de pertenecer a esa clase es más grande que la probabilidad de pertenecer a las demás clases.

El clasificador Bayesiano toma ventaja de las probabilidades condicionales al sustituir, en el proceso de clasificación, el teorema de Bayes en la siguiente regla:

$$X \in C_i, \text{ si } P(C_i \mid X) > P(C_j \mid X)\ \forall i \neq j \tag{2.2}$$

cuya interpretación lógica es la siguiente: si se conoce que el patrón $X$ ocurrió (es decir que fue presentado al sistema), se calcula la probabilidad de que ocurra $C_k$ $\forall k = 1, 2, ..., n$ y se clasifica en la clase $C_i$ si dicha probabilidad es la mayor de todas, es decir, si la probabilidad de pertenencia de $X$ a $C_i$ es mayor a cualquier otra $C_j$.

Al usar el teorema de Bayes en 2.2, y tomando en cuenta que las probabilidades siempre son positivas, queda lo siguiente:

$$\begin{aligned} P(C_i \mid X) &> P(C_j \mid X) \\ \frac{P(C_i)\ p(X \mid C_i)}{P(X)} &> \frac{P(C_j)\ p(X \mid C_j)}{P(X)} \\ P(C_i)\ p(X \mid C_i) &> P(C_j)\ p(X \mid C_j) \end{aligned} \tag{2.3}$$

Ahora bien, dado que ln (la función logaritmo natural) es una función monótona creciente, es decir del tipo: $f(x) < f(y) \Leftrightarrow x < y$, se puede hacer la siguiente sustitución en la expresión 2.3:

$$\begin{aligned} \ln\left(P(C_i)\ p(X \mid C_i)\right) &> \ln\left(P(C_j)\ p(X \mid C_j)\right) \\ \ln(P(C_i)) + \ln(p(X \mid C_i)) &> \ln(P(C_j)) + \ln(p(X \mid C_j)) \\ d_i &> d_j, \text{ con } d_k = \ln(P(C_k)) + \ln(p(X \mid C_k)) \end{aligned} \tag{2.4}$$

donde $d_k$ define una función discriminante para el clasificador.

Así pues, que un patrón desconocido sea clasificado en una clase $C_i$ en particular, implica tener todo el conocimiento *a priori* de cada clase $C_i$ y su distribución de probabilidad correspondiente, lo que raramente sucede en la práctica [3].

Tomando en cuenta lo anterior, el algoritmo para diseñar un clasificador Bayesiano queda como sigue:

**Algoritmo 2.2** *Algoritmo del clasificador Bayesiano.*

1. Obtener una muestra representativa $S$ de los objetos a clasificar.
2. Determinar cada una de las clases $C_k$ que formarán parte del sistema.

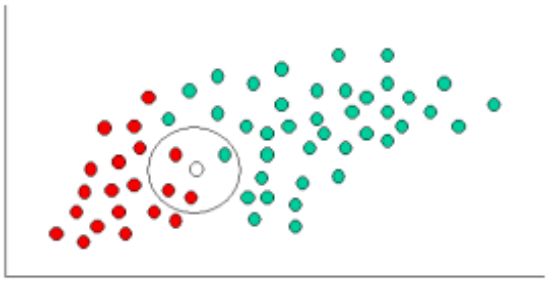


Figura 2.1: Clasificador Bayesiano. Ejemplo de clasificación del elemento blanco, dada la probabilidad de verde y rojo en su vecindad

3. Determinar, con base en la muestra y en la cardinalidad de cada clase, las probabilidades $P(C_k)$.

4. Determinar los rasgos útiles que se van a utilizar para clasificar, y elaborar cada distribución de probabilidad $p(X \mid C_k)$ la cual va a ser dependiente del número y naturaleza de cada rasgo de la variable aleatoria vectorial $X$.

5. Aplicar la siguiente regla para clasificar un patrón desconocido de entrada $X$, :

$$X \in C_i, \text{ si } d_i > d_j \ \forall i \neq j, \text{ con } d_k = \ln(P(C_k)) + \ln(p(X \mid C_k)) \tag{2.5}$$

■

En la figura 2.1 se puede ver un ejemplo de clasificación usando el clasificador bayesiano, tomada de [103].

Como se puede apreciar en los párrafos anteriores, este clasificador, si bien es muy robusto, también tiene la desventaja de que se debe tener una estadística muy amplia y completa sobre todas las variables aleatorias que forman parte del sistema. Entre más grande sea la muestra, y por tanto las mediciones estadísticas sobre ella, más confiable serán los resultados del clasificador, lo cual de alguna manera significa haber hecho el proceso de clasificación a mano durante mucho tiempo para poder tener una buena respuesta. Dado que esta situación pocas veces se presenta en la práctica, el uso del clasificador bayesiano se ve limitado, forzando a los investigadores en este campo a establecer condicines artificiales a las probabilidades condicionales de modo que sea funcional su uso.

## 2.2. Clasificador Euclideano

El funcionamiento básico de los clasificadores basados en métricas, entre los que se encuentra el Clasificador Euclideano [87], [25], [28], [104], [105] se muestra en el siguiente algoritmo:

**Algoritmo 2.3** *Algoritmo de clasificación de patrones basados en una métrica.*

1. Escoger una muestra de patrones clasificada de antemano en $n$ clases $\{C_1, C_2, .., C_n\}$ y una métrica $d$.
2. Con base en la muestra y para cada clase $C_i$, encontrar un patrón $v_i$ que la represente mejor.
3. Si $x$ es un patrón de dimensión $n$ cuya clase se desconoce, este patrón será clasificado en la clase $C_i$, si se cumple lo siguiente:

$$\forall j, j \neq i, d(x, v_i) \leq d(x, v_j)$$

■

Aquí aparece una primera debilidad de este enfoque, ya que el método que se utilice para decidir qué patrón es el mejor representante para una clase, y la métrica elegida, influirán en gran medida en el desempeño del clasificador.

Si en el algortimo anterior se especifica que la métrica a utilizar es la distancia euclideana, entonces se tiene el clasificador euclideano. Esta métrica es intuitivamente la que se utiliza para medir distancias, ya que equivale a medir el tamaño del segmento de recta que une a dos puntos y es la que normalmente se utiliza en geometría analítica y en análisis vectorial. La *distancia euclideana* entre dos vectores $\mathbf{x}$ y $\mathbf{y}$ de dimensión $n \in \mathbb{Z}^+$, con componentes $x_i$ y $y_i$ respectivamente, donde $i \in \{1, 2, \ldots, n\}$, se denota por $d_2(\mathbf{x}, \mathbf{y})$ y se calcula de la siguiente forma:

$$d_2(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^{n} (x_i - y_i)^2\right]^{\frac{1}{2}} \tag{2.6}$$

o en forma vectorial:

$$d_2(\mathbf{x}, \mathbf{y}) = \left[(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})\right]^{\frac{1}{2}} \tag{2.7}$$

A continuación se enuncia un primer algoritmo para el clasificador euclideano el cual es, en escencia, equivalente al algoritmo 2.3.

**Algoritmo 2.4** *Primer algoritmo del clasificador euclideano.*

1. Escoger una muestra de patrones clasificada de antemano en $p$ clases $\{C_1, C_2, .., C_p\}$.
2. Con base en la muestra y para cada clase $C_i$ , calcular el patrón representante $\mu_i = \frac{1}{k} \sum_{x_j \in C_i} x_j$, donde $k$ es el número de elementos en la muestra que pertenecen a $C_i$.

3. Sea $x$ un patrón de dimensión $n$ cuya clase se desconoce. Utilizando la distancia euclideana $d_2$, este patrón será clasificado en la clase $C_i$, si se cumple lo siguiente:

$$\forall j, j \neq i, d_2(x, \mu_i) \leq d_2(x, \mu_j)$$

■

Ahora bien, se puede reescribir el algoritmo del clasificador euclideano con base en el concepto de función discriminante, como sigue.

**Algoritmo 2.5** *Algoritmo del clasificador euclideano que usa funciones discriminantes.*

1. Escoger una muestra de patrones clasificada de antemano en $p$ clases $\{C_1, C_2, .., C_p\}$.
2. Con base en la muestra y para cada clase $C_i$ , calcular el patrón representante $\mu_i = \frac{1}{k} \sum_{x_j \in C_i} x_j$, donde $k$ es el número de elementos en la muestra que pertenecen a $C_i$ .
3. Generar funciones discriminantes $d_{ij}(x)$ para cada par de clases $C_i, C_j$, de forma que $d_{ij}(x) = (\mu_i - \mu_j)^T x - \frac{(\mu_i - \mu_j)^T(\mu_i + \mu_j)}{2}$.
4. Si $x$ es un patrón de dimensión $n$ cuya clase se desconoce, este patrón será clasificado en la clase $C_i$, si se cumple lo siguiente:

$$\forall j, j \neq i, d_{ij}(x) \geq 0$$

■

Así, el clasificador euclideano puede ser visto como un conjunto de funciones discriminantes, que ayudan a decidir si un patrón pertenece a una clase dada. Los patrones que hacen $d_{ij}$ igual a cero definen una *frontera* entre las dos clases $C_i$ y $C_j$. Un ejemplo de lo anterior está representado en la figura 2.2, tomada de [25].

**Nota 2.6** *El clasificador euclideano clasifica correctamente si las clases $C_i$ y $C_j$ son linealmente separables.*

El clasificador euclideano, aunque simple de implementar, posee limitaciones fuertes, como el hecho de que para se presente clasificación correcta, las clases tienen que ser linealmente separables, por lo que la aplicación de este modelo a problemas prácticos resulta muy reducida, y los investigadores se han visto en la necesidad de modificar sustancialmente el modelo para que sea funcional.

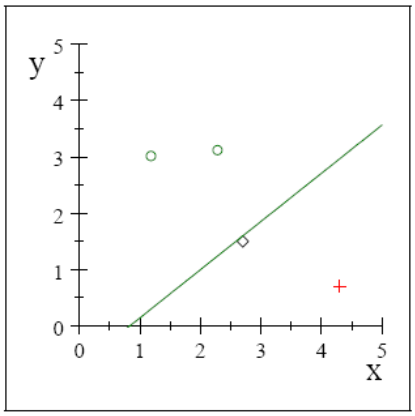



Figura 2.2: Clasificador Euclideano

## 2.3. Support Vector Machines (SVM)

El algoritmo utilizado por las SVMs funciona como un clasificador biclase que minimiza simultáneamente el error empírico de clasificación y maximiza algunas características de las métricas involucradas [106], [107]. Como tal, este algoritmo está fuertemente basado en la teoría de aprendizaje estadístico desarrollado por Vapnik, Chervonenkis y otros, que dio lugar a la implementación de las SVM durante la década de los noventa, en los Bell Laboratories de AT&T por Vapnik y sus colaboradores [88], [89].

El problema básico que ataca este modelo es el de de separar un hiperplano $n$-dimensional en dos clases, por medio de un hiperplano $n-1$-dimensional. Sin embargo, existen más de un hiperplano que logra este cometido. El objetivo de las SVM es encontrar el hiperplano óptimo que mejor generalice la clasificación. Para ello, se utiliza el concepto de *vector de soporte*, que se refiere a los patrones más cercanos al hiperplano buscado; dichos patrones se encuentran en la frontera de las clases. Para encontrar ese hiperplano óptimo, se maximiza la distancia a los vectores de soporte. De manera informal, se puede afirmar que los vectores de soporte son los patrones que proporcionan más información para la tarea de clasificación [3].

Ahora bien, puede darse el caso de que existan patrones, cercanos a la frontera de las clases, que se comportan más como excepciones. Si dichos patrones se tomaran como vectores de soporte, degradarían la generalización de la clasificación. Para solucionar este problema, se incluye cierto margen de error, con lo cual se admite que ciertos patrones cercanos a la frontera no sean tomados como vectores de soporte, siempre y cuando se mantengan a una distancia menor o igual al margen de error establecido. Entonces, el problema de encontrar el hiperplano que divide las dos clases de manera óptima, se resuelve maximizando la distancia entre dicho hiperplano y los patrones más cercanos a la frontera de las clases (vectores de soporte), a la vez que se minimiza el error [89], [108], [109]. Un ejemplo de SVM se puede apreciar en la figura 2.3.

En caso de que el problema no sea linealmente separable en el espacio original, se utiliza una transformación por medio de una función *kernel*, para llevar los patrones

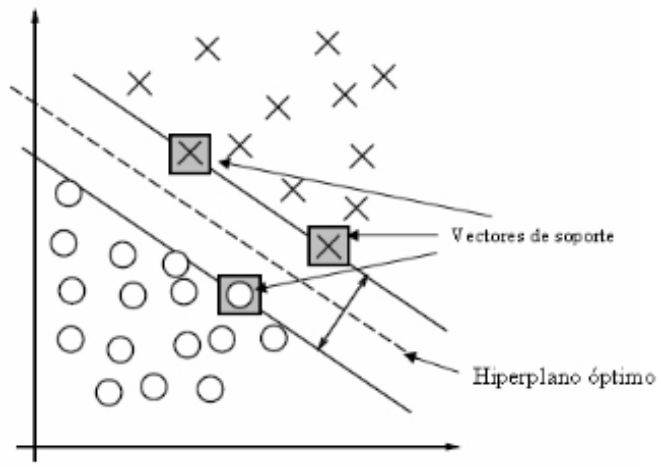



Figura 2.3: Support Vector Machine. Los vectores de soporte están indicados con un recuadro mientras que el hiperplano óptimo aparece como una línea punteada

a un espacio en una dimensión mayor, donde el problema sea linealmente separable [109].

## 2.4. k-Nearest Neighbor (k-NN)

Uno de los algoritmos de clasificación de patrones más notables es sin duda el del *k-Nearest Neighbor* [18]-[24], [26], [27], [72], [73], [110], [111]; a pesar de contar con cuatro décadas de existencia, sigue siendo uno de los más eficaces, como lo mostró Michie con los resultados del proyecto Statlog [13]. En la Figura 2.4 se puede ver un ejemplo de 5-NN, tomada de [3]. A continuación se describe el algoritmo $k$-NN, tanto para el caso de $k = 1$ como para $k > 1$.

**Algoritmo 2.7** *Algoritmo del k-NN para k = 1:*

1. Seleccionar la métrica a utilizar.
2. Calcular la distancia de un patrón $x$ por clasificar, a cada uno de los patrones del conjunto fundamental.
3. Obtener la distancia mínima.
4. Asignar a $x$ la clase del patrón con la mínima distancia.

■

**Algoritmo 2.8** *Algoritmo del k-NN para k > 1:*

1. Seleccionar la métrica a utilizar.

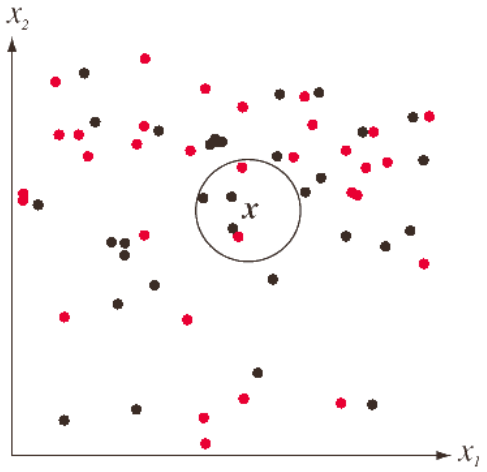



Figura 2.4: *k-Nearest Neighbor*. Para $k = 5$, el punto $x$ sería clasificado con la clase de los puntos negros (3 puntos negros contra 2 puntos rojos)

2. Calcular la distancia de un patrón $x$ por clasificar, a cada uno de los patrones del conjunto fundamental.

3. Ordenar los datos en orden ascendente.

4. Obtener los $k$ menores valores de distancia.

5. Usar la regla de *majority* para asignar la clase al patrón $x$.
   ■

No obstante que el $k$-NN es altamente eficaz, su gran desventaja es la baja eficiencia mostrada cuando se trabaja con un conjunto grande de patrones [18], [74]-[76].

## 2.5. Clasificador Híbrido Asociativo con Traslación (CHAT)

Este clasificador es históricamente el primer representante del enfoque asociativo, pues toma conceptos del área de memorias asociativas para llevar a cabo la tarea de clasificación de patrones [7], [52]. En particular, el CHAT toma principios de la *Lernmatrix* de Steinbuch y del *Linear Associator* de Anderson-Kohonen para realizar la tarea de clasificación, a pesar de que ambos modelos por separado tienen varias desventajas.

Por un lado, la Lernmatrix puede aceptar sólo patrones binarios como entradas, y se llega rápidamente a la saturación, fenómeno que impide la clasificación correcta. Por otro lado, no obstante que el Linear Associator elimina la restricción de patrones binarios a la entrada, puesto que puede aceptar patrones con valores reales en sus componentes, surge una restricción bastante fuerte: se requiere la ortonormalidad de los patrones de entrada para que la clasificación sea correcta.

La combinación de la fase de aprendizaje del Linear Associator y la de recuperación de la Lernmatrix dan como resultado el Clasificador Híbrido Asociativo (CHA), que presenta algunos problemas cuando la magnitud de los patrones pertecientes a una clase es considerablemente mayor a la magnitud de los patrones de las otras clases. Para subsanar este problema, el CHAT realiza una traslación de ejes previa al procesamiento de los patrones.

El algoritmo del CHAT es como sigue:

**Algoritmo 2.9** *Algoritmo del clasificador híbrido asociativo con traslación:*

1. Sea un conjunto fundamental de patrones de entrada de dimensión $n$ con valores reales en sus componentes (a la manera del Linear Associator), que se aglutinan en $m$ clases diferentes.
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 coordenada $k$-ésima, donde el valor es uno (a la manera de la Lernmatrix).
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 ejes coordenados.
5. Se realiza la traslación de todos los patrones del conjunto fundamental.
6. La fase de aprendizaje es similar a la del Linear Associator.
7. La fase de recuperación es similar a la que usa la Lernmatrix.
8. Se traslada todo patrón a clasificar a los nuevos ejes.
9. Se procede a clasificar los patrones desconocidos.

■

El autor del CHAT describe en su trabajo de tesis [52] una gran cantidad de experimentos donde se exhibe la superioridad del enfoque asociativo de clasificación de patrones, respecto de algunos clasificadores de actualidad.

En el Capítulo 5 del presente trabajo de tesis se presentarán los resultados de estudios experimentales realizados con bases de datos de prestigio, con objeto de comparar el desempeño de los clasificadores mostrados en diferentes publicaciones actuales, respecto del algoritmo de clasificador asociativo, tema central de esta tesis.

# Capítulo 3

# Materiales y Métodos

En este Capítulo se presentan, en cinco secciones, los conceptos, materiales y métodos que se requieren para describir y fundamentar teóricamente, tanto el algoritmo central de esta tesis que es desarrollado en el Capítulo 4, como el algoritmo alternativo incluido en el Apéndice B.

En la primera sección se describen los operadores $\alpha$ y $\beta$ y algunas de sus propiedades; estos operadores son tomados de los modelos asociativos Alfa-Beta y servirán de base para el nuevo algortimo. La sección 2 incluye una aportación original del autor de esta tesis: el operador $u_\beta$, que resulta útil en la formulación del nuevo algoritmo; se presentan ejemplos numéricos. Dos importantes conceptos matemáticos, módulo y congruencia, se incluyen en la sección 3, en virtud de que son inherentes al contenido relevante del siguiente Capítulo.

Dado que los vectores con que trabaja el nuevo algoritmo serán codificados con el código Johnson-Möbius modificado, su descripción forma parte de la sección 4, así como algunos ejemplos numéricos tomados directamente de la tesis donde originalmente se dio a conocer este código. Finalmente, la sección 5 está dedicada a cierto material teórico (los operadores de *cadena binaria mínima* y *k-binario de expansión*, junto con ejemplos numéricos) que, si bien no es utilizado en el Capítulo 4, sí es necesario para desarrollar el contenido del Apéndice B.

## 3.1. Operadores $\alpha$ y $\beta$

En esta sección se presentan los operadores $\alpha$ y $\beta$ junto con sus propiedades, algunas de las cuales serán usadas intensamente en la propuesta de esta tesis; el material es tomado directamente de [53]-[58], [60]-[62], [71].

Las memorias asociativas Alfa-Beta son de dos tipos ($max$ y $min$) y pueden operar en dos modos diferentes (autoasociativo y heretoasociativo). El operador $\alpha$ es utilizado en la fase de aprendizaje, mientras que el operador $\beta$ es útil durante la fase de recuperación. Estos dos operadores fueron definidos de manera tabular y sus propiedades demostradas en [53]; a continuación se incluyen las tablas que representan a los operadores $\alpha$ y $\beta$, dados los conjuntos $A = \{0, 1\}$ y $B = \{00, 01, 10\}$:

**Tabla 3.1**. Definición de los operadores Alfa y Beta

| $\alpha : A \times A \to B$ | | |
|---|---|---|
| $x$ | $y$ | $\alpha\,(x, y)$ |
| 0 | 0 | 01 |
| 0 | 1 | 00 |
| 1 | 0 | 10 |
| 1 | 1 | 01 |

| $\beta : B \times A \to A$ | | |
|---|---|---|
| $x$ | $y$ | $\beta\,(x, y)$ |
| 00 | 0 | 0 |
| 00 | 1 | 0 |
| 01 | 0 | 0 |
| 01 | 1 | 1 |
| 10 | 0 | 1 |
| 10 | 1 | 1 |

La operación binaria $\alpha$ exhibe algunas propiedades algebraicas, expuestas en la Tabla 3.2.

**Tabla 3.2**. Propiedades de la operación binaria $\alpha$

| | |
|---|---|
| $\alpha$1.- isoargumentos en $\alpha$ | $\alpha(x, x) = 1$ |
| $\alpha$2.- intercambio de argumentos en $\alpha$ | $(x \leq y) \leftrightarrow \alpha(x, y) \leq \alpha(y, x)$ |
| $\alpha$3.- $\alpha$ es creciente por la izquierda | $(x \leq y) \leftrightarrow [\alpha(x, z) \leq \alpha(y, z)]$ |
| $\alpha$4.- $\alpha$ es decreciente por la derecha | $(x \leq y) \leftrightarrow [\alpha(z, x) \geq \alpha(z, y)]$ |
| $\alpha$5.- $\alpha$ es distributiva por la derecha respecto al $\bigvee$ | $\alpha\left[\left(x \bigvee y\right), z\right] = \alpha(x, z) \bigvee \alpha(y, z)$ |
| $\alpha$6.- $\alpha$ es distributiva por la derecha respecto al $\bigwedge$ | $\alpha\left[\left(x \bigwedge y\right), z\right] = \alpha(x, z) \bigwedge \alpha(y, z)$ |

Asimismo, en la Tabla 3.3 se muestran algunas propiedades de la operación binaria $\beta$.

**Tabla 3.3**. Propiedades de la operación binaria $\beta$

| | |
|---|---|
| $\beta$1 - propiedad del 1 | $\beta(1, x) = x$ |
| $\beta$2 - isoargumentos en $\beta$ | $\beta(x, x) = x \quad \forall x \in A$ |
| $\beta$3.- $\beta$ creciente por la izquierda | $(x \leq y) \to [\beta(x, z) \leq \beta(y, z)]$ |
| $\beta$4.- $\beta$ creciente por la derecha | $(x \leq y) \to [\beta(z, x) \leq \beta(z, y)]$ |
| $\beta$5.- $\beta$ distributiva por la derecha respecto al $\bigvee$ | $\beta\left[\left(x \bigvee y\right), z\right] = \beta\,(x, z) \bigvee \beta\,(y, z)$ |
| $\beta$6.- $\beta$ distributiva por la derecha respecto al $\bigwedge$ | $\beta\left[\left(x \bigwedge y\right), z\right] = \beta\,(x, z) \bigwedge \beta\,(y, z)$ |
| $\beta$7.- $\beta$ distributiva por la izquierda respecto al $\bigvee$ | $\beta\left[x, \left(y \bigvee z\right)\right] = \beta\,(x, y) \bigvee \beta\,(x, z)$ |
| $\beta$8.- $\beta$ distributiva por la izquierda respecto al $\bigwedge$ | $\beta\left[x, \left(y \bigwedge z\right)\right] = \beta\,(x, y) \bigwedge \beta\,(x, z)$ |

Por otro lado, en la Tabla 3.4 se presentan las propiedades de la aplicación combinada de ambas operaciones $\alpha$ y $\beta$.

**Tabla 3.4**. Propiedades de la aplicación combinada de las operaciones $\alpha$ y $\beta$

| | |
|---|---|
| $\alpha\beta$1 - $\beta$ es la inversa de $\alpha$ por la derecha | $\beta\,[\alpha(x, y), y] = x$ |
| $\alpha\beta$2 - $\beta$ es la inversa de $\alpha$ por la izquierda | $\beta\,[\alpha(x, y), x] = x$ |
| $\alpha\beta$3 - isoargumentos en $\alpha$ como argumento de $\beta$ | $\beta\,[\alpha(x, x), y] = y$ |

## 3.2. Operador $u_\beta$

En esta sección se presenta el primer resultado original de esta tesis. El operador aquí descrito y ejemplificado fue creado por el autor y resulta de gran utilidad en el desarrollo, aplicación y prueba del algoritmo propuesto.

**Definición 3.1** *Sean: el conjunto* $A = \{0, 1\}$*, un número* $n \in \mathbb{Z}^+$ *y* $\mathbf{x} \in A^n$ *un vector binario de dimensión* $n$*, con la* $i$*-ésima componente representada por* $x_i$*. Se define el operador* $u_\beta(\mathbf{x})$ *de la siguiente manera:* $u_\beta(\mathbf{x})$ *tiene como argumento de entrada un vector binario* $n$*-dimensional* $\mathbf{x}$ *y la salida es un número entero no negativo que se calcula así:*

$$u_\beta(\mathbf{x}) = \sum_{i=1}^{n} \beta(x_i, x_i)$$

■

**Ejemplo 3.2** *Sea* $\mathbf{x} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$*; obtener* $u_\beta(\mathbf{x})$*.*

Dada la definición 3.1, $u_\beta(\mathbf{x}) = \sum_{i=1}^{5} \beta(\mathbf{x}_i, \mathbf{x}_i)$, por lo que $u_\beta(\mathbf{x}) = \beta(0,0)+\beta(1,1)+\beta(1,1)+\beta(0,0)+\beta(0,0) = 0+1+1+0+0 = 2$. Entonces, $u_\beta(\mathbf{x}) = 2$.
■

**Ejemplo 3.3** *Sea* $\mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$*; obtener* $u_\beta(\mathbf{y})$*.*

En este caso $u(\mathbf{y}) = \sum_{i=1}^{8} \beta(\mathbf{y}_i, \mathbf{y}_i)$, así que $u(\mathbf{y}) = \beta(1,1) + \beta(0,0) + \beta(0,0) + \beta(1,1) + \beta(1,1) + \beta(1,1) + \beta(1,1) + \beta(1,1) = 1+0+0+1+1+1+1+1 = 6$. Entonces, $u_\beta(\mathbf{y}) = 6$.
■

**Ejemplo 3.4** *Sea* $\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$*; obtener* $u_\beta(\mathbf{x})$ *y* $u_\beta(\mathbf{y})$*.*

Ahora $u_\beta(\mathbf{x}) = \beta(1,1) + \beta(1,1) + \beta(1,1) = 1 + 1 + 1 = 3$, mientras que $u_\beta(\mathbf{y}) = \beta(0,0) + \beta(0,0) + \beta(0,0) + \beta(0,0) = 0 + 0 + 0 + 0 = 0$.
■

## 3.3. Módulo y Congruencia

El contenido de esta sección fue tomado de [112]. Los conceptos de módulo y congruencia, junto con el operador de la sección anterior, son relevantes en el desarrollo de los conceptos originales desarrollados en esta tesis.

Hay situaciones en las que, al utilizar el operador de división acostumbrado, resulta de más interés el residuo (entero) de dicha operación que el resultado (fraccional) mismo. Para resolver este problema existe el operador *módulo*, denotado por mod.

**Definición 3.5** *Sean $a$ un número entero y $m$ un número entero positivo. Se denota por $a \bmod m$ al residuo de dividir $a$ por $m$.*
■

Dicho de otra manera, $a \bmod m$ es el número entero $r$ tal que $a = qm + r$ y $0 \leq r < m$.

**Ejemplo 3.6** *Se puede ver que $17 \bmod 5 = 2$, $-133 \bmod 9 = 2$ y $2001 \bmod 101 = 82$.*
■

Cuando dos números tienen el mismo residuo con respecto a un módulo $m$ dado, se dice que son congruentes.

**Definición 3.7** *Si $a$ y $b$ son números enteros y $m$ es un entero positivo, entonces $a$* es congruente con $b$ (módulo $m$) *si $(b-c)/m$ es un entero y se denota por $a \equiv b \bmod m$.*
■

Nótese que $a \equiv b \bmod m$ si y sólo si $a \bmod m = b \bmod m$.

**Ejemplo 3.8** *Determinar: a) si 17 es congruente con 6 módulo 6 y b) si 24 y 14 son congruentes módulo 6.*

a) Dado que 6 divide a 17-5=12, se tiene que $17 \equiv 5 \bmod 6$.

b) Por otro lado, dado que $24 - 14 = 10$ no es divisible por 6, se tiene que $24 \not\equiv 14 \bmod 6$.
■

## 3.4. Código Binario Johnson-Möbius Modificado

El contenido de esta sección está basado en [56] y [57]. Dado que los vectores con que trabaja el nuevo algoritmo serán codificados con el código Johnson-Möbius modificado, se describe éste en la presente sección; además, se muestran algunos ejemplos numéricos tomados directamente de la tesis donde originalmente se dio a conocer este código [56].

**Algoritmo 3.9** *Algoritmo del Código Johnson-Möbius Modificado:*

1. Sea un conjunto de números reales

$$\{r_1, r_2, \ldots, r_i, \ldots, r_n\}$$

donde $n$ es un número entero positivo fijo.

2. Si uno de los números del conjunto (por ejemplo $r_i$) es negativo, crear un nuevo conjunto transformado a través de la operación "restar $r_i$ a cada uno de los $n$ números"

$$\{t_1, t_2, \ldots, t_i, \ldots, t_n\}$$

donde $t_j = r_j - r_i \forall j \in \{1, 2, \ldots, n\}$ y particularmente $t_i = 0$. Nota: si hay más de un negativo, se trabaja con el menor.

3. Escoger un número fijo $d$ de decimales y truncar cada uno de los números del conjunto transformado (los cuales son no negativos) precisamente a $d$ decimales.

4. Realizar un escalamiento de $10d$ en el conjunto del paso 3, para obtener un conjunto de $n$ enteros no negativos

$$\{e_1, e_2, \ldots, e_i, \ldots, e_m, \ldots, e_n\}$$

donde $e_m$ es el número mayor.

5. El código Johnson-Möbius modificado para cada $j = 1, 2, \ldots, n$ se obtiene al generar $(e_m - e_j)$ ceros concatenados por la derecha con $e_j$ unos.

■

**Ejemplo 3.10** *Para representar los números* 3, 7, 11, 17 *y* 19 *con* 20 *bits usando el código Johnson-Möbius modificado, se tiene lo siguiente:*

1. Cada código tendrá 20 bits.
2. Para el número 3 se tendrán $20 - 3 = 17$ ceros seguidos de 3 unos.
3. Para el número 7 se tendrán $20 - 7 = 13$ ceros seguidos de 7 unos.
4. Para el número 11 se tendrán $20 - 11 = 9$ ceros seguidos de 11 unos.
5. Para el número 17 se tendrán $20 - 17 = 3$ ceros seguidos de 17 unos.
6. Para el número 19 se tendrán $20 - 19 = 1$ cero seguidos de 19 unos.

Los códigos correspondientes se muestran en la Tabla 3.5.

**Tabla 3.5**. Código Johnson-Möbius modificado para: 3, 7, 11, 17 y 19

| Número | Código Johnson-Möbius Modificado |
|---|---|
| 3 | 00000000000000000111 |
| 7 | 00000000000001111111 |
| 11 | 00000000011111111111 |
| 17 | 00011111111111111111 |
| 19 | 01111111111111111111 |

■

**Ejemplo 3.11** *Sea el conjunto* $r = \{0.2, 1.25, -0.3, 2.147\}$ $r \subset R$.

- Paso 1: $r = \{0.2, 1.25, -0.3, 2.147\}$
- Paso 2: Existe un número negativo (-0.3), por lo que se obtiene el conjunto transformado $t = \{0.5, 1.55, 0.0, 2.447\}$.
- Paso 3: Se escoge el número fijo $d = 1$ para obtener $t = \{0.5, 1.5, 0.0, 2.4\}$.
- Paso 4: Se realiza el escalamiento de $10d$ para obtener $e = \{5, 15, 0, 24\}$ donde $e_m = 24$.
- Paso 5: Para cada número $e_i$ del conjunto $e$, se generan $e_m - e_i$ ceros concatenados con $e_i$ unos. Los resultados se muestran en la siguiente Tabla.

**Tabla 3.6**. Ejemplos de códigos Johnson-Möbius modificado

| Número | Código Johnson-Möbius Modificado |
|---|---|
| 5 | 000000000000000000011111 |
| 15 | 000000000111111111111111 |
| 0 | 000000000000000000000000 |
| 24 | 111111111111111111111111 |

■

## 3.5. Operadores de Cadena Binaria Mínima y k-Binario de Expansión

Esta sección está dedicada a los operadores de *cadena binaria mínima* y *k-binario de expansión*, que, si bien no son utilizados en el cuerpo de la tesis, serán necesarios para desarrollar el contenido del Apéndice B.

El contenido presentado aquí se basa fuertemente en [113], tomándose textualmente las definiciones y los ejemplos numéricos de la tesis de maestría [85].

**Definición 3.12** *Sea $r$ un número entero no negativo. Se define el operador de cadena binaria mínima $k(r)$ de la siguiente manera: $k(r)$ tiene como argumento de entrada a $r$, y la salida es el menor de los logaritmos de base 2 de entre todos los números mayores que $r$ de la forma $2^k$, donde $k$ es un número entero positivo.*
■

**Ejemplo 3.13** *Calcular $k(6)$.*

De acuerdo con la notación de la Definición 1, se observa que $r = 6$, por lo que los primeros elementos del conjunto de números enteros mayores que 6 son 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, ... y así sucesivamente. Es claro que sólo algunos de estos números son de la forma $2^k$; algunos ejemplos, cuando $k$ toma los valores 3, 4, 5, 6, 7 y 8, son $8 = 2^3$, $16 = 2^4$, $32 = 2^5$, $64 = 2^6$, $128 = 2^7$ y $256 = 2^8$. Sin embargo, hay dos casos notables: por un lado, existen algunos valores de $k$ cuyos correspondientes números de la forma $2^k$ no están en el conjunto, y por otro, hay números en ese conjunto para los que no existe un número entero positivo $k$ que cumpla con que el número mencionado sea igual a $2^k$. Para el primer caso, se observa que el número que corresponde a $k = 2$ no se encuentra en este conjunto, dado que $2^2 = 4$ y 4 es menor que 6; y para el segundo caso, es posible notar que no existe un número $k$ que sea entero positivo y que cumpla con $10 = 2^k$. Ahora, al tomar los logaritmos de base 2 de los números que sí cumplen con las dos condiciones para este ejemplo, se tienen los siguientes valores: $log_2 8 = log_2 2^3 = 3$, $log_2 16 = log_2 2^4 = 4$, $log_2 32 = log_2 2^5 = 5$, $log_2 64 = log_2 2^6 = 6$, $log_2 128 = log_2 2^7 = 7$, $log_2 256 = log_2 2^8 = 8$,.$log_2 512 = log_2 2^9 = 9$,... así sucesivamente, siendo 3 el menor de todos. Es decir, $k(6) = 3$.
■

**Ejemplo 3.14** *Calcular: a) $k(255)$; b) $k(1025)$ y c) $k(64)$.*

a) Al considerar los números mayores que 255 se observa que el menor número que cumple las condiciones es $256 = 2^8$, por lo que $k(255) = 8$.

b) No obstante que hay un número muy cercano a 1025, el número 1024 que cumple con la condición de que existe $k = 10$ tal que $1024 = 2^{10}$, se observa que 1024 es menor que 1025, así que 10 no es la solución correcta. Se tiene que 2048 es el menor número entero positivo, de entre todos los mayores que 1025, para el cual existe un número entro positivo $k = 11$ tal que $2048 = 2^k$, lo cual significa que $log_2 2048 = log_2 2^{11} = 11$. La solución es $k(1025) = 11$.

c) Este es un caso curioso, porque $64 = 2^6$; sin embargo, $k(64) \neq 6$ simplemente porque 64 no es mayor que 64. Por ello, es necesario considerar los números mayores que 64 para los cuales se cumplan las condiciones de la Definición 1, y el más pequeño de todos es 128; dado que $128 = 2^7$, y además $log_2 128 = 7$, el resultado es $k(64) = 7$.
■

**Definición 3.15** *Sean: $r$ un número entero no negativo y $k$ un número entero positivo, los cuales cumplen con la expresión $k \geq k(r)$. Se define el operador k-binario de expansión $\mathcal{E}(r,k)$ de la siguiente manera: $\mathcal{E}(r,k)$ tiene como argumentos de entrada a $r$ y $k$, y la salida es un vector columna binario $k$-dimensional cuyas componentes corresponden a la expansión binaria de $k$ bits de $r$, con el bit menos significativo en la parte inferior.*

■

**Ejemplo 3.16** *Obtener $\mathcal{E}(6,3)$.*

En este caso $r = 6$ y $k = 3$ y de acuerdo con el Ejemplo 3.13 se tiene que $k(6) = 3$ por lo que se cumple la condición $k \geq k(r)$. Dado que la cadena binaria que representa al número 6 es 110, siendo 0 el bit menos significativo, el operador 3-binario de expansión para el número 6 produce el vector columna siguiente:

$$\mathcal{E}(6,3) = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

■

Obsérvese que el número de bits que se especificó en este operador coincide con el número de bits que se requieren para representar el número 6 en binario; es decir $k = k(r)$.

**Ejemplo 3.17** *Obtener $\mathcal{E}(6,4)$.*

En este caso $r = 6$ y $k = 4$ por lo que se cumple la condición $k \geq k(r)$. Sin embargo, a diferencia del Ejemplo 3.16, en este caso $k > k(r)$ y esto significa que 4 es mayor que el número de bits que requiere la representación binaria de 6, por lo que se hace necesario agregar un bit para lograr tener una cadena binaria de 4 bits; para ello, al considerar que la Definición 3.15 especifica que el bit menos significativo de la cadena binaria de $r$ debe estar en la parte inferior del vector columna, se coloca un cero en la componente inicial, y a continuación se coloca la cadena binaria que le corresponde al número 6:

$$\mathcal{E}(6,4) = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

■

**Ejemplo 3.18** *Obtener $\mathcal{E}(6,2)$.*

En este caso$r = 6$ y $k = 2$, y en virtud de que $k(6) = 3$, es claro que la condición $k \geq k(r)$ no se cumple. Por ello, la aplicación de la Definición 3.15 no tiene sentido, y esto significa que el vector que corresponde a $\mathcal{E}(6,2)$ no existe.

■

# Capítulo 4

# Algoritmo Propuesto

El algoritmo propuesto, parte fundamental de la presente tesis, es el tema central de este Capítulo. Sin embargo, antes de presentarlo, es necesario introducir un nuevo operador, que resulta de gran importancia para la formulación del algoritmo. Así pues, en la primera sección se presenta el operador $\gamma$ de similitud, junto con ejemplos numéricos, mientras que la sección 2 está dedicada al nuevo algoritmo clasificador de patrones.

## 4.1. Operador Gama de Similitud

El operador gama de similitud deriva su nombre de sus características: en primer lugar, está basado fuertemente en la operaciones $\alpha$ y $\beta$ de las memorias asociativas Alfa-Beta, aunque de una manera novedosa; por ello, se opta por nombrarle gama, siendo el nombre de la letra griega que está a continuación de las primeras dos, $\alpha$ y $\beta$. Por otro lado, este operador indica si dos vectores son parecidos o no, dado un grado de disimilitud $\theta$; por ello, este operador es de similitud. En este sentido, el argumento $\theta$ indica la tolerancia para que dos vectores, al compararlos, sean considerados similares, no obstante que son diferentes. A continuación se define el operador gama de similitud.

**Definición 4.1** *Sean: el conjunto* $A = \{0, 1\}$*, un número* $n \in \mathbb{Z}^+$*,* $\mathbf{x} \in A^n$ *y* $\mathbf{y} \in A^n$ *dos vectores binarios n-dimensionales, con la i-ésima componente representada por* $x_i$ *y* $y_i$*, respectivamente, y además,* $\theta$ *un número entero no negativo. Se define el operador gama de similitud* $\gamma(\mathbf{x}, \mathbf{y}, \theta)$ *de la siguiente manera:* $\gamma(\mathbf{x}, \mathbf{y}, \theta)$ *tiene como argumentos de entrada dos vectores binarios n-dimensionales* $\mathbf{x}$ *y* $\mathbf{y}$*, y un número entero no negativo* $\theta$*, y la salida es un número binario que se calcula así:*

$$\gamma(\mathbf{x}, \mathbf{y}, \theta) = \begin{cases} 1 & si\ n - u_\beta\left[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2\right] \leq \theta \\ 0 & en\ otro\ caso \end{cases}$$

■

**Ejemplo 4.2** *Sean* $\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ , $\mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ *y* $\theta = 3$*; calcular* $\gamma(\mathbf{x}, \mathbf{y}, \theta)$.

En este caso se puede observar que $n = 8$. Al calcular $\alpha(\mathbf{x}, \mathbf{y})$ se obtiene $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \\ 0 \\ 2 \\ 2 \\ 1 \end{pmatrix}$; calculado el módulo 2 de cada componente, se obtiene el vector $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$; ahora bien, $u_\beta \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 2$, que al restarlo de $n = 8$ da como resultado $8 - 2 = 6$; pero $6 > 3$, por lo que $6 \nleq 3$, entonces $\gamma(\mathbf{x}, \mathbf{y}, \theta) = 0$.

■

**Ejemplo 4.3** *Sean* $\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ , $\mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ *y* $\theta = 5$*; calcular* $\gamma(\mathbf{x}, \mathbf{y}, \theta)$.

Se puede observar que para este caso $n = 10$. Luego, $\alpha(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}$,

$\begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} \bmod 2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $u_\beta \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 5$, $10 - 5 = 5$, y como $5 = 5$ entonces $5 \leq 5$, por lo que $\gamma(\mathbf{x}, \mathbf{y}, \theta) = 1$.

■

**Ejemplo 4.4** *Sean* $\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ *y* $\theta = 3$*; calcular* $\gamma(\mathbf{x}, \mathbf{y}, \theta)$.

Ahora se tiene que $n = 5$. Entonces $\alpha(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix} \bmod 2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$,

$u_\beta \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = 3$, $5 - 3 = 2$, y como $2 < 3$, entonces $2 \leq 3$, y el resultado es $\gamma(\mathbf{x}, \mathbf{y}, \theta) = 1$.

■

**Ejemplo 4.5** *Sean* $\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ *y* $\theta = 0$*; calcular* $\gamma(\mathbf{x}, \mathbf{y}, \theta)$.

Para este caso se tiene que $n = 3$. Así, $\alpha(\mathbf{x}, \mathbf{y}) = \begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix} \bmod 2 = \begin{pmatrix}1\\1\\1\end{pmatrix}$, $u_\beta \begin{pmatrix}1\\1\\1\end{pmatrix} = 3$, $3 - 3 = 0$, y como $0 \leq 0$ puesto que $0 = 0$, el resultado es $\gamma(\mathbf{x}, \mathbf{y}, \theta) = 1$.
■

**Nota 4.6** *Nótese que* $\gamma(\mathbf{x}, \mathbf{y}, 0) = 1 \longleftrightarrow \mathbf{x} = \mathbf{y}$.

## 4.2. Nuevo Algoritmo Clasificador de Patrones

El algoritmo propuesto, tema central de la presente tesis, hace uso de los operadores $\alpha$ y $\beta$ y sus propiedades, así como del operador $\gamma$ de similitud (que a su vez requiere del operador $u_\beta$ y del concepto de módulo) y sus propiedades cuando se utiliza con patrones codificados con el código Johnson-Möbius modificado, para llevar a cabo de manera eficaz y eficiente la tarea de clasificación de patrones. Es por ello que este algoritmo de denomina Clasificador Gama. A continuación se presenta dicho algoritmo.

**Algoritmo 4.7** *Sean los números* $k, m, n, p \in \mathbb{Z}^+$, $\{\mathbf{x}^\mu \,|\mu = 1, 2, \ldots, p\}$ *el conjunto fundamental de patrones de cardinalidad* $p$, *donde* $\forall \mu\, \mathbf{x}^\mu \in \mathbb{R}^n$ *y* $\mathbf{y} \in \mathbb{R}^n$ *un vector real* $n$*-dimensional a ser clasificado. Se asume que el conjunto fundamental está particionado en* $m$ *clases diferentes, donde cada clase tiene cardinalidad* $k_i$, $i = 1, 2, \ldots, m$, *por lo que* $\sum_{i=1}^{m} k_i = p$. *Para clasificar el patrón* $\mathbf{y}$, *se realiza lo siguiente:*

1. Codificar cada componente de cada patrón del conjunto fundamental con el código Johnson-Möbius modificado $\forall \mu \forall j\, x_j^\mu$; para ello, se calcula la magnitud de $e_m$ del algoritmo 3.9, equivalente al número de bits necesarios para la codificación. Al contrario de lo que normalmente se hace (codificar las componentes de un patrón con el mismo $e_m$ y concatenarlas), aquí se consideran las componentes de todos lo patrones del conjunto fundamental que tienen el mismo índice, en cada uno de los pasos de la codificación usando el código Johnson-Möbius modificado; es decir, para cada $j = 1, 2, \ldots, n$ se calcula

$$e_m(j) = \bigvee_{i=1}^{p} x_j^i$$

una vez que dichas componentes $x_j^i$ han sido transformadas y escaladas, de modo que todos los valores sean enteros no negativos, en caso de que se requiera. Así, la componente $x_j^i$ se transforma en un vector binario de dimensión $e_m(j)$.

2. Calcular el parámetro de paro $\rho = \bigwedge_{j=1}^{n} e_m(j)$.

3. Codificar cada componente del patrón a clasificar con el código Johnson-Möbius modificado, utilizando las mismas condiciones que se utilizaron para codificar las componentes de los patrones fundamentales; es decir, para cada índice de componente, se utilizan los mismos valores de $r_i$, $d$ y $e_m$ que se describen en el Algoritmo 3.9.

4. Realizar una transformación de índices en los patrones del conjunto fundamental, de manera que el índice único que tenía un patrón originalmente en el conjunto fundamental, por ejemplo $\mathbf{x}^{\mu}$, se convierta en dos índices: uno para la clase (por ejemplo la clase $i$) y otro para el orden que le corresponde a ese patrón dentro de esa clase (por ejemplo $\omega$). Bajo estas condiciones ejemplificadas, la notación para el patrón $\mathbf{x}^{\mu}$ será ahora, con la transformación, $\mathbf{x}^{i\omega}$. Lo anterior se realiza para todos los patrones del conjunto fundamental.

5. Inicializar $\theta$ a 0.

6. Realizar la operación $\gamma\left(\mathbf{x}_j^{i\omega}, \mathbf{y}_j, \theta\right)$ para cada clase y para cada componente de cada uno de los patrones fundamentales que corresponden a esa clase, y del patrón a clasificar:

$$\gamma\left(\mathbf{x}_j^{i\omega}, \mathbf{y}_j, \theta\right) \text{ donde } i = 1, 2, \ldots, m,\ \omega = 1, 2, \ldots, k_i,\ j = 1, 2, \ldots, n$$

7. Calcular la suma ponderada $c_i$ de los resultados obtenidos en el paso 6, para cada clase $i = 1, 2, \ldots, m$:

$$c_i = \frac{\sum_{\omega=1}^{k_i} \sum_{j=1}^{n} \gamma\left(\mathbf{x}_j^{i\omega}, \mathbf{y}_j, \theta\right)}{k_i}$$

8. Si existe más de un máximo entre las sumas ponderadas por clase, incrementar $\theta$ en 1 y repetir los pasos 6 y 7 hasta que exista un máximo único, o se cumpla con la condición de paro $\theta \geq \rho$.

9. Si existe un máximo único, asignar al patrón a clasificar la clase correspondiente a ese máximo:

$$C_y = C_j \text{ tal que } \bigvee_{i=1}^{m} c_i = c_j$$

10. En caso contrario: si $\lambda$ es el índice más pequeño de clase que corresponde a uno de los máximos, asignar al patrón a clasificar la clase $C_\lambda$.

■

**Ejemplo 4.8** *Sean los patrones fundamentales* $\mathbf{x}^1 = \begin{pmatrix}2\\8\end{pmatrix}$, $\mathbf{x}^2 = \begin{pmatrix}1\\9\end{pmatrix}$, $\mathbf{x}^3 = \begin{pmatrix}6\\3\end{pmatrix}$, $\mathbf{x}^4 = \begin{pmatrix}7\\4\end{pmatrix}$, *agrupados en dos clases* $C_1 = \{\mathbf{x}^1, \mathbf{x}^2\}$, $C_2 = \{\mathbf{x}^3, \mathbf{x}^4\}$; *clasificar los patrones* $\mathbf{y}^1 = \begin{pmatrix}6\\2\end{pmatrix}$, $\mathbf{y}^2 = \begin{pmatrix}3\\9\end{pmatrix}$.

Se observa que, para este caso, la dimensión de los patrones es $n = 2$, y se tienen $m = 2$ clases, donde las clases agrupan $k_1 = 2$ y $k_2 = 2$ patrones, respectivamente. Entonces, siguiendo los pasos del algoritmo:

1. Se codifican las componentes de los patrones fundamentales, resultando que para la primera componente, $e_m(1) = 7$ y para la segunda $e_m(2) = 9$. Los patrones fundamentales codificados quedan como sigue.

$$\mathbf{x}^1 = \begin{pmatrix} 0000011 \\ 011111111 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0000001 \\ 111111111 \end{pmatrix}, \mathbf{x}^3 = \begin{pmatrix} 0111111 \\ 000000111 \end{pmatrix}, \mathbf{x}^4 = \begin{pmatrix} 1111111 \\ 000001111 \end{pmatrix}$$

2. Puesto que $e_m(1) = 7$ y $e_m(2) = 9$, el parámetro de paro es en este caso:

$$\rho = \bigwedge_{j=1}^{2} e_m(j) = 7$$

3. Usando las mismas $e_m(j)$, los patrones a clasificar quedan codificados como sigue.

$$\mathbf{y}^1 = \begin{pmatrix} 0111111 \\ 000000011 \end{pmatrix}, \mathbf{y}^2 = \begin{pmatrix} 0000111 \\ 111111111 \end{pmatrix}$$

4. Los patrones fundamentales se agrupan por sus clases, $C_1 = \{\mathbf{x}^1, \mathbf{x}^2\}$, $C_2 = \{\mathbf{x}^3, \mathbf{x}^4\}$, por lo que $\mathbf{x}^1 = \mathbf{x}^{11}$, $\mathbf{x}^2 = \mathbf{x}^{12}$; $\mathbf{x}^3 = \mathbf{x}^{21}$, $\mathbf{x}^4 = \mathbf{x}^{22}$.

5. $\theta = 0$.

Pasos restantes del algoritmo para la clasificación de $\mathbf{y}^1$:

6. $\gamma\left(\mathbf{x}_1^{11}, \mathbf{y}_1^1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{11}, \mathbf{y}_2^1, 0\right) = 0$; $\gamma\left(\mathbf{x}_1^{12}, \mathbf{y}_1^1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{12}, \mathbf{y}_2^1, 0\right) = 0$;
   $\gamma\left(\mathbf{x}_1^{21}, \mathbf{y}_1^1, 0\right) = 1$, $\gamma\left(\mathbf{x}_2^{21}, \mathbf{y}_2^1, 0\right) = 0$; $\gamma\left(\mathbf{x}_1^{22}, \mathbf{y}_1^1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{22}, \mathbf{y}_2^1, 0\right) = 0$;

7. Se suman lo resultados anteriores, obteniéndose: $c_1 = \frac{0+0+0+0}{2} = \frac{0}{2}$, $c_2 = \frac{1+0+0+0}{2} = \frac{1}{2}$.

8. Con lo anterior se obtiene un máximo único: $\bigvee_{i=1}^{2} c_i = c_2 = \frac{1}{2}$.

9. Entonces, a $\mathbf{y}^1$ se le asigna la clase $C_2$.

Pasos restantes del algoritmo para la clasificación de $\mathbf{y}^2$:

6. $\gamma\left(\mathbf{x}_1^{11}, \mathbf{y}_1^2, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{11}, \mathbf{y}_2^2, 0\right) = 0$; $\gamma\left(\mathbf{x}_1^{12}, \mathbf{y}_1^2, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{12}, \mathbf{y}_2^2, 0\right) = 1$;
   $\gamma\left(\mathbf{x}_1^{21}, \mathbf{y}_1^2, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{21}, \mathbf{y}_2^2, 0\right) = 0$; $\gamma\left(\mathbf{x}_1^{22}, \mathbf{y}_1^2, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{22}, \mathbf{y}_2^2, 0\right) = 0$.

7. Se suman lo resultados anteriores, obteniéndose: $c_1 = \frac{0+0+0+1}{2} = \frac{1}{2}$, $c_2 = \frac{0+0+0+0}{2} = \frac{0}{2}$.

8. Con lo anterior se obtiene un máximo único: $\bigvee_{i=1}^{2} c_i = c_1 = \frac{1}{2}$.

9. Entonces, a $\mathbf{y}^2$ se le asigna la clase $C_1$

■

**Nota 4.9** *Los siguientes dos Ejemplos, el 4.10 y 4.11, están tomados de la base de datos Iris Plant [114], una base de datos clásica en el área de Reconocimiento de Patrones, misma que será abordada a mayor detalle en el Capítulo 5. En particular, la clase $C_1$ corresponde a la clase* Iris-setosa*, mientras que la clase $C_2$ es la clase* Iris-versicolor *y la clase $C_3$ es la clase* Iris-virginica*. El patrón a clasificar $\mathbf{y}$ del Ejemplo 4.10 pertenece a la clase $C_1$ (*Iris-setosa*), mientras que los patrones $\mathbf{y}^1$, $\mathbf{y}^2$ y $\mathbf{y}^3$ del Ejemplo 4.11 pertenecen a las clases $C_2$ (*Iris-versicolor*), $C_1$ (*Iris-setosa*) y $C_3$ (*Iris-virginica*), respectivamente.*

**Ejemplo 4.10** *Sean los patrones fundamentales* $\mathbf{x}^1 = \begin{pmatrix} 4.8 \\ 3.4 \\ 1.6 \\ 0.2 \end{pmatrix}$, $\mathbf{x}^2 = \begin{pmatrix} 4.4 \\ 3.0 \\ 1.3 \\ 0.2 \end{pmatrix}$, $\mathbf{x}^3 = \begin{pmatrix} 6.4 \\ 2.9 \\ 4.3 \\ 1.3 \end{pmatrix}$, $\mathbf{x}^4 = \begin{pmatrix} 6.0 \\ 2.7 \\ 5.1 \\ 1.6 \end{pmatrix}$, $\mathbf{x}^5 = \begin{pmatrix} 6.3 \\ 3.3 \\ 6.0 \\ 2.5 \end{pmatrix}$, $\mathbf{x}^6 = \begin{pmatrix} 5.8 \\ 2.7 \\ 5.1 \\ 1.9 \end{pmatrix}$, *agrupados en tres clases* $C_1 = \{\mathbf{x}^1, \mathbf{x}^2\}$, $C_2 = \{\mathbf{x}^3, \mathbf{x}^4\}$ *y* $C_3 = \{\mathbf{x}^5, \mathbf{x}^6\}$*; clasificar el patrón* $\mathbf{y} = \begin{pmatrix} 4.6 \\ 3.4 \\ 1.4 \\ 0.3 \end{pmatrix}$.

En este caso se observa que $n = 4$, $m = 3$, $k_1 = 2$, $k_2 = 2$ y $k_3 = 2$. Una vez convertidos al código Johnson-Möebius modificado con las $e_m(j)$ correspondientes a cada componente $e_m(1) = 64$, $e_m(2) = 34$, $e_m(3) = 60$ y $e_m(4) = 25$ los patrones quedan como sigue.

$$\mathbf{x}^1 = \begin{pmatrix} 0000000000000000111111111111111111111111111111111111111111111111 \\ 1111111111111111111111111111111111 \\ 000000000000000000000000000000000000000000001111111111111111 \\ 00000000000000000000011 \end{pmatrix}$$

$$\mathbf{x}^2 = \begin{pmatrix} 0000000000000000000011111111111111111111111111111111111111111111 \\ 0000111111111111111111111111111111 \\ 000000000000000000000000000000000000000000001111111111111 \\ 00000000000000000000011 \end{pmatrix}$$

$$\mathbf{x}^3 = \begin{pmatrix} 1111111111111111111111111111111111111111111111111111111111111111 \\ 0000011111111111111111111111111111 \\ 000000000000000001111111111111111111111111111111111111111111 \\ 00000000000011111111111111 \end{pmatrix}$$

$$\mathbf{x}^4 = \begin{pmatrix} 000001111111111111111111111111111111111111111111111111111111111111 \\ 000000011111111111111111111111111111 \\ 000000000111111111111111111111111111111111111111111111111111 \\ 000000000111111111111111 \end{pmatrix}$$

$$\mathbf{x}^5 = \begin{pmatrix} 011111111111111111111111111111111111111111111111111111111111111111 \\ 011111111111111111111111111111111 \\ 111111111111111111111111111111111111111111111111111111111111 \\ 111111111111111111111111 \end{pmatrix}$$

$$\mathbf{x}^6 = \begin{pmatrix} 000000111111111111111111111111111111111111111111111111111111111111 \\ 000000011111111111111111111111111111 \\ 000000000111111111111111111111111111111111111111111111111111 \\ 000000111111111111111111 \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} 000000000000000000011111111111111111111111111111111111111111111111 \\ 111111111111111111111111111111111 \\ 000000000000000000000000000000000000000000000011111111111111 \\ 000000000000000000000111 \end{pmatrix}$$

$\rho = \bigwedge_{j=1}^{4} e_m(j) = 25$, mientras que $\mathbf{x}^1 = \mathbf{x}^{11}$, $\mathbf{x}^2 = \mathbf{x}^{12}$; $\mathbf{x}^3 = \mathbf{x}^{21}$, $\mathbf{x}^4 = \mathbf{x}^{22}$; $\mathbf{x}^5 = \mathbf{x}^{31}$, $\mathbf{x}^6 = \mathbf{x}^{32}$.

Para $\theta = 0$ :

$\gamma\left(\mathbf{x}_1^{11}, \mathbf{y}_1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{11}, \mathbf{y}_2, 0\right) = 1$, $\gamma\left(\mathbf{x}_3^{11}, \mathbf{y}_3, 0\right) = 0$, $\gamma\left(\mathbf{x}_4^{11}, \mathbf{y}_4, 0\right) = 0$;
$\gamma\left(\mathbf{x}_1^{12}, \mathbf{y}_1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{12}, \mathbf{y}_2, 0\right) = 0$, $\gamma\left(\mathbf{x}_3^{12}, \mathbf{y}_3, 0\right) = 0$, $\gamma\left(\mathbf{x}_4^{12}, \mathbf{y}_4, 0\right) = 0$;
$\gamma\left(\mathbf{x}_1^{21}, \mathbf{y}_1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{21}, \mathbf{y}_2, 0\right) = 0$, $\gamma\left(\mathbf{x}_3^{21}, \mathbf{y}_3, 0\right) = 0$, $\gamma\left(\mathbf{x}_4^{21}, \mathbf{y}_4, 0\right) = 0$;
$\gamma\left(\mathbf{x}_1^{22}, \mathbf{y}_1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{22}, \mathbf{y}_2, 0\right) = 0$, $\gamma\left(\mathbf{x}_3^{22}, \mathbf{y}_3, 0\right) = 0$, $\gamma\left(\mathbf{x}_4^{22}, \mathbf{y}_4, 0\right) = 0$;
$\gamma\left(\mathbf{x}_1^{31}, \mathbf{y}_1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{31}, \mathbf{y}_2, 0\right) = 0$, $\gamma\left(\mathbf{x}_3^{31}, \mathbf{y}_3, 0\right) = 0$, $\gamma\left(\mathbf{x}_4^{31}, \mathbf{y}_4, 0\right) = 0$;
$\gamma\left(\mathbf{x}_1^{32}, \mathbf{y}_1, 0\right) = 0$, $\gamma\left(\mathbf{x}_2^{32}, \mathbf{y}_2, 0\right) = 0$, $\gamma\left(\mathbf{x}_3^{32}, \mathbf{y}_3, 0\right) = 0$, $\gamma\left(\mathbf{x}_4^{32}, \mathbf{y}_4, 0\right) = 0$; por lo que $c_1 = \frac{1}{2}$, $c_2 = \frac{0}{2}$ y $c_3 = \frac{0}{2}$, con lo que se obtiene un máximo único $\bigvee_{i=1}^{3} c_i = c_1 = \frac{1}{2}$. Entonces, a $\mathbf{y}$ se le asigna la clase $C_1$. De acuerdo con lo mencionado en la Nota 4.9, este resultado es correcto.

■

**Ejemplo 4.11** *Usando el conjunto fundamental del Ejemplo 4.10, clasificar los patrones* $\mathbf{y}^1 = \begin{pmatrix} 6.2 \\ 2.8 \\ 4.7 \\ 1.2 \end{pmatrix}$, $\mathbf{y}^2 = \begin{pmatrix} 4.7 \\ 3.2 \\ 1.6 \\ 0.2 \end{pmatrix}$ *y* $\mathbf{y}^3 = \begin{pmatrix} 5.7 \\ 2.5 \\ 5.0 \\ 2.0 \end{pmatrix}$.

Los patrones fundamentales codificados son los mismos que los del Ejemplo 4.10, mientras que los patrones a clasificar ya codificados (con los mismos valores de $e_m(j)$

usados para codificar los patrones fundamentales) quedan como sigue:

$$\mathbf{y}^1 = \begin{pmatrix} 00111111111111111111111111111111111111111111111111111111111111111111 \\ 00000011111111111111111111111111111 \\ 00000000000001111111111111111111111111111111111111111111111111111111 \\ 000000000000011111111111 \end{pmatrix}$$

$$\mathbf{y}^2 = \begin{pmatrix} 00000000000000000111111111111111111111111111111111111111111111111111 \\ 00111111111111111111111111111111 \\ 000000000000000000000000000000000000000000001111111111111111 \\ 000000000000000000000011 \end{pmatrix}$$

$$\mathbf{y}^3 = \begin{pmatrix} 00000001111111111111111111111111111111111111111111111111111111111111 \\ 00000000011111111111111111111111 \\ 00000000001111111111111111111111111111111111111111111111111111 \\ 0000011111111111111111111 \end{pmatrix}$$

- Para clasificar $\mathbf{y}^1$:

Con $\theta = 0$, $c_1 = \frac{0}{2}$, $c_2 = \frac{0}{2}$ y $c_3 = \frac{0}{2}$, y como el máximo no es único y además $\theta < \rho$ (calculado en el Ejemplo 4.10 como $\rho = 25$), se incrementa $\theta$ y se repiten los pasos 6 y 7.

Con $\theta = 1$, $c_1 = \frac{0}{2}$, $c_2 = \frac{4}{2}$ y $c_3 = \frac{1}{2}$, con lo que se obtiene un máximo único $\bigvee_{i=1}^{3} c_i = c_2 = \frac{4}{2}$; entonces, a $\mathbf{y}^1$ se le asigna la clase $C_2$.

- Ahora bien, para clasificar $\mathbf{y}^2$:

Con $\theta = 0$, $c_1 = \frac{3}{2}$, $c_2 = \frac{0}{2}$ y $c_3 = \frac{0}{2}$, con lo que se obtiene un máximo único $\bigvee_{i=1}^{3} c_i = c_1 = \frac{3}{2}$; entonces, a $\mathbf{y}^2$ se le asigna la clase $C_1$.

- Por último, para clasificar $\mathbf{y}^3$:

Con $\theta = 0$, $c_1 = \frac{0}{2}$, $c_2 = \frac{0}{2}$ y $c_3 = \frac{0}{2}$, y como el máximo no es único y además $\theta < \rho$ (calculado en el Ejemplo 4.10 como $\rho = 25$), se incrementa $\theta$ y se repiten los pasos 6 y 7.

Con $\theta = 1$, $c_1 = \frac{0}{2}$, $c_2 = \frac{1}{2}$ y $c_3 = \frac{3}{2}$, con lo que se obtiene un máximo único $\bigvee_{i=1}^{3} c_i = c_3 = \frac{3}{2}$; entonces, a $\mathbf{y}^3$ se le asigna la clase $C_3$.

De acuerdo con lo mencionado en la Nota 4.9, estos resultados de clasificación son correctos.

■

**Ejemplo 4.12** *Sea el conjunto fundamental integrado por los patrones* $\mathbf{x}^1 = \begin{pmatrix} -8 \\ 6 \\ 4 \\ 2 \end{pmatrix}$, $\mathbf{x}^2 = \begin{pmatrix} -11 \\ 10 \\ 0 \\ -1 \end{pmatrix}$, $\mathbf{x}^3 = \begin{pmatrix} 3 \\ 15 \\ 15 \\ 10 \end{pmatrix}$, $\mathbf{x}^4 = \begin{pmatrix} -7 \\ 7 \\ 3 \\ 0 \end{pmatrix}$, $\mathbf{x}^5 = \begin{pmatrix} 5 \\ 18 \\ 10 \\ 5 \end{pmatrix}$, *agrupados en dos clases* $C_1 = \{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^4\}$ *y* $C_2 = \{\mathbf{x}^3, \mathbf{x}^5\}$*; clasificar el patrón* $\mathbf{y} = \begin{pmatrix} 0 \\ 23 \\ 7 \\ 7 \end{pmatrix}$.

En este caso se observa que $n = 4$, $m = 2$, $k_1 = 3$ y $k_2 = 2$. El primer paso indica que se debe codificar todos los patrones fundamentales con el código Johnson-Möebius modificado; pero como en este caso se tienen componentes con valores negativos, es necesario realizar primero una transformación a los patrones. Así, los patrones transformados quedan como sigue:

$$\mathbf{x}^1 = \begin{pmatrix} 3 \\ 6 \\ 4 \\ 3 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0 \\ 10 \\ 0 \\ 0 \end{pmatrix}, \mathbf{x}^3 = \begin{pmatrix} 14 \\ 15 \\ 15 \\ 11 \end{pmatrix}, \mathbf{x}^4 = \begin{pmatrix} 4 \\ 7 \\ 3 \\ 1 \end{pmatrix}, \mathbf{x}^5 = \begin{pmatrix} 16 \\ 18 \\ 10 \\ 6 \end{pmatrix}$$

Entonces, los valores $e_m(j)$ para $j = 1, 2, 3, 4$ son, respectivamente: 16, 18, 15, 11; y los patrones ya codificados quedan como sigue:

$$\mathbf{x}^1 = \begin{pmatrix} 0000000000000111 \\ 000000000000111111 \\ 000000000001111 \\ 00000000111 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0000000000000000 \\ 000000001111111111 \\ 000000000000000 \\ 00000000000 \end{pmatrix}$$

$$\mathbf{x}^3 = \begin{pmatrix} 0011111111111111 \\ 000111111111111111 \\ 111111111111111 \\ 11111111111 \end{pmatrix}, \mathbf{x}^4 = \begin{pmatrix} 0000000000001111 \\ 000000000001111111 \\ 000000000000111 \\ 00000000001 \end{pmatrix}$$

$$\mathbf{x}^5 = \begin{pmatrix} 1111111111111111 \\ 111111111111111111 \\ 000001111111111 \\ 00000111111 \end{pmatrix}$$

El segundo paso, por su parte, indica que se deben codificar los patrones a clasificar. Como los patrones fundamentales fueron transformados, también $\mathbf{y}$ debe ser transformado, quedando así:

$$\mathbf{y} = \begin{pmatrix} 11 \\ 23 \\ 7 \\ 8 \end{pmatrix}$$

Y aquí surge un problema. Como el valor de $e_m(2) = 18$, correspondiente a la segunda componente, es menor que $y_2$, no es posible realizar la codificación como está definida.

■

Pero entonces, aparecen varias preguntas: ¿qué tan probable es que se presente un caso como éste? ¿Es posible evitar que suceda? Y, en caso de que no sea posible, ¿qué se puede hacer cuando suceda?

En respuesta a la primera pregunta y, en cierto sentido, a la segunda, se puede decir que, por la forma en la que normalmente se trabaja al utilizar un clasificador automático de patrones durante la investigación científica, lo más común es que se cuente con una base de datos, que los investigadores particionan en dos colecciones mutuamente ajenas: un conjunto fundamental y un conjunto de prueba. Bajo estas condiciones, es posible considerar todos los patrones, tanto fundamentales como de prueba, al momento de calcular los valores de $e_m(j)$. Sin embargo, cabe mencionar que en el caso más general, que incluye la aplicación de los clasificadores de patrones a actividades reales, no es posible asegurar que se conocen todos los patrones de antemano. De hecho, esa es una de las grandes bondades de estos sistemas, que puedan ser utilizados en situaciones en las que los patrones a clasificar son totalmente desconocidos. Entonces, la situación mostrada en el Ejemplo 4.12 es bastante real y, aunque no es fácil calcular la probabilidad de que suceda fuera del laboratorio, se puede asegurar que en general, sucederá.

Como no es posible evitar que este problema se presente, para el caso general, se vuelve necesario encontrar una solución al mismo. Para ello, será necesario modificar y generalizar el operador gama de similitud para que contemple esta situación.

**Definición 4.13** *Sean: el conjunto* $A = \{0, 1\}$*, dos números* $n, m \in \mathbb{Z}^+$*,* $n \leq m$*,* $\mathbf{x} \in A^n$ *y* $\mathbf{y} \in A^m$ *dos vectores binarios, n-dimensional y m-dimensional, respectivamente, con la i-ésima componente representada por* $x_i$ *y* $y_i$*, respectivamente; y* $\theta$ *un número entero no negativo. Se define el operador gama de similitud generalizado* $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ *de la siguiente manera:* $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ *tiene como argumentos de entrada dos vectores binarios* $\mathbf{x}$ *y* $\mathbf{y}$*, y un número entero no negativo* $\theta$*, y la salida es un número binario que se calcula así:*

$$\gamma_g(\mathbf{x}, \mathbf{y}, \theta) = \begin{cases} 1 & si\ m - u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2] \leq \theta \\ 0 & en\ otro\ caso \end{cases}$$

■

Lo que se ha modificado en esta Definición con respecto a la Definición 4.1 es que se le ha sumado la diferencia entre $m$ y $n$:

$$\begin{aligned} n - u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2] + (m - n) &= -u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2] + m \\ &= m - u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2] \end{aligned}$$

Es claro que, si $m = n$, la Definición 4.13 es equivalente a la Definición 4.1.

Entonces, el algoritmo de clasificación propuesto se modifica para utilizar $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ en lugar de $\gamma(\mathbf{x}, \mathbf{y}, \theta)$, quedando como sigue (Figura 4.1).

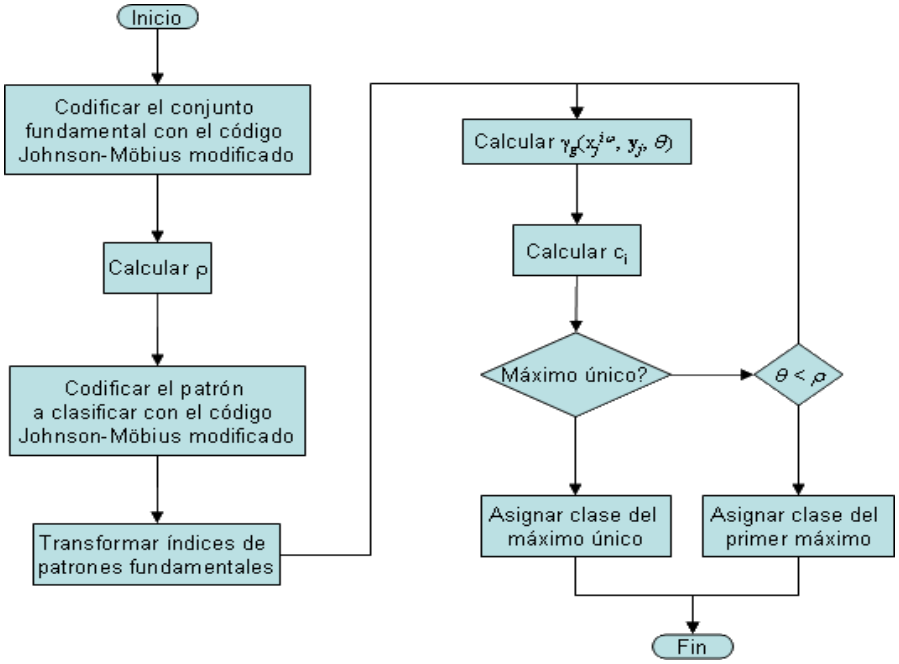



Figura 4.1: Diagrama de bloques del algoritmo del Calsificador Gama (generalizado)

**Algoritmo 4.14** *Sean los números* $k, m, n, p \in \mathbb{Z}^+$, $\{\mathbf{x}^\mu \,|\mu = 1, 2, \ldots, p\}$ *el conjunto fundamental de patrones de cardinalidad* $p$, *donde* $\forall \mu\, \mathbf{x}^\mu \in \mathbb{R}^n$ *y* $\mathbf{y} \in \mathbb{R}^n$ *es un vector real* $n$*-dimensional a ser clasificado. Se asume que el conjunto fundamental está particionado en* $m$ *clases diferentes, donde cada clase tiene cardinalidad* $k_i$, $i = 1, 2, \ldots, m$, *por lo que* $\sum_{i=1}^{m} k_i = p$. *Para clasificar el patrón* $\mathbf{y}$, *se realiza lo siguiente:*

1. Codificar cada componente de cada patrón del conjunto fundamental con el código Johnson-Möbius modificado, obteniéndose un valor $e_m = \bigvee_{i=1}^{p} x_j^i$ por cada componente. Así, la componente $x_j^i$ se transforma en un vector binario de dimensión $e_m(j)$.

2. Calcular el parámetro de paro $\rho = \bigwedge_{j=1}^{n} e_m(j)$.

3. Codificar cada componente del patrón a clasificar con el código Johnson-Möbius modificado, utilizando las mismas condiciones que se utilizaron para codificar las componentes de los patrones fundamentales. En caso de que alguna componente del patrón a clasificar sea mayor al $e_m$ correspondiente ($y_\xi > e_m(\xi)$), igualar esa componente a $e_m(\xi)$ y guardar su valor anterior en la variable $nmax_\xi$.

4. Realizar una transformación de índices en los patrones del conjunto fundamental, de manera que el índice único que tenía un patrón originalmente en el conjunto

fundamental, por ejemplo $\mathbf{x}^{\mu}$, se convierta en dos índices: uno para la clase (por ejemplo la clase $i$) y otro para el orden que le corresponde a ese patrón dentro de esa clase (por ejemplo $\omega$). Bajo estas condiciones ejemplificadas, la notación para el patrón $\mathbf{x}^{\mu}$ será ahora, con la transformación, $\mathbf{x}^{i\omega}$. Lo anterior se realiza para todos los patrones del conjunto fundamental.

5. Inicializar $\theta$ a 0.

6. Realizar la operación $\gamma_g\left(\mathbf{x}_j^{i\omega}, \mathbf{y}_j, \theta\right)$ para cada clase y para cada componente de cada uno de los patrones fundamentales que corresponden a esa clase, y del patrón a clasificar, considerándose $nmax_{\xi}$ como la dimensión del patrón binario $y_{\xi}$.

7. Calcular la suma ponderada $c_i$ de los resultados obtenidos en el paso 6, para cada clase $i = 1, 2, \ldots, m$:

$$c_i = \frac{\sum_{\omega=1}^{k_i} \sum_{j=1}^{n} \gamma_g\left(\mathbf{x}_j^{i\omega}, \mathbf{y}_j, \theta\right)}{k_i}$$

8. Si existe más de un máximo entre las sumas ponderadas por clase, incrementar $\theta$ en 1 y repetir los pasos 6 y 7 hasta que exista un máximo único, o se cumpla con la condición de paro $\theta \geq \rho$.

9. Si existe un máximo único, asignar al patrón a clasificar la clase correspondiente a ese máximo:

$$C_y = C_j \text{ tal que } \bigvee_{i=1}^{m} c_i = c_j$$

10. En caso contrario: si $\lambda$ es el índice más pequeño de clase que corresponde a uno de los máximos, asignar al patrón a clasificar la clase $C_{\lambda}$.

■

Como los patrones con los que se opera $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ en esta algoritmo están codificados con el código Johnson-Möbius modificado, todos los patrones en la $i$-ésima componente tendrán la misma dimensión: $e_m(i)$; la única excepción es cuando $y_{\xi} > e_m(\xi)$, en cuyo caso se trunca a $e_m(\xi)$ y $\gamma_g\left(\mathbf{x}_j^{i\omega}, \mathbf{y}_j, \theta\right)$ toma $nmax_{\xi}$ como dimensión de $\mathbf{y}_{\xi}$.

Con esta modificación al Algoritmo, es posible resolver el Ejemplo 4.12.

**Ejemplo 4.15** *Sea el conjunto fundamental integrado por los patrones* $\mathbf{x}^1 = \begin{pmatrix} -8 \\ 6 \\ 4 \\ 2 \end{pmatrix}$, $\mathbf{x}^2 = \begin{pmatrix} -11 \\ 10 \\ 0 \\ -1 \end{pmatrix}$, $\mathbf{x}^3 = \begin{pmatrix} 3 \\ 15 \\ 15 \\ 10 \end{pmatrix}$, $\mathbf{x}^4 = \begin{pmatrix} -7 \\ 7 \\ 3 \\ 0 \end{pmatrix}$, $\mathbf{x}^5 = \begin{pmatrix} 5 \\ 18 \\ 10 \\ 5 \end{pmatrix}$, *agrupados en dos clases*

$C_1 = \{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^4\}$ *y* $C_2 = \{\mathbf{x}^3, \mathbf{x}^5\}$*; clasificar el patrón* $\mathbf{y} = \begin{pmatrix} 0 \\ 23 \\ 7 \\ 7 \end{pmatrix}$.

De nueva cuenta $n = 4$, $m = 2$, $k_1 = 3$ y $k_2 = 2$. Para codificar los patrones fundamentales con el código Johnson-Möebius modificado es necesario transformarlos, quedando de la siguiente manera:

$$\mathbf{x}^1 = \begin{pmatrix} 3 \\ 6 \\ 4 \\ 3 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0 \\ 10 \\ 0 \\ 0 \end{pmatrix}, \mathbf{x}^3 = \begin{pmatrix} 14 \\ 15 \\ 15 \\ 11 \end{pmatrix}, \mathbf{x}^4 = \begin{pmatrix} 4 \\ 7 \\ 3 \\ 1 \end{pmatrix}, \mathbf{x}^5 = \begin{pmatrix} 16 \\ 18 \\ 10 \\ 6 \end{pmatrix}$$

Entonces, los valores para $e_m(j)$ son, respectivamente: 16, 18, 15, 11, quedando $\rho = 11$; y los patrones ya codificados quedan como sigue:

$$\mathbf{x}^1 = \begin{pmatrix} 0000000000000111 \\ 000000000000111111 \\ 000000000001111 \\ 00000000111 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0000000000000000 \\ 000000001111111111 \\ 000000000000000 \\ 00000000000 \end{pmatrix}$$

$$\mathbf{x}^3 = \begin{pmatrix} 0011111111111111 \\ 000111111111111111 \\ 111111111111111 \\ 11111111111 \end{pmatrix}, \mathbf{x}^4 = \begin{pmatrix} 0000000000001111 \\ 000000000001111111 \\ 000000000000111 \\ 00000000001 \end{pmatrix}$$

$$\mathbf{x}^5 = \begin{pmatrix} 1111111111111111 \\ 111111111111111111 \\ 000001111111111 \\ 00000111111 \end{pmatrix}$$

Ahora bien, para codificar $\mathbf{y}$ es necesario transformarlo antes, quedando así:

$$\mathbf{y} = \begin{pmatrix} 11 \\ 23 \\ 7 \\ 8 \end{pmatrix}$$

Una vez realizada la conversión, queda como sigue.

$$\mathbf{y} = \begin{pmatrix} 0000011111111111 \\ 111111111111111111 \\ 000000001111111 \\ 00011111111 \end{pmatrix}$$

Nótese que $y_3$ ha sido truncado a 18, el valor correspondiente de $e_m(2)$; quedando $nmax_2 = 23$. Ahora si, es posible continuar con el algoritmo.

- Para $\theta = 0$:

$\gamma_g\left(\mathbf{x}_1^{11}, \mathbf{y}_1, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{11}, \mathbf{y}_2, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{11}, \mathbf{y}_3, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{11}, \mathbf{y}_4, 0\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{12}, \mathbf{y}_1, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{12}, \mathbf{y}_2, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{12}, \mathbf{y}_3, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{12}, \mathbf{y}_4, 0\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{13}, \mathbf{y}_1, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{13}, \mathbf{y}_2, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{13}, \mathbf{y}_3, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{13}, \mathbf{y}_4, 0\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{21}, \mathbf{y}_1, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{21}, \mathbf{y}_2, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{21}, \mathbf{y}_3, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{21}, \mathbf{y}_4, 0\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{22}, \mathbf{y}_1, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{22}, \mathbf{y}_2, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{22}, \mathbf{y}_3, 0\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{22}, \mathbf{y}_4, 0\right) = 0$; por lo que $c_1 = \frac{(0+0+0+0)+(0+0+0+0)+(0+0+0+0)}{3} = \frac{0}{3}$ y $c_2 = \frac{(0+0+0+0)+(0+0+0+0)}{2} = \frac{0}{2}$, y como el máximo no es único (0 en ambos casos) y además $\theta = 0 < \rho = 11$, se incrementa $\theta$ y repiten los pasos 6 y 7.

- Para $\theta = 1$ :

$\gamma_g\left(\mathbf{x}_1^{11}, \mathbf{y}_1, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{11}, \mathbf{y}_2, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{11}, \mathbf{y}_3, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{11}, \mathbf{y}_4, 1\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{12}, \mathbf{y}_1, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{12}, \mathbf{y}_2, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{12}, \mathbf{y}_3, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{12}, \mathbf{y}_4, 1\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{13}, \mathbf{y}_1, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{13}, \mathbf{y}_2, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{13}, \mathbf{y}_3, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{13}, \mathbf{y}_4, 1\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{21}, \mathbf{y}_1, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{21}, \mathbf{y}_2, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{21}, \mathbf{y}_3, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{21}, \mathbf{y}_4, 1\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{22}, \mathbf{y}_1, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{22}, \mathbf{y}_2, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{22}, \mathbf{y}_3, 1\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{22}, \mathbf{y}_4, 1\right) = 0$; por lo que $c_1 = \frac{(0+0+0+0)+(0+0+0+0)+(0+0+0+0)}{3} = \frac{0}{3}$ y $c_2 = \frac{(0+0+0+0)+(0+0+0+0)}{2} = \frac{0}{2}$, y como el máximo no es único y además $\theta = 1 < \rho = 11$, se incrementa $\theta$ y se repiten los pasos 6 y 7.

- Para $\theta = 2$:

$\gamma_g\left(\mathbf{x}_1^{11}, \mathbf{y}_1, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{11}, \mathbf{y}_2, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{11}, \mathbf{y}_3, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{11}, \mathbf{y}_4, 2\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{12}, \mathbf{y}_1, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{12}, \mathbf{y}_2, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{12}, \mathbf{y}_3, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{12}, \mathbf{y}_4, 2\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{13}, \mathbf{y}_1, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{13}, \mathbf{y}_2, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{13}, \mathbf{y}_3, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{13}, \mathbf{y}_4, 2\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{21}, \mathbf{y}_1, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{21}, \mathbf{y}_2, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{21}, \mathbf{y}_3, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{21}, \mathbf{y}_4, 2\right) = 0$;
$\gamma_g\left(\mathbf{x}_1^{22}, \mathbf{y}_1, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_2^{22}, \mathbf{y}_2, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_3^{22}, \mathbf{y}_3, 2\right) = 0$, $\gamma_g\left(\mathbf{x}_4^{22}, \mathbf{y}_4, 2\right) = 1$; por lo que $c_1 = \frac{(0+0+0+0)+(0+0+0+0)+(0+0+0+0)}{3} = \frac{0}{3}$ y $c_2 = \frac{(0+0+0+0)+(0+0+0+1)}{2} = \frac{1}{2}$, con lo que se obtiene un máximo único $\bigvee_{i=1}^{2} c_i = c_2 = \frac{1}{2}$. Entonces, a $\mathbf{y}$ se le asigna la clase $C_2$.

■

## 4.3. Complejidad del algoritmo propuesto

Una medida de la eficiencia de un algoritmo es el tiempo usado por la computadora para resolver un problema utilizando dicho algoritmo. El análisis del tiempo requerido para resolver un problema de un tamaño en particular implica la complejidad en tiempo del algoritmo utilizado para resolver ese problema. La complejidad en tiempo de un algoritmo se puede expresar en términos del número de operaciones usadas por el algoritmo cuando la entrada tiene un tamaño en particular. Las operaciones utilizadas para medir la complejidad en tiempo pueden ser la comparación de enteros, la suma de enteros, la división de enteros, asignaciones de variables, comparaciones lógicas, o cualquier otra operación elemental.

Entonces, es posible analizar la complejidad en tiempo del algoritmo propuesto al estudiar cuántas operaciones elementales necesita realizar. Tomando como base el diagrama del algoritmo mostrado en la Figura 4.1, el algoritmo queda como sigue:

1. Codificar el conjunto fundamental con el código Johnson-Möbius modificado.
2. Calcular $\rho$.
3. Codificar el patrón a clasificar con el código Johnson-Möbius modificado.
4. Transformar los índices del conjunto fundamental.
5. Repetir el ciclo al menos una vez, y hasta $\rho$ veces.
   *a*) Calcular $\gamma_g$.
   *b*) Calcular $c_i$.
   *c*) Determinar si hay un máximo único; si $\theta < \rho$, continuando con el siguiente ciclo; o si se ha presentado la condición de paro.
6. Asignar la clase, dependiendo si se encontró un máximo único o se cumplió con la condición de paro.

A continuación se presenta la cantidad de operaciones elementales requeridas en cada paso, tomándose donde corresponde el peor caso.

1. $e_m \times n \times p$ comparaciones, $e_m \times n \times p$ asignaciones. Nótese que para cada componente se tienen $e_m$ diferentes; para este análisis, se asume el valor mayor para todas las componentes.
2. $n$ comparaciones, 1 asignación.
3. $e_m \times n$ comparaciones, $e_m \times n$ asignaciones. Situación similar al paso 1.
4. $p$ comparaciones, $p + p$ asignaciones.
5. 1 asignación (calor de $\theta$); entre 1 y $\rho$ repeticiones. Para este análisis, se toma en cuenta $\rho$.
   *a*) $n \times p \times \big($complejidad de $\gamma_g\big)$.
   *b*) $p - m$ sumas, $m$ divisiones.
   *c*) 2 comparaciones.
6. 1 asignación.

Ahora bien, antes de continuar, es necesario analizar la complejidad del operador gama de similitud generalizado. Para ello, se parte de la Definición 4.13, realizando una división en pasos al estilo del Algoritmo B.1, presentado en el Apéndice B.

1. Calcular $\alpha(\mathbf{x}, \mathbf{y})$.

2. Calcular $\alpha(\mathbf{x}, \mathbf{y}) \bmod 2$.
3. Calcular $u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2]$.
4. Calcular $m - u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2]$.
5. Evaluar la desigualdad $m - u_\beta[\alpha(\mathbf{x}, \mathbf{y}) \bmod 2] \leq \theta$.
6. Asignar el resultado.

La cantidad de operaciones elementales que cada uno de los pasos anteriores requiere son las siguientes.

1. $n$ comparaciones, $n$ asignaciones.
2. $n$ comparaciones, $n$ asignaciones.
3. $n$ sumas.
4. 1 resta.
5. 1 comparación.
6. 1 asignación.

Entonces, la complejidad en tiempo del operador gama de similitud generalizado, expresado en función de $n$ ($f(n)$), es:

$$\begin{aligned} f(n) &= 2n + 2n + n + 1 + 1 + 1 \\ &= 5n + 3 \end{aligned}$$

Así, la complejidad en tiempo del algoritmo propuesto queda expresada como sigue:

$$\begin{aligned} f(e_m, n, p, \rho) &= 2e_m np + n + 1 + 2e_m n + 3p + \rho[np(5n+3) + p - m + m + 2 + 1] + 1 \\ &= 2e_m n(p+1) + n + 3p + 2 + \rho\left(5n^2 p + 3np + 3\right) \end{aligned}$$

Para analizar la factibilidad del algoritmo es necesario entender qué tan rápido crece la función obtenida conforme aumenta el valor de alguna variable. Para este cometido, se utilizará la notación BIG-O [112], que se define a continuación.

Sean $f$ y $g$ funciones de un conjunto de enteros o de un conjunto de números reales a un conjunto de números reales. Se dice que $f(x)$ es $O(g(x))$ si existen dos constantes $C$ y $k$ tal que:

$$\|f(x)\| \leq C\|g(x)\| \text{ cuando } x > k \tag{4.1}$$

Ahora bien, dado que la complejidad del algoritmo está expresada con respecto a cuatro variables, se analizará cada una de ellas por separado, dando valores típicos a las demás en cada caso. Como caso de estudio se toma la base de datos Iris Plant [114], utilizada en los ejemplos 4.10 y 4.11 y también en algunos de los experimentos descritos en el Capítulo 5.

### 4.3.1. Complejidad con respecto a $e_m$

Sean $n = 4$, $p = 100$ y $\rho = 25$; entonces la complejidad queda expresada como

$$\begin{aligned} f(e_m) &= 2e_m(4)[(100)+1] + (4) + 3(100) + 2 + 25\left[5(4)^2(100) + 3(4)(100) + 3\right] \\ &= 2e_m(4 \times 101) + 4 + 3(100) + 2 + 25[5(16)(100) + 3(400) + 3] \\ &= 808e_m + 230\,381 \end{aligned}$$

Ahora, se deben encontrar $g(x)$, $C$ y $k$ que cumplan con la desigualdad 4.1, para lo cual se propone

$$808e_m + 230\,381e_m = 231\,189e_m$$

con lo que tenemos $g(e_m) = e_m$, $C = 231\,189$ y $k = 1$, lo que hace que la desigualdad 4.1 se cumpla:

$$\|f(e_m)\| \leq 231\,189\,\|g(e_m)\| \text{ cuando } x > 1, \text{ por lo tanto } O(e_m) \tag{4.2}$$

En la Figura 4.3.1 se muestra gráficamente lo expresado anteriormente.

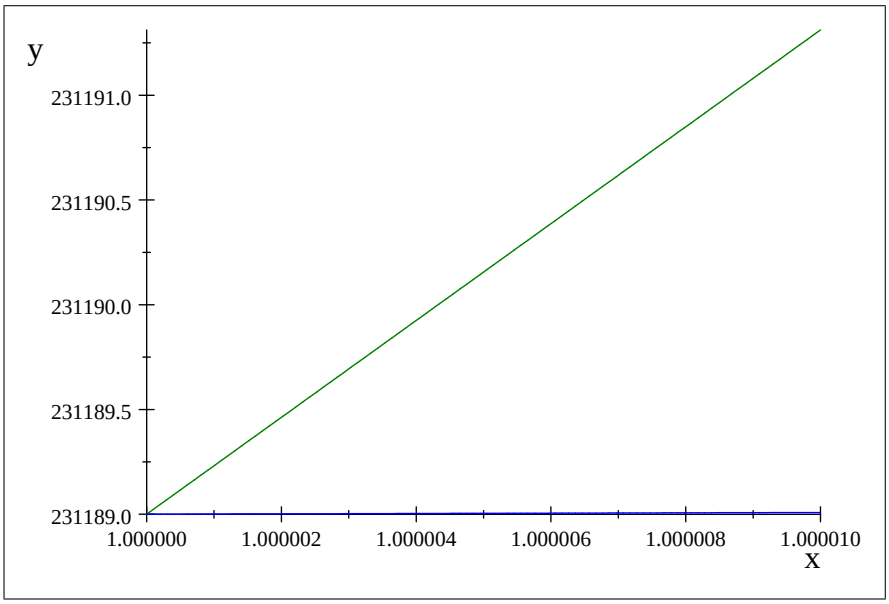



Figura 4.3.1. Complejidad con respecto a $e_m$; $O(e_m)$ en verde, $f(e_m)$ en azul

Como se puede apreciar tanto en la Expresión 4.2 como en la Figura 4.3.1, la complejidad del algoritmo con respecto a $e_m$ es lineal.

### 4.3.2. Complejidad con respecto a $n$

Sean $e_m = 64$, $p = 100$ y $\rho = 25$; entonces la complejidad queda expresada como

$$\begin{aligned} f(n) &= 2(64)n[(100)+1] + n + 3(100) + 2 + 25\left[5n^2(100) + 3n(100) + 3\right] \\ &= 128n(101) + n + 300 + 2 + 25\left(500n^2 + 300n + 3\right) \\ &= 12\,500n^2 + 20\,429n + 377 \end{aligned}$$

Ahora, se deben encontrar $g(x)$, $C$ y $k$ que cumplan con la desigualdad 4.1, para lo cual se propone

$$12\,500n^2 + 20\,429n^2 = 32\,929n^2$$

con lo que tenemos $g(n) = n^2$, $C = 32\,929$ y $k = 1$, lo que hace que la desigualdad 4.1 se cumpla:

$$\|f(n)\| \leq 32\,929 \|g(n)\| \text{ cuando } x > 1, \text{ por lo tanto } O\left(n^2\right) \tag{4.3}$$

En la Figura 4.3.2 se muestra gráficamente lo expresado anteriormente.

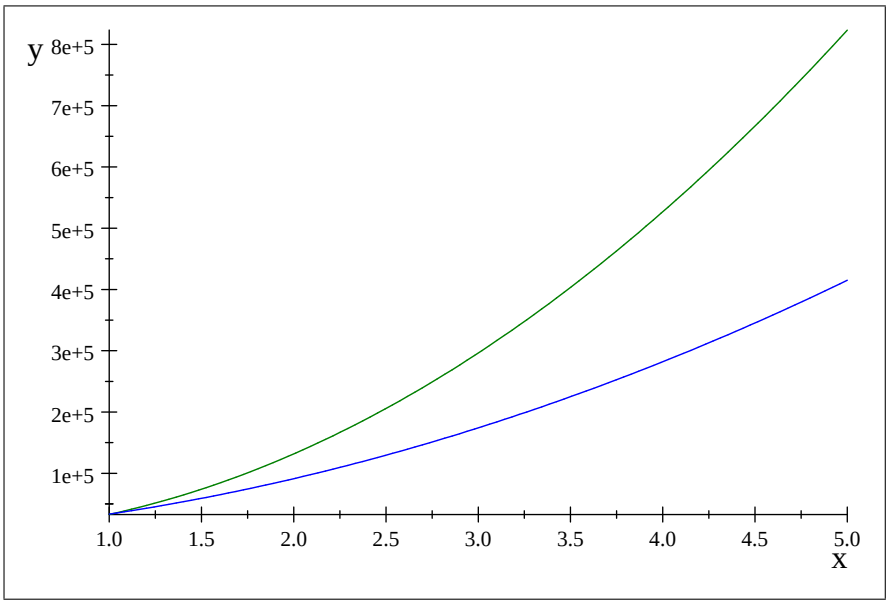



Figura 4.3.2. Complejidad con respecto a $n$; $O(n)$ en verde, $f(n)$ en azul

Como se puede apreciar tanto en la Expresión 4.3 como en la Figura 4.3.2, la complejidad del algoritmo con respecto a $n$ es cuadrática.

### 4.3.3. Complejidad con respecto a $p$

Sean $e_m = 64$, $n = 4$ y $\rho = 25$; entonces la complejidad queda expresada como

$$\begin{aligned} f(p) &= 2(64)(4)(p+1) + (4) + 3p + 2 + 25\left[5(4)^2 p + 3(4)p + 3\right] \\ &= 512(p+1) + (4) + 3p + 2 + 25\left[5(16)p + 3(4)p + 3\right] \\ &= 2815p + 593 \end{aligned}$$

Ahora, se deben encontrar $g(x)$, $C$ y $k$ que cumplan con la desigualdad 4.1, para lo cual se propone

$$2815p + 593p = 3408p$$

con lo que tenemos $g(p) = p$, $C = 3408$ y $k = 1$, lo que hace que la desigualdad 4.1 se cumpla:

$$\|f(p)\| \leq 3408 \|g(p)\| \text{ cuando } x > 1, \text{ por lo tanto } O(p) \tag{4.4}$$

En la Figura 4.3.3 se muestra gráficamente lo expresado anteriormente.

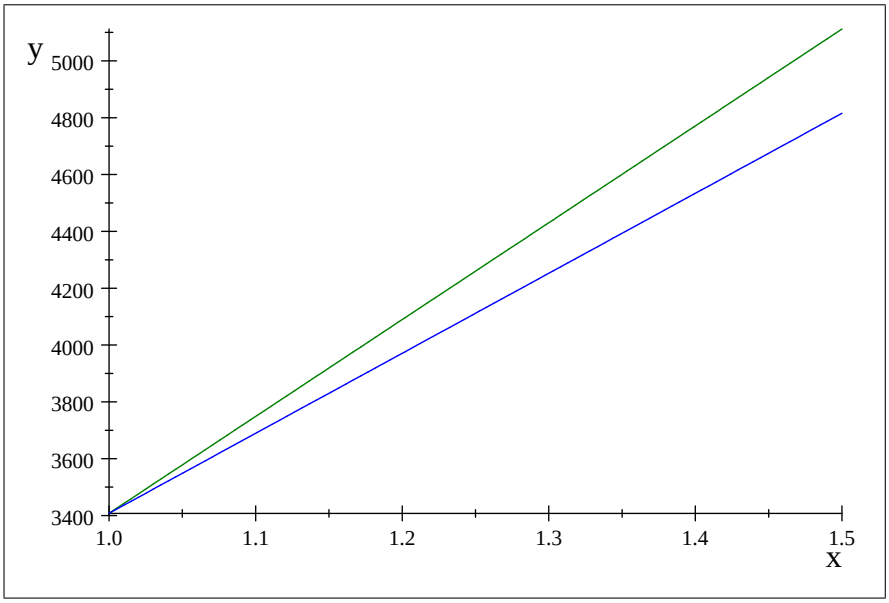



Figura 4.3.3. Complejidad con respecto a $p$; $O(p)$ en verde, $f(p)$ en azul

Como se puede apreciar tanto en la Expresión 4.4 como en la Figura 4.3.3, la complejidad del algoritmo con respecto a $p$ es lineal.

### 4.3.4. Complejidad con respecto a $\rho$

Sean $e_m = 64$, $n = 4$ y $p = 100$; entonces la complejidad queda expresada como

$$\begin{aligned} f(\rho) &= 2(64)(4)[(100)+1] + (4) + 3(100) + 2 + \rho\left(5(4)^2(100) + 3(4)(100) + 3\right) \\ &= 512(101) + 4 + 300 + 2 + \rho[500(16) + 1200 + 3] \\ &= 9203\rho + 52018 \end{aligned}$$

Ahora, se deben encontrar $g(x)$, $C$ y $k$ que cumplan con la desigualdad 4.1, para lo cual se propone

$$9203\rho + 52018\rho = 61221\rho$$

con lo que tenemos $g(\rho) = \rho$, $C = 61221$ y $k = 1$, lo que hace que la desigualdad 4.1 se cumpla:

$$\|f(\rho)\| \leq 61221\,\|g(\rho)\| \text{ cuando } x > 1, \text{ por lo tanto } O(\rho) \tag{4.5}$$

En la Figura 4.3.4 se muestra gráficamente lo expresado anteriormente.

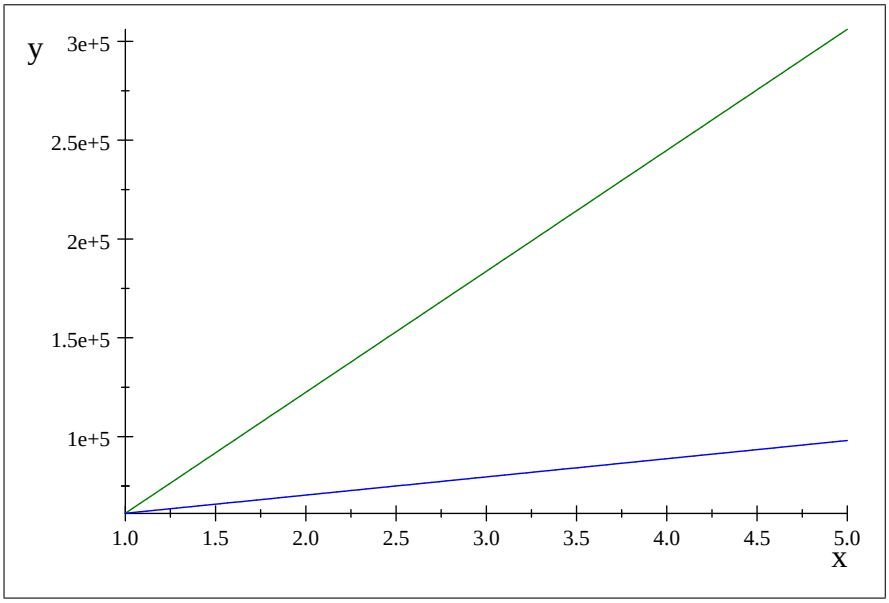



Figura 4.3.4. Complejidad con respecto a $\rho$; $O(\rho)$ en verde, $f(\rho)$ en azul

Como se puede apreciar tanto en la Expresión 4.5 como en la Figura 4.3.4, la complejidad del algoritmo con respecto a $\rho$ es lineal.

Una vez analizada la complejidad del algoritmo propuesto desde la perspectiva de cada una de sus cuatro variables ($e_m$, $n$, $p$, $\rho$), es posible concluir que en el peor de los casos, su complejidad (en notación BIG-O) es cuadrática con respecto a la dimensión de los patrones de entrada: $O\left(n^2\right)$.

# Capítulo 5

# Resultados y Discusión

Este Capítulo está dedicado a la presentación de resultados experimentales realizados con el algoritmo propuesto y su discusión. Los experimentos se llevaron a cabo con un *software* diseñado e implementado en Borland C++ Builder 6.0, *ex-profeso* para este fin (cuyo manual de uso se incluye en el Apéndice D). Dicho *software* se ejecutó en una computadora personal con microprocesador DualCore Intel Pentium D 950 de 64 bits a 3400 MHz, 2 GBytes de memoria DDR2 SDRAM, con disco duro SCSI de 80 GBytes; sobre el sistema operativo Windows XP Professional x64 bits.

Diversas bases de datos fueron utilizados con el fin de probar la eficacia del algortimo propuesto, así como de comparar sus resultados frente a otros algoritmos clasificadores de patrones. Dichas bases da datos son: *Iris Plants Database*, tomada de [114]; *Wireless Sensor Network Localization Measurement Repository*, reportada en [115] y disponible en [116].

## 5.1. Iris Plant Database

En primera instancia se presenta la base de datos *Iris Plants Database*, que fue tomada del Repositorio de *Machine Learning* del Departamento de Información y Ciencias de la Computación de la University of California, Irvine [114]. Esta es una base de datos clásica en el área de Reconocimiento de Patrones y, a pesar de ser una de las primeras bases de datos utilizadas para probar algoritmo del área, sigue siendo muy utilizada. El conjunto de datos contiene 3 clases de 50 instancias cada una, donde cada clase hace referencia a un tipo de planta de iris (Iris Setosa, Iris Versicolor e Iris Virginia). Cada patrón que representa una planta de iris tiene 4 atributos más uno que representa la clase. En esta base de datos los rasgos que representan el largo y ancho de los pétalos están altamente correlacionados. Asimismo, se conoce que mientras la clase 1 (Iris Setosa) es linealmente separable de las otras dos clases, las clases 2 y 3 (Iris Versicolor e Iris Virginica, respectivamente) no son linealmente separables entre sí. En la Tabla 5.1 se describen los rasgos presentes en esta base de datos.

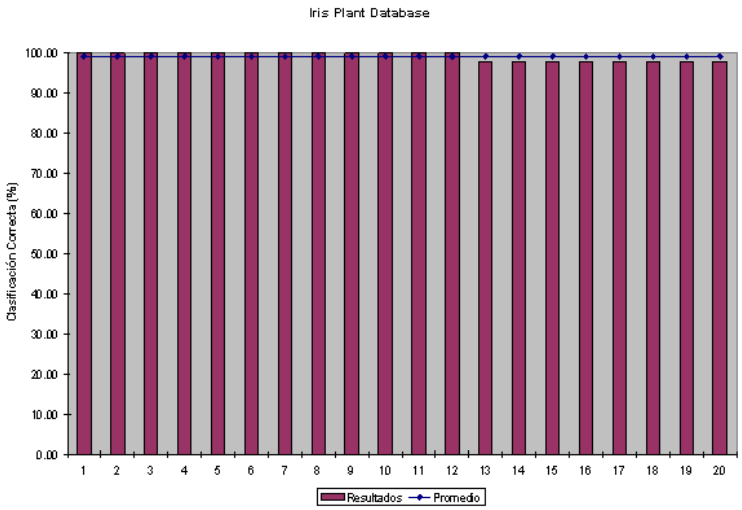



Figura 5.1: Experimentos con la base de datos *Iris Plants Database*

**Tabla 5.1**. Rasgos de la base de datos *Iris Plant*

| **Rasgo** | **Descripción** |
|---|---|
| 1 | Largo del sépalo en centímetros |
| 2 | Ancho del sépalo en centímetros |
| 3 | Largo del pétalo en centímetros |
| 4 | Ancho del pétalo en centímetros |
| 5 | Clase |

Se realizaron 1000 experimentos, particionando aleatoriamente la base de datos en dos conjuntos mutuamente excluyentes: un conjunto fundamental y un conjunto de prueba, en donde la cantidad de patrones fundamentales que aportó cada clase fue igual al aportado por las otras dos clases. De esos 1000 experimentos, se tomaron los 20 mejores resultados de clasificación y se muestran en la Figura 5.1. En esta Figura, las barras indican los resultados de clasificación correcta de los mencionados experimentos, siendo 100 % el mejor de ellos.

En la Tabla 5.2 se presenta una comparación de los redimientos reportados en la tesis [52] y en un artículo publicado el mes de enero de 2007 en la prestigiosa revista Pattern Recognition [117], del desempeño de varios clasificadores con respecto a la base de datos *Iris Plant*. En particular, es importante hacer notar que los dos mejores algoritmos según la Tabla 6.4 de la mencionada tesis, al tomar los mejores 20 resultados de 1000 experimentos, son el 1-NN con la mejor clasificación correcta de 96.67 % y el CHAT con la mejor clasificación correcta de 98.67 %. Por otro lado, los dos mejores clasificadores multiclase presentados en la Tabla 7 del artículo mencionado, son el de 3 redes neuronales construido con el método OAO y el de 1 red neuronal construido con el método OAA, ambos con la mejor clasificación correcta de 98 % al probarlos con un proceso de *10-fold cross validation*.

El algoritmo propuesto en esta tesis supera de manera notable a los cuatro algoritmos del párrafo anterior, y por ende a todos los demás algoritmos reportados, tanto en la tesis y el artículo mencionados en el párrafo anterior, como en otros trabajos [118]: en los 12 mejores resultados, tomados de los 1000 experimentos que se realizaron al particionar aleatoriamente la base de datos en dos conjuntos mutuamente excluyentes (un conjunto fundamental y un conjunto de prueba), la clasificación correcta fue del 100 %, con lo que se supera el rendimiento de todos los clasificadores con los cuales se le compara en la Tabla 5.2. Además, si se toman los 20 mejores resultados, el promedio del porcentaje de recuperación correcta es 99.11 %, valor representado por la línea en la Figura 5.1 y que, por sí mismo, supera a los demás métodos.

**Tabla 5.2**. Comparación del rendimiento con la base de datos *Iris Plant*

| Clasificador | Mejor rendimiento |
|---|---|
| **1-NN** | **96.67 %** |
| 3-NN | 95.33 % |
| C-means | 89.33 % |
| C-means difuso | 90.0 % |
| C-means difuso usando func. de disimilaridad | 96.0 % |
| **CHAT** | **98.67 %** |
| OAA 3 nets | 96 % |
| **OAO 3 nets** | **98 %** |
| OAHO 2 nets | 96 % |
| **OAA 1 net** | **98 %** |
| **Algoritmo propuesto** | **100 %** |

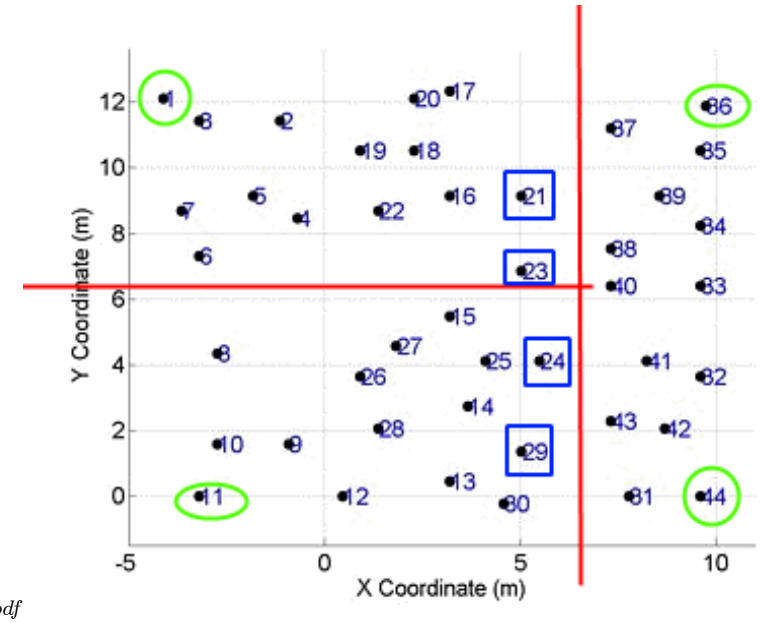
## 5.2. Wireless Sensor Network Localization

En segunda instancia, se presentan los experimentos realizados con la base de datos *Wireless Sensor Network Localization*, que fue desarrollada y publicada por Patwari *et al.* en [115]. Esta base de datos está compuesta por las mediciones de poder entre 44 sensores distribuidos en las oficinas de Motorala en Plantation, Florida, generándose una matriz simétrica de 44 por 44 valores. De aquí, se tomaron arbitrariamente 4 sensores como referencia, quedando una matriz de 4 rasgos (las medidas de poder de la señal entre cada sensor y cada uno de los sensores de referencia) y 40 registros, como se puede apreciar en la Tabla 5.3.

**Tabla 5.3**. Rasgos de la base de datos *Wireless Sensor Network Localization*

| Rasgo | Descripción |
|---|---|
| 1 | Potencia de la señal al sensor 1 |
| 2 | Potencia de la señal al sensor 11 |
| 3 | Potencia de la señal al sensor 36 |
| 4 | Potencia de la señal al sensor 44 |
| 5 | Clase |

02




2.*pdf*

Figura 5.2: Elementos de la base de datos *Wireless Sensor Network Localization* y clasificación. Los círculos indican los sensores de referencia, mientras que los rectángulos indican los elemntos ignorados en este experimento; las líneas indican los limites entre clases

Dado que la base de datos original no incluye una partición en clases, estas clases se definieron arbitrariamente, procurando aproximar la distribución de un espacio en habitaciones convencionales y balancear las clases. Así, se tienen 3 clases de 12 elementos cada una, tal y como se muestra en la Figura 5.2.

Se realizaron 1000 experimentos, particionando aleatoriamente la base de datos en dos conjuntos mutuamente excluyentes: un conjunto fundamental y un conjunto de prueba, en donde la cantidad de patrones fundamentales que aportó cada clase fue igual al aportado por las otras dos clases. De esos 1000 experimentos, se tomaron los 20 mejores resultados de clasificación y se muestran en la Figura 5.3. En esta Figura, las barras indican los resultados de clasificación correcta de los mencionados experimentos, siendo 91.67 % el mejor de ellos.

Hasta la fecha, no se reportan publicaciones en las que se utilice esta base de datos para probar algoritmos clasificadores de patrones. Sin embargo, resulta notable que de los 20 mejores resultados obtenidos con el clasificador Gama, 7 presentan una clasificación correcta de 91.67 % (3 errores en 36 registros) y los restantes 13 presentan una clasificación correcta de 88.89 % (4 errores en 36 registros), presentando un promedio de clasificación correcta de 89.86 %. Al comparar estos resultados con los obtenidos por el clasificador 1-NN implementado por Tengli [119], que presenta 14 errores para un porcentaje de clasificación correcta de 65 % al realizar un proceso de *10-fold cross validation*, es claro que el clasificador gama supera al 1-NN en esta base de datos (Tabla 5.4).

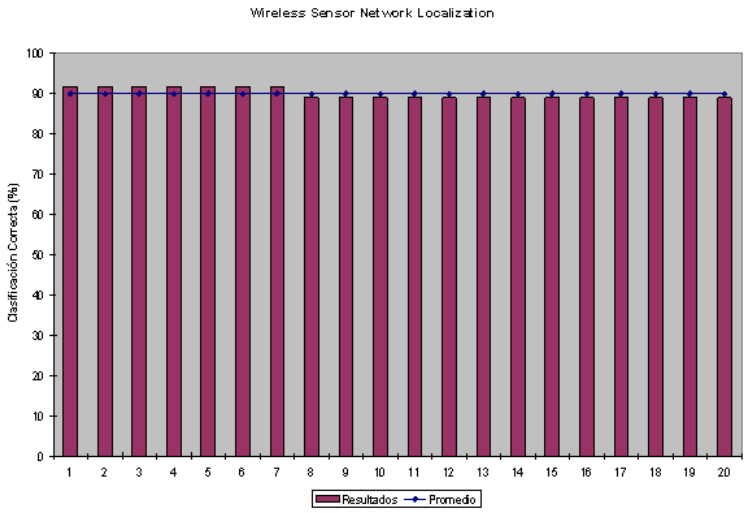



Figura 5.3: Experimentos con la base de datos *Wireless Sensor Network Localization*

**Tabla 5.4**. Comparación del rendimiento con la base de datos *Wireless Sensor Network Localization*

| **Clasificador** | **Mejor rendimiento** |
|---|---|
| 1-NN (Fast Classifiers) | 65 % |
| **Algoritmo propuesto** | **91.67 %** |

## 5.3. Teorema de *No-Free-Lunch*

Con los resultados presentados hasta ahora, es posible ver que el algoritmo propuesto exhibe un desempeño de clasificación competitivo en diversas bases de datos, aunque en algunas es superado por otros clasificadores. Sin embargo, esto no es sorprendente; de acuerdo con el teorema de *No-Free-Lunch*, "... para cualquier algoritmo, cualquier desempeño elevado en una clase de problemas es pagado exactamente en desempeño en otra clase"[120], [121]. O como lo exponen Duda y Hart: "Si un algoritmo parece superar a otro en una situación particular, esto es consecuencia de su adaptación al problema particular de reconocimiento de patrones, no a la superioridad general del algoritmo"[3]. Una representación gráfica de lo anterior se puede ver en la Figura 5.4, tomada de [122].

Entonces, es de esperarse que en otros problemas, el algoritmo propuesto no se adpate tan bien como otros algoritmos, presentando un desempeño inferior. En efecto, al probar el Clasificador Gama propuesto en esta tesis con algunas otras bases de datos, como por ejemplo la *Glass Identification Database* [114], los resultados no fueron competitivos.

Sin embargo, cabe mencionar que una de las bondades apreciadas en el algoritmo, es que resulta muy rápido adaptarlo a configuraciones de clases diferentes: sólo es necesario transformar los índices y recalcular las sumas ponderadas, puesto que los

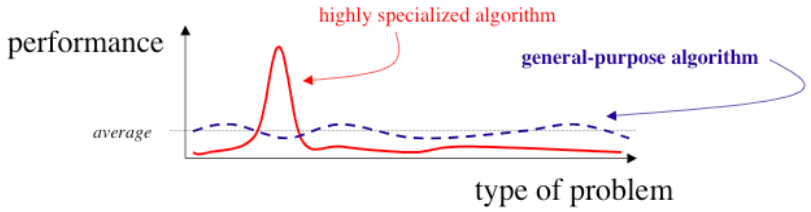



Figura 5.4: Representación gráfica del teorema de *No-Free-Lunch*

resultados de aplicar el operador $\gamma_g$ ya se tienen disponibles. Estas dos operaciones tienen una complejidad temporal exactamente igual a $p$, lo que resulta en una operación de complejidad total equivalente a dos veces la cantidad de patrones fundamentales. Al compararla con lo que necesitan otros algoritmos (por ejemplo, SVMs requerirían volver a calcular los vectores de soporte y, tal vez, incluso utilizar otra función kernel), resulta ser una ventaja importante.

# Capítulo 6

# Conclusiones y Trabajo Futuro

En este capitulo se presentan las conclusiones derivadas de los resultados obtenidos en el proceso de este trabajo de tesis; además, se proponen algunos de los posibles trabajos que se podrían realizar con objeto de continuar con las ideas propuestas aquí, dando la pauta a futuros investigadores sobre los puntos no cubiertos, pero que pudieran ser afrontados en otros trabajos de investigación.

## 6.1. Conclusiones

1. En este trabajo de tesis se introduce un nuevo algoritmo clasificador de patrones, el clasificador gama.

2. Aunque el algoritmo propuesto es iterativo, no sufre de problemas de convergencia, ya que incluye un parámetro de paro.

3. Dado que el valor del parámetro de paro se obtiene automáticamente de los patrones de aprendizaje, el algoritmo propuesto no necesita que se le proporcionen parámetros; esto es, el algoritmo clasificador de patrones introducido es no paramétrico.

4. Se definen los nuevos operadores que sirven de base y fundamentan el algoritmo propuesto: el operador $u_\beta$, el operador $\gamma$ de similitud y el operador $\gamma_g$ de similitud generalizado.

5. Se define un algoritmo alternativo para calcular el resultado del operador $\gamma_g$ de similitud generalizado.

6. Con los resultados anteriores se diseña un nuevo algoritmo que permite realizar la tarea de clasificación de patrones de una manera eficaz y eficiente.

7. El algoritmo clasificador de patrones propuesto es competitivo con respecto a los clasificadores de patrones presentes en la literatura científica actual, superando incluso a la mayoría en algunas bases de datos.

8. El algoritmo propuesto exhibe una amplia facilidad para reconfigurarse en caso de cambiar la estructura de las clases.

9. El nuevo algoritmo clasificador de patrones tiende a sufrir problemas de ineficiencia temporal cuando el conjunto fundamental de patrones es muy pequeño.

10. Se presentan tres teoremas que permiten identificar los tres casos de las relaciones de orden (tricotomía): menor que, igual y mayor que, entre patrones binarios codificados con el código Johnson-Möbius modificado.

## 6.2. Trabajo Futuro

1. Probar de una manera más extensiva y exhaustiva el algoritmo propuesto, utilizando otras bases de datos y comparando los resultados obtenidos con aquellos presentados por otros clasificadores de patrones [123].

2. Caracterizar el algoritmo propuesto, de manera que se conozcan las cotas inferior y superior del tamaño del conjunto fundamental, que garanticen un buen desempeño temporal del algoritmo.

3. Desarrollar los aspectos teóricos que fundamenten el funcionamiento, eficacia y eficiencia del algoritmo propuesto.

4. Aplicar el algoritmo clasificador de patrones propuesto a problemas específicos de actualidad .

# Apéndice A

# Simbología

| | |
|---|---|
| $\in$ | pertenencia de un elemento a un conjunto |
| $\bigvee$ | operador máximo |
| $\bigwedge$ | operador mínimo |
| $\forall$ | cuantificador universal |
| $\exists$ | cuantificador existencial |
| $\alpha$, $\beta$ | operadores en los que se basan las memorias $\alpha\beta$ |
| $u_\beta$ | operador original útil para el algoritmo propuesto |
| mod | operador módulo |
| $k(r)$ | operador de cadena binaria mínima |
| $\mathcal{E}(r,k)$ | operador $k$-binario de expansión |
| $\gamma(\mathbf{x},\mathbf{y},\theta)$ | operador gama de similitud (original de esta tesis) |
| $k_i$ | cardinalidad de la $i$-ésima clase del conjunto fundamental |
| $e_m$ | número de bits necesarios para la codificación en código Johnson-Möbius modificado |
| $\rho$ | parámetro de paro del algoritmo propuesto |
| $c_i$ | suma ponderada por clase |
| $\gamma_g(\mathbf{x},\mathbf{y},\theta)$ | operador gama de similitud generalizado (original de esta tesis) |
| $\theta$ | parámetro de restricción del operador gama de similitud (normal y generalizado) |
| $\mathbf{x}$ | vector correspondiente a un patrón fundamental |
| $\mathbf{y}$ | vector correspondiente a un patrón a ser clasificado |
| $n$ | dimensión de los patrones, tanto fundamentales como por clasificar |
| $m$ | cantidad de clases en el conjunto fundamental |
| $p$ | cardinalidad del conjunto fundamental |

# Apéndice B

# Algoritmo Alternativo al Operador $\gamma_g$

En este Apéndice se presenta una forma algorítmica alternativa al operador gama de similitud generalizado $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$.

Este operador es idea original del autor de la presente tesis y puede resolverse alternativamente siguiendo el algoritmo que a continuación se presenta.

**Algoritmo B.1** *Sean: los conjuntos* $A = \{0, 1\}$ *y* $B = \{0, 1, 2\}$*, dos números* $n, m \in \mathbb{Z}^+$*,* $n \leq m$*,* $\mathbf{x} \in A^n$ *y* $\mathbf{y} \in A^m$ *dos vectores binarios, n-dimensional y m-dimensional, respectivamente, con la i-ésima componente representada por* $x_i$ *y* $y_i$*, respectivamente; y* $\theta$ *un número entero no negativo. Se define el operador gama de similitud generalizado* $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ *algorítmicamente de la siguiente manera:* $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ *tiene como argumentos de entrada dos vectores binarios* $\mathbf{x}$ *y* $\mathbf{y}$*, y un número entero no negativo* $\theta$*, y la salida es un número binario que se calcula así:*

1. Truncar los $m-n$ bits izquierdos de $\mathbf{y}$, de tal manera que ahora $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^n$.
2. Calcular el vector entero $n$-dimensional $\mathbf{r}^1 \in B^n$ al realizar la operación $\alpha(\mathbf{x}, \mathbf{y})$.
3. Realizar la expansión binaria para cada componente de $\mathbf{r}^1$; esto es $r_i^2 = \mathcal{E}\left(r_i^1, 2\right)\ \forall i \in \{1, 2, \ldots, n\}$. Con lo anterior y suponiendo que $\mathbf{r}^1$ fuera un vector fila, $\mathbf{r}^2$ puede considerarse como una matriz de dimensiones $2 \times n$, con el bit menos significativo de la expansión binaria en la segunda fila de $\mathbf{r}^2$.
4. Construir el vector $\mathbf{r}^3$ con los bits menos significativos de cada componente de $\mathbf{r}^2$, esto es $r_i^3 = r_{2i}^2$.
5. Calcular el valor entero $r^4 = m - u_\beta\left(\mathbf{r}^3\right)$.
6. Calcular el valor de salida de acuerdo con lo siguiente:

$$\gamma_g = \begin{cases} 1 & \text{si } r^4 \leq \theta \\ 0 & \text{de otra forma} \end{cases}$$

■

**Ejemplo B.2** *Sean* $\mathbf{x} = \begin{pmatrix}0\\1\\1\\0\\0\\1\\0\\1\end{pmatrix}$ *y* $\mathbf{y} = \begin{pmatrix}1\\0\\1\\0\\0\\1\\1\\0\end{pmatrix}$*; obtener* $\gamma_g(\mathbf{x}, \mathbf{y}, 2)$.

Como se puede ver, para este caso $m = n = 8$ y $\theta = 2$. Entonces, siguiendo los pasos del algoritmo:

1. Como $m = n$, $\mathbf{y}$ no se modifica.
2. $\mathbf{r}^1 = \alpha(\mathbf{x}, \mathbf{y}) = \begin{pmatrix}0\\2\\1\\1\\1\\1\\0\\2\end{pmatrix}$.
3. $\mathbf{r}^2 = \begin{pmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix}$.
4. $\mathbf{r}^3 = \begin{pmatrix}0\\0\\1\\1\\1\\1\\0\\0\end{pmatrix}$.
5. $r^4 = 8 - u_\beta(\mathbf{r}^3) = 8 - 4 = 4$.
6. Como $4 > 2$, $4 \not\leq 2$ y $\gamma_g(\mathbf{x}, \mathbf{y}, 2) = 0$.

■

**Ejemplo B.3** *Sean* $\mathbf{x} = \begin{pmatrix}1 & 0 & 0 & 1 & 0 & 1 & 1 & 0\end{pmatrix}$ *y* $\mathbf{y} = \begin{pmatrix}0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix}$*; obtener* $\gamma_g(\mathbf{x}, \mathbf{y}, 3)$.

En este caso, $m = n = 8$ y $\theta = 3$. Entonces $\mathbf{r}^1 = \begin{pmatrix}2 & 1 & 0 & 1 & 0 & 1 & 2 & 1\end{pmatrix}$, $\mathbf{r}^2 = \begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\0 & 1 & 0 & 1 & 0 & 1 & 0 & 1\end{pmatrix}$, $\mathbf{r}^3 = \begin{pmatrix}0 & 1 & 0 & 1 & 0 & 1 & 0 & 1\end{pmatrix}$, $r^4 = 8 - 4 = 4$; como $4 > 3$ entonces $4 \not\leq 3$, por lo que $\gamma_g(\mathbf{x}, \mathbf{y}, 3) = 0$.

■

**Ejemplo B.4** *Sean* $\mathbf{x} = \begin{pmatrix}1 & 0 & 0 & 1 & 1 & 1 & 1 & 0\end{pmatrix}$ *y* $\mathbf{y} = \begin{pmatrix}1 & 0 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix}$*; obtener* $\gamma_g(\mathbf{x}, \mathbf{y}, 3)$*.*

En este caso, $m = n = 8$ y $\theta = 3$. Entonces $\mathbf{r}^1 = \begin{pmatrix}1 & 1 & 0 & 1 & 1 & 1 & 2 & 1\end{pmatrix}$, $\mathbf{r}^2 = \begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1\end{pmatrix}$, $\mathbf{r}^3 = \begin{pmatrix}1 & 1 & 0 & 1 & 1 & 1 & 0 & 1\end{pmatrix}$, $r^4 = 8 - 6 = 2$; como $2 < 3$ entonces $2 \leq 3$ y $\gamma_g(\mathbf{x}, \mathbf{y}, 3) = 1$.
■

**Ejemplo B.5** *Sean* $\mathbf{x} = \begin{pmatrix}1 & 0 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix}$ *y* $\mathbf{y} = \begin{pmatrix}1 & 0 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix}$*; obtener* $\gamma_g(\mathbf{x}, \mathbf{y}, 1)$*.*

En este caso, $m = n = 8$ y $\theta = 1$. Entonces $\mathbf{r}^1 = \begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}$, $\mathbf{r}^2 = \begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}$, $\mathbf{r}^3 = \begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}$, $r^4 = 8 - 8 = 0$; como $0 < 1$ $0 \leq 3$, $\gamma(\mathbf{x}, \mathbf{y}, 3) = 1$.
■

**Ejemplo B.6** *Sean* $\mathbf{x} = \begin{pmatrix}0 \\ 1 \\ 1 \\ 0 \\ 0\end{pmatrix}$ *y* $\mathbf{y} = \begin{pmatrix}1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0\end{pmatrix}$*; obtener* $\gamma_g(\mathbf{x}, \mathbf{y}, 2)$*.*

En este caso $m = 8$, $n = 5$, $\theta = 2$. Entonces, $\mathbf{y}$ se trunca a $\mathbf{y} = \begin{pmatrix}0 \\ 0 \\ 1 \\ 1 \\ 0\end{pmatrix}$; ahora $\mathbf{r}^1 = \begin{pmatrix}1 \\ 2 \\ 1 \\ 0 \\ 1\end{pmatrix}$, $\mathbf{r}^2 = \begin{pmatrix}0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1\end{pmatrix}$, $\mathbf{r}^3 = \begin{pmatrix}1 & 0 & 1 & 0 & 1\end{pmatrix}$, $r^4 = m - u_\beta(\mathbf{r}^3) = 8 - 3 = 5$.
Como $5 > 2$, $5 \not\leq 2$ y $\gamma_g(\mathbf{x}, \mathbf{y}, 2) = 0$.
■

**Ejemplo B.7** *Sean* $\mathbf{x} = \begin{pmatrix}0 \\ 1 \\ 1 \\ 0 \\ 0\end{pmatrix}$ *y* $\mathbf{y} = \begin{pmatrix}1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0\end{pmatrix}$*; obtener* $\gamma_g(\mathbf{x}, \mathbf{y}, 2)$*.*

En este caso $m = 6$, $n = 5$, $\theta = 2$. Entonces, $\mathbf{y}$ se trunca a $\mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$; ahora $\mathbf{r}^1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$, $\mathbf{r}^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$, $\mathbf{r}^3 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \end{pmatrix}$, $r^4 = m - u_\beta\left(\mathbf{r}^3\right) = 6 - 4 = 2$.
Como $2 = 2$, $2 \leq 2$ y $\gamma_g(\mathbf{x}, \mathbf{y}, 2) = 1$.
■

# Apéndice C

# Teoremas de Tricotomía

El presente Apéndice está dedicado a tres Teoremas que fueron enunciados y probados durante el desarrollo de este trabajo de tesis. Estos Teoremas permiten caracterizar las relaciones de orden (menor que, igual que y mayor que) entre patrones binarios codificados con el código Johnson-Möbius modificado.

**Teorema C.1** *Sean: los conjuntos* $A = \{0, 1\}$ *y* $B = \{0, 1, 2\}$*, el número* $n \in \mathbb{Z}^+$*,* $\mathbf{x} \in A^n$ *y* $\mathbf{y} \in A^n$ *dos vectores binarios n-dimensionales codificados con el código Johnson-Möbius modificado, con la k-ésima componente representada por* $x_k$ *y* $y_k$*, respectivamente, que representan al código Johnson-Möbius modificado correspondiente a los números* $a, b \in \mathbb{Z}^+$*; y además i y j dos índices enteros no negativos. Si el vector n-dimensional* $\mathbf{z} \in B^n$*, cuya k-ésima componente está representada por* $z_k$*, es igual al resultado de operar los vectores* $\mathbf{x}$ *y* $\mathbf{y}$ *con el operador* $\alpha$*, esto es* $\mathbf{z} = \alpha(\mathbf{x}, \mathbf{y})$*, entonces a es menor que b si y sólo si existe un índice i tal que la i-ésima componente de* $\mathbf{z}$ *tenga el valor de* 0*:*

$$a < b \iff \exists i \; z_i = 0$$

**Demostración.** Dado que se trata de una doble implicación, ésta se demostrará en dos pasos: primero en un sentido y posteriormente en el otro sentido.

□ $a < b \longrightarrow \exists i \; z_i = 0$

Dado que ambos vectores $\mathbf{x}$ y $\mathbf{y}$ están codificados con el código Johnson-Möbius modificado, se sabe que $u_\beta(\mathbf{x}) = a$ y $u_\beta(\mathbf{y}) = b$; como $a < b$ tenemos que $u_\beta(\mathbf{x}) < u_\beta(\mathbf{y})$.

Ahora bien, sean: los índices $i = n - u_\beta(\mathbf{x})$ el ídice de mayor valor para el cual $x_i = 0$ y $j = n - u_\beta(\mathbf{y})$ el ídice de mayor valor para el cual $y_j = 0$, respectivamente. Entonces si $k > i \longrightarrow x_k = 1 \; \forall k = 1, 2, \cdots, n$; y, similarmente, si $k > j \longrightarrow y_k = 1 \; \forall k = 1, 2, \cdots, n$.

Como $u_\beta(\mathbf{x}) < u_\beta(\mathbf{y})$, es claro que $i > j$, y tomando en cuenta cómo se eligieron los índices $i$ y $j$, resulta que $x_i = 0$ y $y_i = 1$.

Luego, se sigue de la definición del operador $\alpha$ que $z_i = \alpha(\mathbf{x}_i, \mathbf{y}_i) = \alpha(0, 1) = 0$.

Por lo tanto, $\exists i \; z_i = 0$.

□ $\exists i \; z_i = 0 \longrightarrow a < b$

Por la definición del operador $\alpha$, $z_i = 0 \longrightarrow x_i = 0$ y $y_i = 1$.

Asimismo, se sabe que tanto $\mathbf{x}$ como $\mathbf{y}$ están codificados con el código Johnson-Möbius modificado, por lo que $x_{i+1}$ puede ser otra componente con valor de 0 o bien, la primera componente de $\mathbf{x}$ con valor de 1, mientras que $y_{i-1}$ puede ser otra componente con valor de 1 o bien, la última componente de $\mathbf{y}$ con valor de 0.

Por lo anterior y tomando en cuenta que $x_i = 0$, es claro que $n - i \geq u_\beta(\mathbf{x})$ y, similarmente, como $y_i = 1$ es claro que $n-i+1 \leq u_\beta(\mathbf{y})$. Pero resulta que $n-i \geq u_\beta(\mathbf{x})$ también se puede expresar como $u_\beta(\mathbf{x}) \leq n - i$.

Ahora bien, como $\mathbf{x}$ y $\mathbf{y}$ están codificados con el código Johnson-Möbius modificado, se sabe que $u_\beta(\mathbf{x}) = a$ y $u_\beta(\mathbf{y}) = b$, por lo que se puede afirmar que $a \leq n - i$ y $n - i + 1 \leq b$. Es claro también que $n - i < n - i + 1$.

Pero por transitividad, si $a \leq n - i$ y $n - i < n - i + 1$, entonces $a < n - i + 1$, y como además $n - i + 1 \leq b$, por transitividad tenemos que $a < b$.

■ ■

**Teorema C.2** *Sean: los conjuntos $A = \{0, 1\}$ y $B = \{0, 1, 2\}$, el número $n \in \mathbb{Z}^+$, $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^n$ dos vectores binarios n-dimensionales codificados con el código Johnson-Möbius modificado, con la k-ésima componente representada por $x_k$ y $y_k$, respectivamente, que representan al código Johnson-Möbius modificado correspondiente a los números $a, b \in \mathbb{Z}^+$; y además i y j dos índices enteros no negativos. Si el vector n-dimensional $\mathbf{z} \in B^n$, cuya k-ésima componente está representada por $z_k$, es igual al resultado de operar los vectores $\mathbf{x}$ y $\mathbf{y}$ con el operador $\alpha$, esto es $\mathbf{z} = \alpha(\mathbf{x}, \mathbf{y})$, entonces a es igual que b si y sólo si para todo índice i, la i-ésima componente de $\mathbf{z}$ tiene el valor de 1:*

$$a = b \iff z_i = 1 \ \forall i$$

**Demostración.** Dado que se trata de una doble implicación, ésta se demostrará en dos pasos: primero en un sentido y posteriormente en el otro sentido.

□ $a = b \longrightarrow z_i = 1 \ \forall i$

Como los vectores $\mathbf{x}$ y $\mathbf{y}$ son la codificación con el código Johnson-Möbius modificado de los valores $a$ y $b$, respectivamente, se sabe que si $a = b$ entonces $\mathbf{x} = \mathbf{y}$.

Ahora bien, dado que $\mathbf{z} = \alpha(\mathbf{x}, \mathbf{y})$ se tiene que $z_i = \alpha(x_i, y_i) \ \forall i$, y como $\mathbf{x} = \mathbf{y}$ son dos vectores binarios, existen dos posibilidades: $x_i = y_i = 0$ o $x_i = y_i = 1$. Así, $x_i = y_i = 0$ entonces $z_i = \alpha(x_i, y_i) = z_i = \alpha(0, 0) = 1$, mientras que si $x_i = y_i = 1$ entonces $z_i = \alpha(x_i, y_i) = z_i = \alpha(1, 1) = 1$. Como esto sucede, se tiene que $z_i = 1 \ \forall i$, y por lo tanto $a = b \longrightarrow z_i = 1 \ \forall i$.

□ $z_i = 1 \ \forall i \longrightarrow a = b$

Por la forma en la que está calculada cada componente del vector $\mathbf{z}$, a saber $z_i = \alpha(x_i, y_i) \ \forall i$, el hecho de que $z_i = 1 \ \forall i$ indica que se tienen dos casos: o bien $z_i = 1 = \alpha(x_i, y_i) = \alpha(0, 0) \ \forall i$, o bien $z_i = 1 = \alpha(x_i, y_i) = \alpha(1, 1) \ \forall i$. En ambos casos es claro que, para que lo anterior suceda, es necesario que $x_i = y_i \ \forall i$, lo que lleva a que $\mathbf{x} = \mathbf{y}$.

Ahora bien, como se sabe que los vectores $\mathbf{x}$ y $\mathbf{y}$ son la codificación con el código Johnson-Möbius modificado de los valores $a$ y $b$, respectivamente, se sabe que si $\mathbf{x} = \mathbf{y}$ entonces $a = b$.

Por lo tanto $z_i = 1 \ \forall i \longrightarrow a = b$.

■ ■

**Teorema C.3** *Sean: los conjuntos* $A = \{0,1\}$ *y* $B = \{0,1,2\}$*, el número* $n \in \mathbb{Z}^+$*,* $\mathbf{x} \in A^n$ *y* $\mathbf{y} \in A^n$ *dos vectores binarios n-dimensionales codificados con el código Johnson-Möbius modificado, con la k-ésima componente representada por* $x_k$ *y* $y_k$*, respectivamente, que representan al código Johnson-Möbius modificado correspondiente a los números* $a, b \in \mathbb{Z}^+$*; y además i y j dos índices enteros no negativos. Si el vector n-dimensional* $\mathbf{z} \in B^n$*, cuya k-ésima componente está representada por* $z_k$*, es igual al resultado de operar los vectores* $\mathbf{x}$ *y* $\mathbf{y}$ *con el operador* $\alpha$*, esto es* $\mathbf{z} = \alpha(\mathbf{x}, \mathbf{y})$*, entonces a es mayor que b si y sólo si existe un índice j tal que la j-ésima componente de* $\mathbf{z}$ *tenga el valor de* 2*:*

$$a > b \iff \exists j \; z_j = 2$$

**Demostración.** Dado que se trata de una doble implicación, ésta se demostrará en dos pasos: primero en un sentido y posteriormente en el otro sentido.

□ $a > b \longrightarrow \exists i \; z_i = 2$

Dado que ambos vectores $\mathbf{x}$ y $\mathbf{y}$ están codificados con el código Johnson-Möbius modificado, se sabe que $u_\beta(\mathbf{x}) = a$ y $u_\beta(\mathbf{y}) = b$; como $a > b$ tenemos que $u_\beta(\mathbf{x}) > u_\beta(\mathbf{y})$.

Ahora bien, sean: los índices $i = n - u_\beta(\mathbf{x})$ el ídice de mayor valor para el cual $x_i = 0$ y $j = n - u_\beta(\mathbf{y})$ el ídice de mayor valor para el cual $y_j = 0$, respectivamente. Entonces si $k > i \longrightarrow x_k = 1 \; \forall k = 1, 2, \cdots, n$; y, similarmente, si $k > j \longrightarrow y_k = 1 \; \forall k = 1, 2, \cdots, n$.

Como $u_\beta(\mathbf{x}) > u_\beta(\mathbf{y})$, es claro que $i < j$, y tomando en cuenta cómo se eligieron los índices $i$ y $j$, resulta que $x_j = 1$ y $y_j = 0$.

Luego, se sigue de la definición del operador $\alpha$ que $z_j = \alpha(\mathbf{x}_j, \mathbf{y}_j) = \alpha(1,0) = 2$.

Por lo tanto, $\exists j \; z_j = 2$.

□ $\exists j \; z_j = 2 \longrightarrow a > b$

Por la definición del operador $\alpha$, $z_j = 2 \longrightarrow x_j = 1$ y $y_j = 0$.

Asimismo, se sabe que tanto $\mathbf{x}$ como $\mathbf{y}$ están codificados con el código Johnson-Möbius modificado, por lo que $x_{j-1}$ puede ser otra componente con valor de 1 o bien, la última componente de $\mathbf{x}$ con valor de 0, mientras que $y_{j+1}$ puede ser otra componente con valor de 0 o bien, la primera componente de $\mathbf{y}$ con valor de 1.

Por lo anterior y tomando en cuenta que $x_j = 1$, es claro que $n - j + 1 \leq u_\beta(\mathbf{x})$, lo que también se puede expresar como $u_\beta(\mathbf{x}) \geq n - j + 1$; similarmente, como $y_j = 0$ es claro que $n - j \geq u_\beta(\mathbf{y})$.

Ahora bien, como $\mathbf{x}$ y $\mathbf{y}$ están codificados con el código Johnson-Möbius modificado, se sabe que $u_\beta(\mathbf{x}) = a$ y $u_\beta(\mathbf{y}) = b$, por lo que se puede afirmar que $a \geq n - j + 1$ y $n - j \geq b$. Es claro también que $n - j + 1 > n - j$.

Pero por transitividad, si $a \geq n - j + 1$ y $n - j + 1 > n - j$, entonces $a > n - j$, y como además $n - j \geq b$, por transitividad tenemos que $a > b$.

■ ■

# Apéndice D

# Manual del Software

El contenido de este Apéndice está integrado por el manual de usuario del *software* creado *ex-profeso* para probar el algoritmo propuesto en este trabajo de tesis, Clasificador Gama versión 1.3. Este software fue diseñado y desarrollado en Borland C++ Builder 6.0 y puede ser ejecutado en los sistemas operativos Windows de 32 y 64 bits.

## D.1. Clasificador Gama Versión 1.3

El área de trabajo de Clasificador Gama está dividad en 4 principales áreas:

- **Barra de menús**. Aquí se presentan las principales acciones a realizar, organizadas en 3 menús:
  - **Archivo**. Incluye las acciones **Abrir...**, **Guardar...**, **Cerrar**, que se refieren a acciones sobre un archivo de datos, y **Salir**, que termina el programa (ver Figura D.2).
  - **Herramientas**. Incluye las opciones **Clasificar** y **Comparar**, que necesitan se hayan generado el conjunto fundamentel y el conjunto de prueba a partir del archivo de datos, y las opciones **Corridas** y **Leave-One-Out**, que se refieren a conjuntos de experimentos (ver Figura D.3).
  - **Ayuda**. Incluye las opciones **Contenido**, que muestra el presente archivo, y **Acerca de...**, que muestra una ventana con información general acerca del programa (ver Figuras D.4 y D.5).

- **Panel de opciones**. Es en esta área donde se especifican los parámetors del experimento a realizar; se encuentra dividada a su vez en tres áreas:
  - **Selección del C. F.** Está dedicada a determinar la manera en que se seleccionarán los patrones que pertenecerán al conjunto fundamental, siendo los restantes asignados al conjunto de prueba.
  - **Herramientas**. Aquí se presentan botones de acceso a las acciones presentes en el menú **Herramientas**.

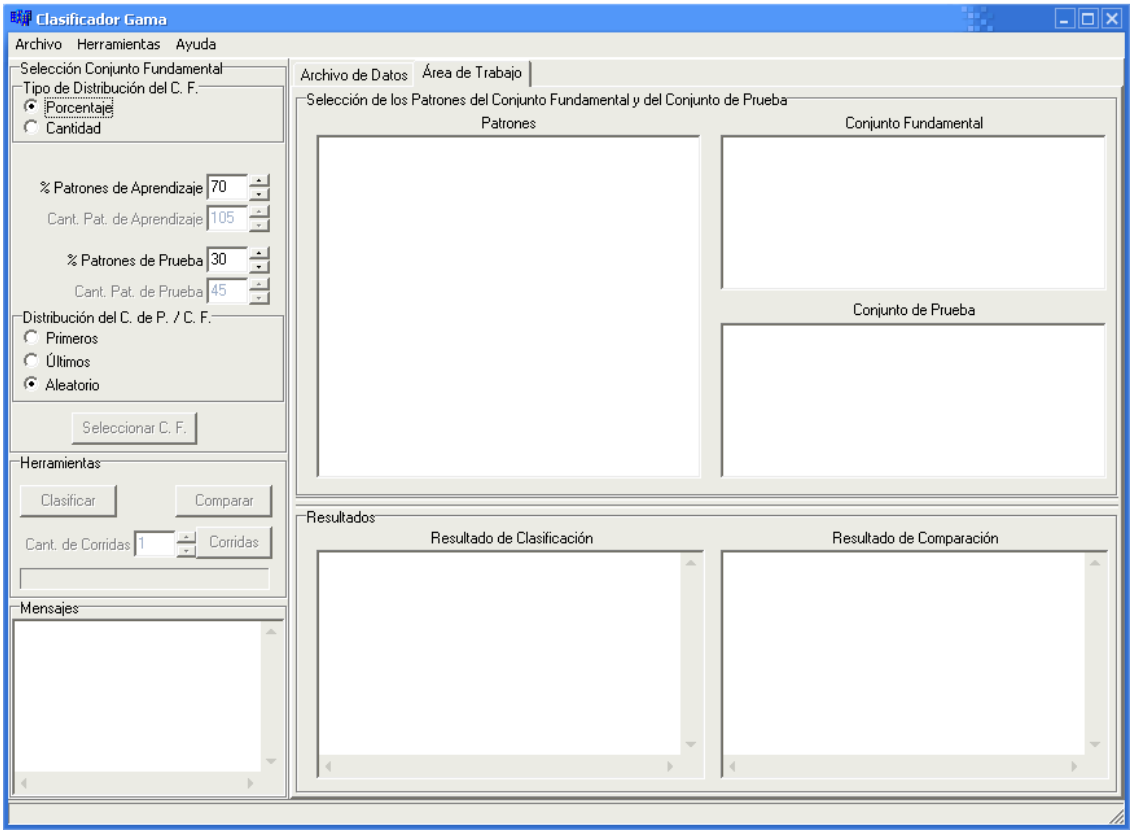



Figura D.1: Clasificador Gama 1.3

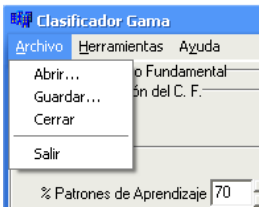



Figura D.2: Menú Archivo

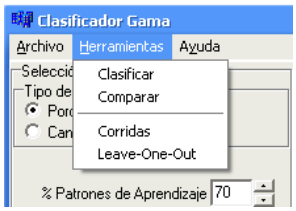



Figura D.3: Menú Herramientas

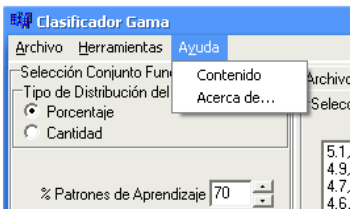



Figura D.4: Menú Ayuda

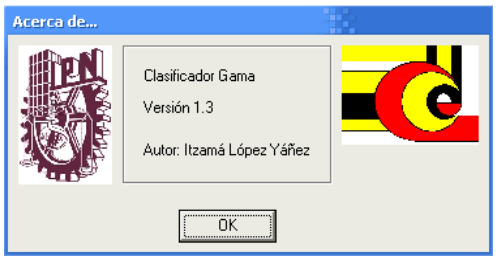



Figura D.5: Ventana Acerca de...

- **Mensajes**. En esta área se muestran mensajes generales del programa.

- **Panel de trabajo**. Es aquí donde se muestra el avance de cada experimento. Está dividido en el área de **Archivo de Datos**, que muestra el contenido del archivo de datos que actualmente se encuentre abierto, y el **Área de Trabajo**, donde se puede ver el avance del experimento. Esta área está dividida a su vez en dos áreas:

  - **Selección de los Patrones del Conjunto Fundamental y del Conjunto de Prueba**. Aquí se muestran los patrones del conjunto de datos, así como aquellos que pertenecen al conjunto de prueba.
  - **Resultados**. Aquí se muestran los resultados de la clasificación y de la comparación de dicha clasificación con la información incliuda en el archivo de datos.

- **Barra de status**. Aquí se muestran mensajes cortos que indican la acción que realiza el programa en un momento dado, o la acción que ha terminado.

Para llevar a cabo los experimentos, se pueden seguir dos métodos:

- Realizar un experimento único.
- Realizar un conjunto de experimentos.

El primer caso es descrito en la siguiente Sección, mientras que el último caso se explica en la Sección 3.

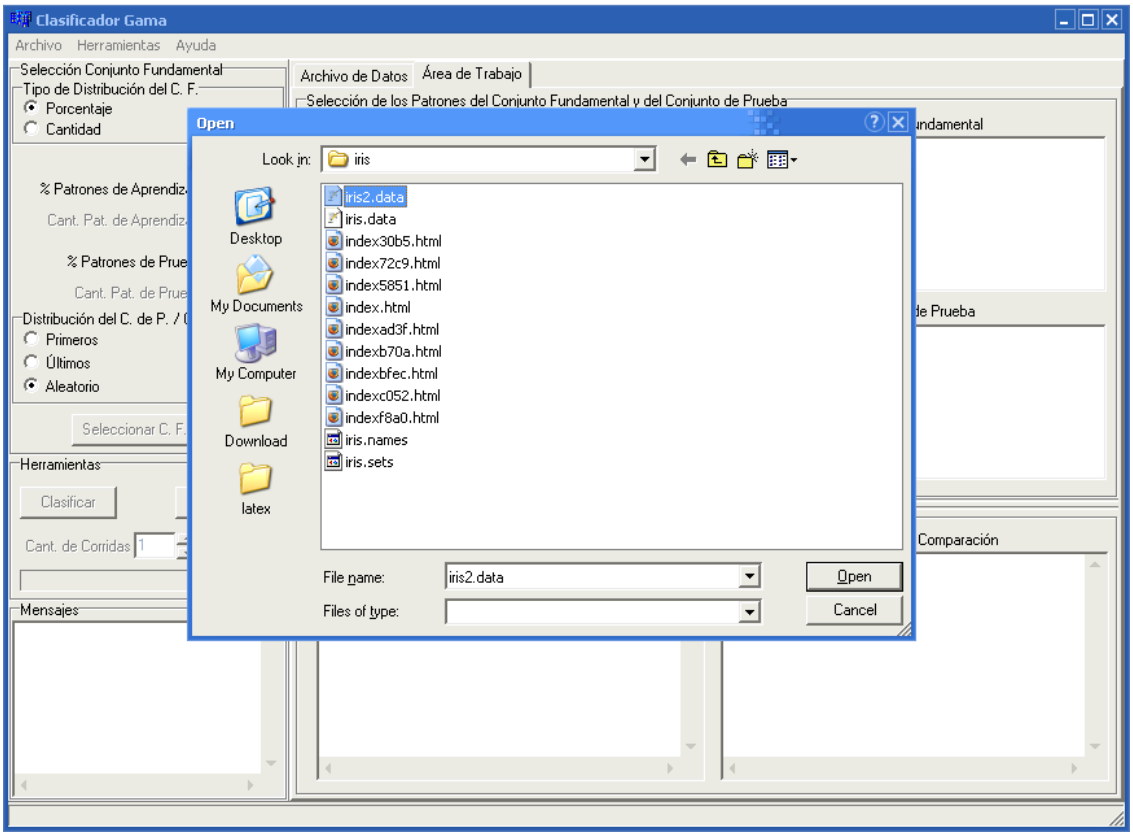



Figura D.6: Abrir un archivo de datos

## D.2. Experimento Único

Para realizar un experimento único se deben seguir los siguientes pasos:

1. Abrir el archivo de datos (ver Figuras D.6 y D.7).
   - Seleccionar el menú **Archivo** | **Abrir...**.
   - Seleccionar el archivo de datos a utilizar.
   - Dar click en el botón **Abrir**.
2. Seleccionar la forma en que se seleccionará el conjunto fundamental.
   - Elegir si el conjunto fundamental será seleccionado usando un parámetro de Porcentaje o de Cantidad.
   - Elegir los parámetors correspondientes a cada conjunto: fundamental y de prueba.
   - Elegir cómo serán tomados los patrones del conjunto fundamental: los Primeros $p$, los Últimos $p$, o Aleatoriamente.

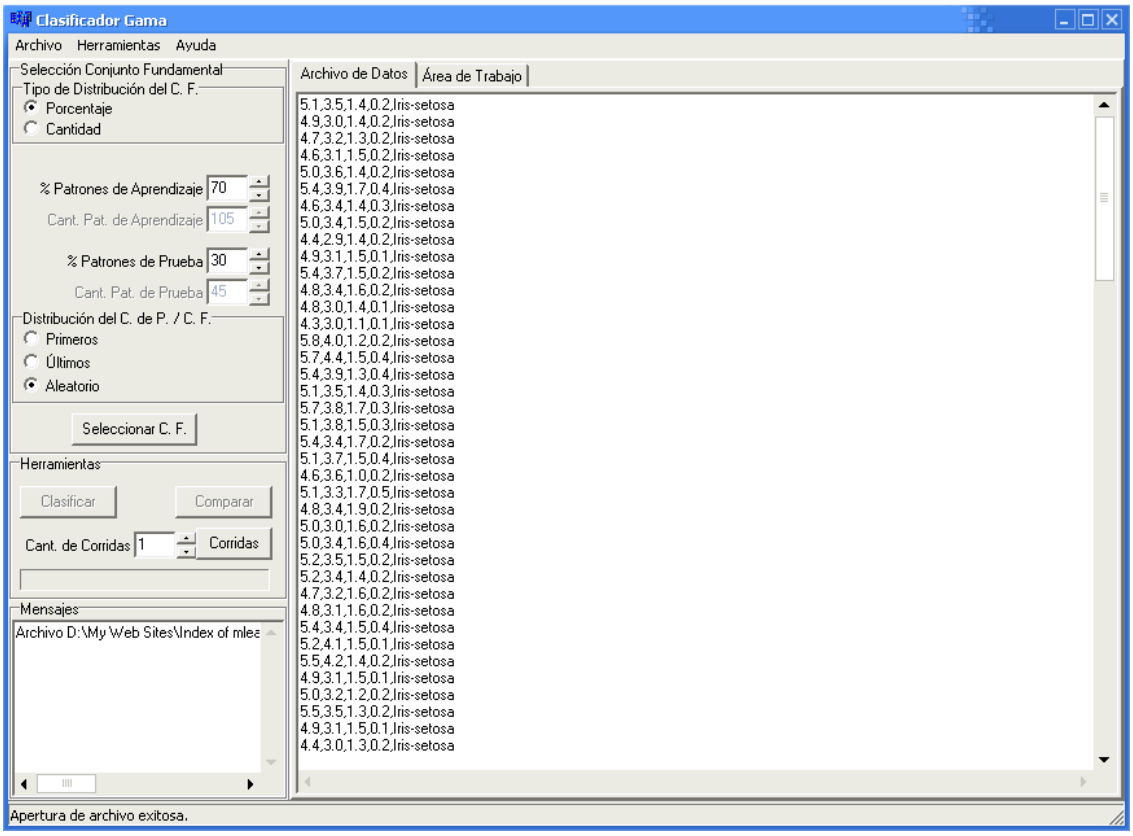



Figura D.7: Contenido del archivo de datos abierto

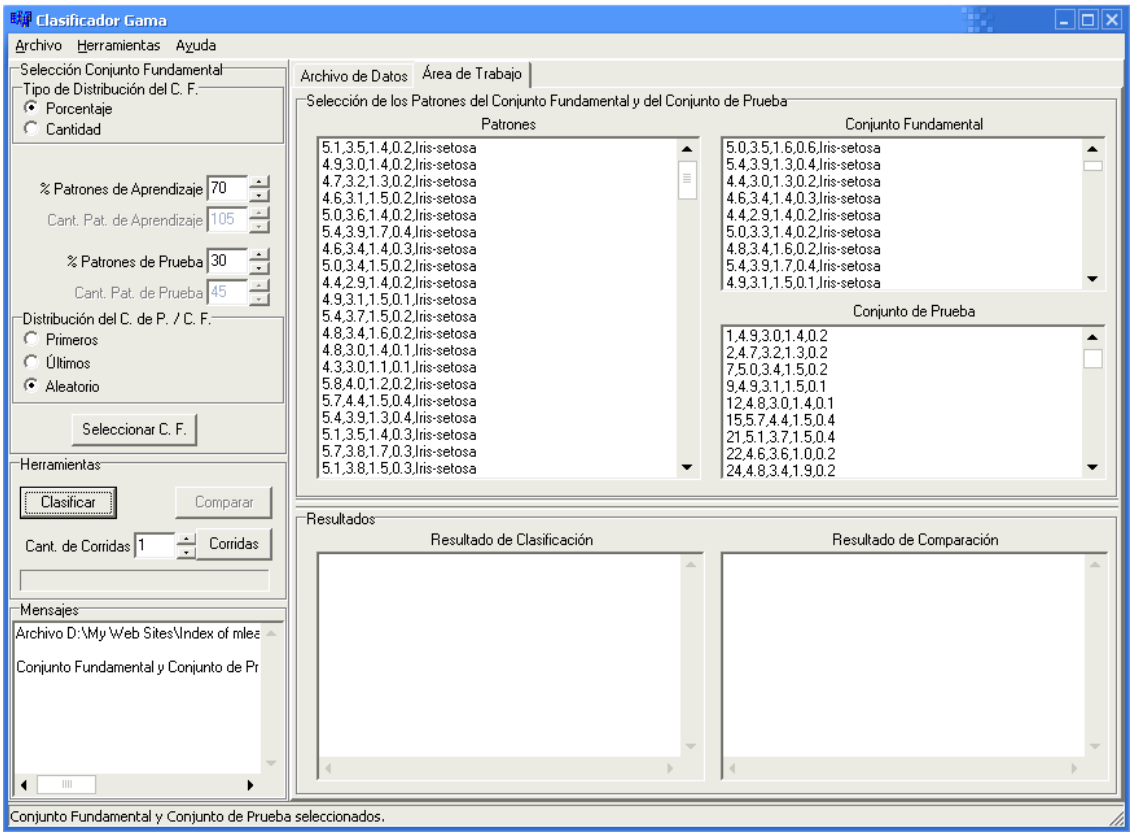



Figura D.8: Generar el conjunto fundamental

3. Generar el conjunto fundamental (ver Figura D.8).
   - Dar click en el botón **Seleccionar C. F.**
4. Clasificar los patrones del conjunto de prueba (ver Figura D.9).
   - Dar click en el botón **Clasificar**, o seleccionar el menú **Herramientas** | **Clasificar**.
5. Comparar los resultados de la clasificación con los datos del archivo de datos (ver Figura D.10).
   - Dar click en el botón **Comparar**, o seleccionar el menú **Herramientas** | **Comparar**.
6. El porcentaje de clasificación correcta se muestra en el área de **Mensajes**.

## D.3. Conjunto de Experimentos

Para realizar un conjunto de experimentos se deben seguir los siguientes pasos:

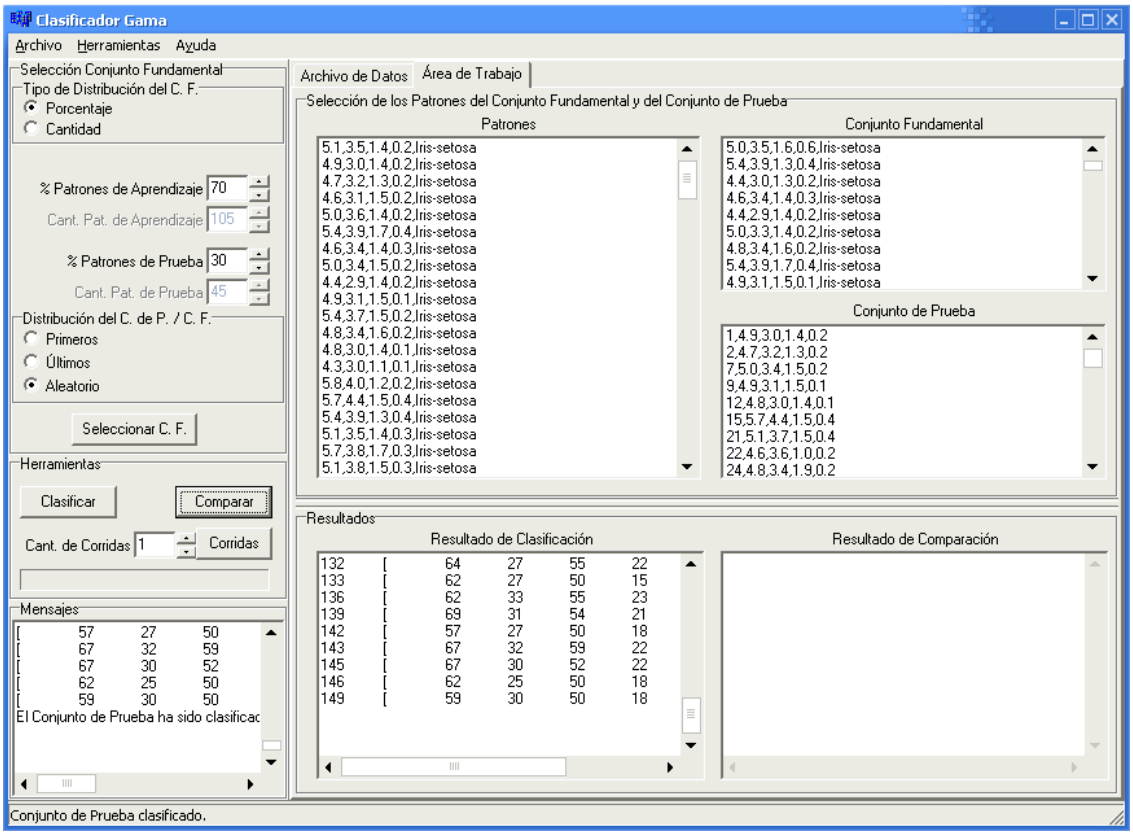



Figura D.9: Clasificar los patrones del conjunto de prueba

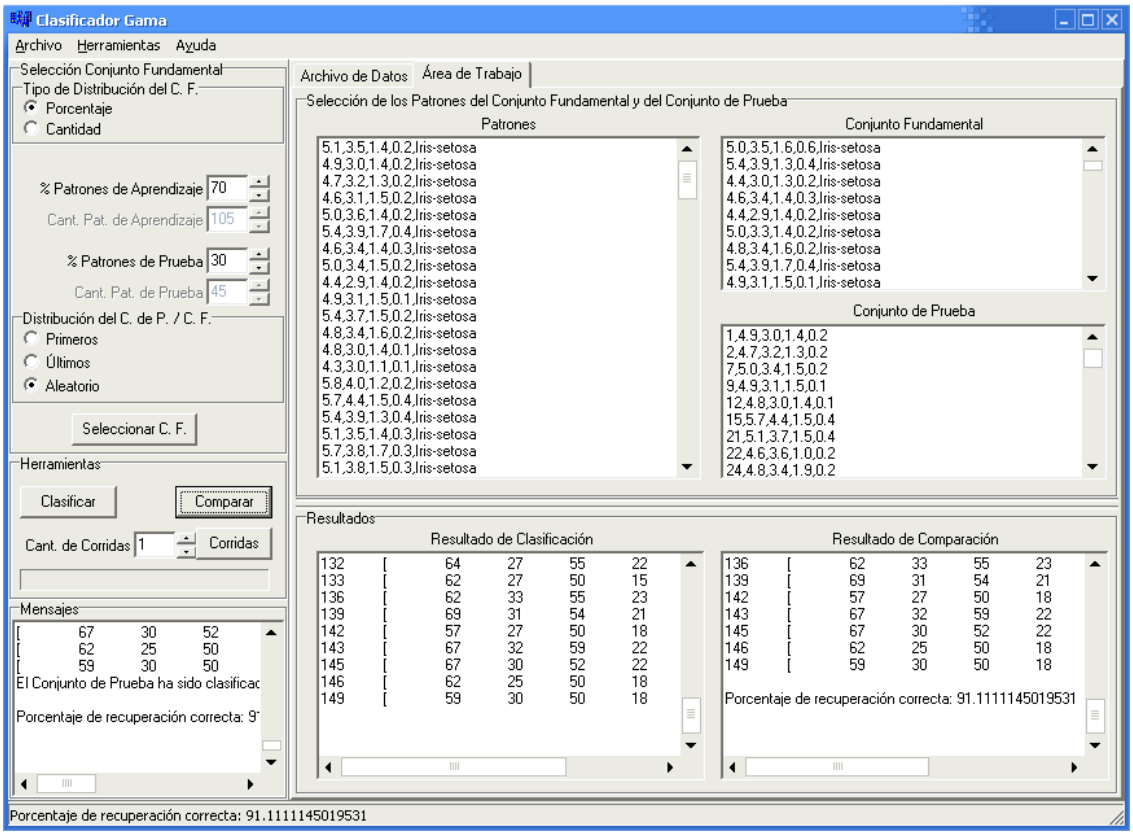



Figura D.10: Clasificar los patrones del conjunto de prueba

1. Abrir el archivo de datos (ver Figuras D.6 y D.7).

   - Seleccionar el menú **Archivo** | **Abrir...**.
   - Seleccionar el archivo de datos a utilizar.
   - Dar click en el botón **Abrir**.

En caso de que se desee realizar un conjunto de corridas, los pasos restantes son los que acontinuación se muestran.

2. Seleccionar la forma en que se seleccionará el conjunto fundamental.

   - Elegir si el conjunto fundamental será seleccionado usando un parámetro de Porcentaje o de Cantidad.
   - Elegir los parámetors correspondientes a cada conjunto: fundamental y de prueba.
   - Elegir cómo serán tomados los patrones del conjunto fundamental: los Primeros $p$, los Últimos $p$, o Aleatoriamente.

3. En caso de que se desee realizar un conjunto de corridas, indicar el número de corridas.

4. Realizar los experimentos (ver Figura D.11).

   - Dar click en el botón **Corridas** o seleccionar el menú **Herramientas** | **Corridas**.

5. Los 20 mejores porcentajes de clasificación correcta se muestra en el área de **Mensajes**.

En caso de que se desee realizar un experimento con la metodología **Leave-One-Out**, los pasos restantes son los que acontinuación se muestran.

2. Realizar el experimento.

   - Seleccionar el menú **Herramientas** | **Leave-One-Out**.

3. El porcentaje de clasificación correcta se muestra en el área de **Mensajes**.

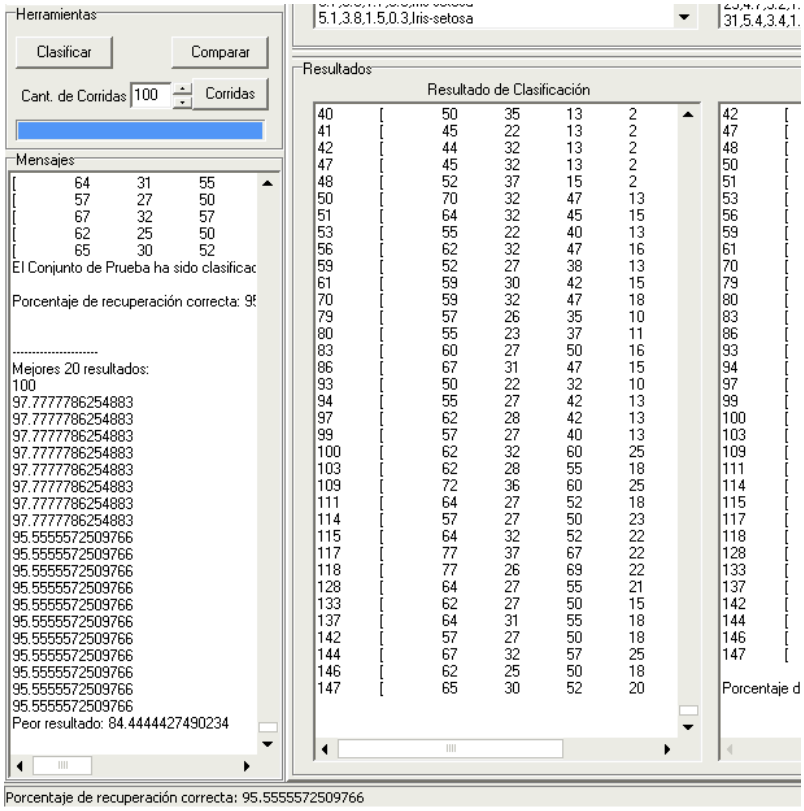



Figura D.11: Resultado de 100 corridas

# Bibliografía


[1] Sánchez-Garfias, F.A., Díaz-de-León, J.L., Yáñez-Márquez, C.: Reconocimiento automático de patrones. Conceptos básicos, IT 79, Serie Verde, Centro de Investigación en Computación, IPN, México (2003)

[2] Marqués de Sá, J.P.: Pattern Recognition, Concepts, Methods and Application. Germany, Springer (2001)

[3] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, USA.(2001)

[4] Aldape-Pérez, M., Yáñez-Márquez, C. , López -Leyva, L.O.: Feature Selection using a Hybrid Associative Classifier with Masking Technique. En: IEEE Computer Society, Proc. Fifth Mexican International Conference on Artificial Intelligence, MICAI 2006 pp. 151-160. ISBN: 0-7695-2722

[5] Webb, A.: Statistical Pattern Recognition. Oxford University Press, USA (1999)

[6] Kuncheva, L.I.: A theoretical Study on Six Classifier Fusion Strategies. IEEE Transations on Pattern Analysis and Machine Intellegence, vol. 24, no. 2 (2002) 281-286

[7] Santiago-Montero, R., Yáñez-Márquez, C., Diaz-de-León, J. L.: Clasificador híbrido de patrones basado en la Lenmarix de Steinbuch y el Linear Associator de Anderson-Kohonen. Research on Computing Science. Reconocimiento de patrones, avances y perspectivas. Centro de Investigación en Computación, IPN, México (2002) 449-460

[8] Sánchez-Garfias, F.A: Condiciones necesarias y suficientes para recuperación perfecta de patrones. Lernmatrix de Steinbuch. Tesis de Maestría en Ciencias de la Computación. CIC IPN, México (2004)

[9] Friedman, M., Kandel, A.: Introduction to Pattern Recognition (Statistical, Structural, Neural and Fuzzy Logic Approaches). Singapore, World Scientific (2000)

[10] Sarünas, R.: Statistical and Neural Classifiers, An integrated Approach to disign. MIT Press, England (2001)

[11] Schürmann, J.: Pattern classification, A unified view of statistical and neural approaches. John Wiley, USA.(1996)

[12] Shalkoff, R.: Pattern recognition, Statistical, Structural and Neural Approaches. John Wiley, USA.(1992)

[13] Michie,D., D.J. , Spiegelhalter, Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood, Chichester, England (1994)

[14] Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annual Eugenics,vol. 7, part II (1936) 179-188

[15] Lu, Y., Tan, C.L.: Combination of multiple classifiers using probabilistic dictionary and its application to postcode recognition, Pattern Recognition, vol. 35 (2002) 2823-2832

[16] Rueda, L., Oommen, B.J.: On optimal pairwise linear classifiers for normal distributions the two-dimensional case. IEEE Transations on Pattern Analysis and Machine Intellegence, vol. 24, no. 2 (2002) 274-273

[17] Díaz-de-León, J.L., Yáñez-Márquez, C., Sánchez-Garfias, F.A.: Reconocimiento de patrones. Enfoque probabilístico-estadístico. IT 83, Serie Verde, Centro de Investigación en Computación, IPN, México (2003)

[18] Cover, T.M., Hart, P.E.: Nearest Pattern Classification. IEEE Trans. on Information Theory, vol. IT-13, no. 1 (1967) 21-27

[19] Bandyopadhyay, S., Maulik, U.: Efficient prototype reordering in nearest neighbor classifation. Pattern Recognition, vol. 35 (2002) 2791-2799

[20] Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NNpattern Classification Techniques. IEEE Computer Society Press, USA (1991)

[21] Demeniconi, C., Peng, J., Gunopulos, D.: Locally Adaptive Metric Nearest-Neighbor Classification. IEEE Transations on Pattern Analysis and Machine Intellegence, vol. 24, no. 9 (2002) 1281-1285

[22] Ho, S.Y., Liu, C.C., Liu, S.: Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. Pattern Recognition Letters, vol. 23 (2002) 1495-1503

[23] Huang, Y.S., Chiang, C.C., Shieh, J. W., Grimson, E.: Prototype optimization for nearest-neighbor classification. Pattern Recognition, 35 (2002), 1237-1245

[24] Wu, Y., Ianekiev, K., Govindaraju, V.: Improved k-nearest neighbor classification. Pattern Recognition, vol. 35 (2002) 2311-2318

[25] Díaz-de-León, J.L., Yáñez-Márquez, C., Sánchez-Garfias, F.A.: Clasificador euclideano de patrones. IT 80, Serie Verde, Centro de Investigación en Computación, IPN, México (2003)

[26] Flores Carapia, R., Yáñez Márquez, C.: Minkowski's Metrics-Based k-NN Classifier Algorithm: A Comparative Study. Research on Computing Science, Vol. 14, IPN México (2005) pp. 191-202. ISSN 1665-9899

[27] Flores Carapia, R., Yáñez Márquez, C.: Minkowski's Metrics-Based Classifier Algorithm: A Comparative Study. En: Memoria del XIV Congreso Internacional de Computación CIC IPN, México (2005) pp. 304-315. ISBN: 970-36-0267-3

[28] Muñoz Torija, J.M.,Yáñez Márquez, C: Un Estudio Comparativo del Perceptron y el Clasificador Euclideano. En: Memoria del XIV Congreso Internacional de Computación CIC IPN México (2005) pp. 316-326. ISBN: 970-36-0267-3

[29] Anil, K. Jain, Robert, P. W., Duin, Mao, Jianchang Mao: Statistical Pattern Recognition. A Review:. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, num. 1 (2000) 4-37

[30] Yáñez-Márquez, C., Diaz de León, J.L., Sánchez-Garfias, F.A.: Reconocimiento de patrones. Enfoque sintáctico-estructural. IT 84, Serie Verde, Centro de Investigación en Computación, IPN, México (2003)

[31] Gonzalez, R.C., Thomason, M.G.: Syntactic Pattern Recognition: an Introduction. Addison Wesley (1982)

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

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

[34] Pal, S.: Neuro-Fuzzy, Pattern Recognition: Methods in Soft Computing. USA, John Wiley & Sons (1999)

[35] Pandya, A.S.: Pattern recognition with neural networks in C++. Springer-Verlag, Great Britain (1996)

[36] Abe, S.: Pattern classification, Neuro-Fuzzy Methods and their Comparison. Springer-Verlag, Grat Britain (2001)

[37] Acharya, U.R., Bhat, P.S., Iyengar, S.S., Rao, R., Dua, S.: Classification of heart rate data using artificial neural network and fuzzy equivalence relation. Pattern Recognition, vol. 36, issue 1 (2003) 61-81

[38] Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, vol. 79 (1982) 2554-2558

[39] Hopfield, J.J.: Neurons with graded respose have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences, vol. 81 (1984) 3088-3092

[40] Anderson, J.A., Rosenfeld, E(eds.): Neurocomputing: Fundations of Research. MIT Press, Cambridge (1990)

[41] Anderson, J.A., Silverstein, J., Ritz, S., Jones, R.: Distinctive features, categorical perception, and probability learning: some applications of a neural model. Psichological Review, vol. 84 (1977) 413-451

[42] Haykin, S.: Neural Networks, A Comprehensive Foundation. Prentice Hall, USA (1999)

[43] Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge (1995)

[44] Kishan, M., Chilukuri, K.M., Sanjay, R.: Elements of Artificial Neural Networks. MIT Press, USA (1997)

[45] Minsky, M., Papert, S.: Perceptrons. MIT Press, Cambridge (1969)

[46] Rumelhart, D.E., Hinton, G. E., Williams, R.J.: Learning internal representation by Backpropagating errors. Nature, 323 (1986) 533–536

[47] Lecun, Y.: Une Procedure d´apprentissage pour reseau a seuil assymetrique cognitiva 85 : A la Frontiere de l'Intelligence Artificielle des Sciences de la Connaissance des Neurosciences, Paris, France. (1985) 559–604

[48] Krauth, W., Mezard, M.: Learning algorithms with optimal stability in neural networks. J. Phys. A: Math. Gen, vol. 20:l745 1987

[49] F. Rosenblatt.: Principles of Neurodynamics. Spartam Books, New York (1992)

[50] Ritter, G. X., Sussner, P. : An Introduction to Morphological Neural Networks. in Proceedings of the 13th International Conference on Pattern Recognition, vol. IV, Track D (1996) 709-717

[51] Argüelles, A.J., Yáñez, C., Díaz-de-León Santiago, J.L, Camacho, O.: Pattern recognition and classification using weightless neural networks and Steinbuch Lernmatrix. en: Proc. Optics & Photonics Conference 5916 Mathematical Methods in Pattern and Image Analysis, SPIE , San Diego, CA.(2005) pp. (59160)P1-P8. ISBN: 0-8194-5921-6, ISSN: 0277-786X

[52] Santiago-Montero, R.: Clasificador híbrido de patrones basado en la Lernmatrix de Steinbuch y el Linear Associator de Anderson-Kohonen. Tesis de Maestría en: Ciencias de la Computación CIC IPN, México (2003)

[53] Yáñez-Márquez, C.: Memorias Asociativas Basadas en Relaciones de Orden y Operadores Binarios. Tesis doctoral, Centro de Investigación en Computación, IPN, México (2002)

[54] Acevedo-Mosqueda, M.E.: Memorias Asociativas Bidireccionales Alfa-Beta. Tesis doctoral, Centro de Investigación en Computación, IPN, México (2006)

[55] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C., López-Yáñez, I.: Complexity of Alpha-Beta Bidirectional Associative Memories. Lecture Notes in Computer Science (Revista internacional ISI), LNCS 4293, Springer-Verlag Berlin Heidelberg (2006) pp. 357-366. ISSN: 0302-9743

[56] Flores-Carapia, R.: Memorias asociativas Alfa-Beta basadas en el código Johnson-Möbius modificado. Tesis de Maestría en: Ciencias de la Computación. CIC IPN, México (2006)

[57] Yáñez-Márquez, C., Felipe-Riverón, E.M., López-Yáñez, I., Flores-Carapia, R.: A Novel Approach to Automatic Color Matching. Lecture Notes in Computer Science (Revista internacional ISI), LNCS 4225, Springer-Verlag, Berlin Heidelberg (2006) pp. 529-538. ISSN: 0302-9743

[58] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C., López-Yáñez, I.: A New Model of BAM: Alpha-Beta Bidirectional Associative Memories. Lecture Notes in Computer Science (Revista internacional ISI), LNCS 4263, Springer-Verlag Berlin Heidelberg (2006) pp. 286-295. ISSN: 0302-9743

[59] Sánchez Garfias, F.A., Díaz-de-León Santiago, J.L.,Yáñez Márquez, C: Lernmatrix de Steinbuch: avances teóricos, Computación y Sistemas (Revista Iberoamericana de Computación incluida en el Índice CONACyT), Vol. 7, No. 3, México (2004) pp. 175-189. ISSN 1405-5546

[60] Yáñez Márquez, C., Díaz-de-León Santiago, J.L.: Memorias Asociativas Basadas en Relaciones de Orden y Operaciones Binarias. Computación y Sistemas (Revista Iberoamericana de Computación incluida en el Índice de CONACyT), Vol. 6, No. 4, México (2003) pp. 300-311. ISSN 1405-5546

[61] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C., López-Yáñez, I.: Alpha-Beta Bidirectional Associative Memories. IJCIR International Journal of Computational Intelligence Research, Vol. 3, No. 1, México (2006) pp. 105-110. ISSN: 0973-1873

[62] Acevedo-Mosqueda, M.E., Yáñez-Márquez, C., López-Yáñez, I.: Alpha-Beta Bidirectional Associative Memories Based Translator. IJCSNS International Journal of Computer Science and Network Security, Vol. 6, No. 5A, México (2006) pp. 190-194. ISSN: 1738-7906

[63] Aldape-Pérez, M., Yáñez-Márquez, C., López -Leyva, L.O.: Optimized Implementation of a Pattern Classifier using Feature Set Reduction. Research in Computing Science, Vol. 24, Special issue: Control, Virtual Instrumentation and Digital Systems, IPN México (2006) pp. 11-20. ISSN 1870-4069

[64] Sánchez Garfias, F.A., Díaz-de-León Santiago, J.L., Yáñez Márquez, C.: New Results on the Lernmatrix Properties. Research on Computing Science Series, Vol. 10, IPN, México (2004) pp. 91-102. ISSN 1665-9899

[65] Román-Godínez, I., López-Yáñez, I., Yáñez-Márquez, C.: A New Classifier Based on Associative Memories. IEEE Computer Society, Proc. 15th International Conference on Computing, CIC México (2006) pp. 55-59. ISBN: 0-7695-2708-6

[66] Aldape Pérez, M., Yáñez Márquez, C., López Leyva L.O.: Reducción del espacio de rasgos para el diseño de Clasificadores de Patrones Optimizados. Artículo Id

EO-014 en Proc. del 9o. Congreso Nacional de Ingeniería Electromecánica y de Sistemas, ESIME IPN, México (2006) pp. 64-69. ISBN: 970-36-0355-6

[67] Sánchez-Garfias, F.A., Díaz-de-León Santiago, J.L, Yáñez-Márquez, C.: A new theoretical framework for the Steinbuch's Lernmatrix. en: Proc. Optics & Photonics, Conference 5916 Mathematical Methods in Pattern and Image Analysis, SPIE, San Diego, CA (2005). pp. (59160)N1-N9. ISBN: 0-8194-5921-6, ISSN: 0277-786X

[68] Hassoun, M.H.(ed.): Associative Neural Memories. Oxford University Press, New York (1993)

[69] Kohonen, T.: Self-Organization and Associative Memory. Springer-Verlag, Berlin (1989)

[70] Sossa, H., Barrón, R., Vázquez, R.: New Associative Memories to Recall Real-Valued Patterns. CIARP 2004, LNCS 3287 (2004) 195-202

[71] Díaz-de-León, J.L., Yáñez-Márquez, C.: Memorias asociativas con respuesta perfecta y capacidad infinita. Memoria del TAINA'99, México, D.F. (1999) 23-38

[72] Brier, G.W. : Verification of forecasts expressed in terms of probabilities. Monthly Weather Review (1950) 78,1–3

[73] Kononenko, I.:Estimating attributes: analysis and extensions of RELIEF. Proc European Conference on Machine Learning (ECML), 171–182

[74] Hart, P.: The condensed nearest neighbor rule (Corresp.). IEEE Transactions on Information Theory, vol. 14, issue 3. ISSN 0018-9448 (1968) 515- 516

[75] Djouadi, A., Bouktache, E.: A fast algorithm for the nearest-neighbor classifier. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, issue 3. ISSN 0162-8828 (1997) 277-282

[76] Zhang, B., Srihari, S.N.: A Fast Algorithm for Finding k-Nearest Neighbors with Non-metric Dissimilarity. Proc. of the Eighth International Workshop on Frontiers in Handwriting Recognition, 2002. IEEE, ISBN 0-7695-1692-0 (2002) 13-18

[77] Steinbuch, K.: Die Lernmatrix. Kybernetik, vol. 1, num. 1 (1961) 36-45

[78] Sossa, H, Barrón, R, Cuevas, F: Extended associative memories for recalling gray level patterns. Lecture Notes in Computer Science, Springer Heidelberg, 2004 .LNCS 3287, pp. 187-194

[79] Sossa, H, Oropeza, J. L.,Cortés, H.: Associative memory based real-valued pattern recall. IEEE Computer Society Proc. of the fifth Mexican International Conference in: Computer Science, ENC 2004, USA, 2004, pp. 206-212

[80] Humberto Sossa, Ricardo Barrón, Oropeza, José L.: Real-valued pattern recall by associative memory. Lecture Notes in Artificial Intelligence, Subseries of Computer Science, Springer Heidelberg, 2004, LNCS 3315, pp. 646-655

[81] Sossa,Humberto, Barrón, Ricardo, Cuevas, Francisco, Carlos , Aguilar, Héctor C.: Binary associative memories applied to gray level pattern recalling. Lecture Notes in Artificial Intelligence, Subseries of Computer Science, Springer Heidelberg, 2004, LNCS 3315, pp. 656-666

[82] Sossa, Humberto, Barrón, Ricardo, A, Roberto: Transforming fundamental set of pattern to a canonical form to improve pattern recall. Lecture Notes in Artificial Intelligence, Subseries of Computer Science, Springer Heidelberg, 2004, LNCS 3315, pp. 687-696

[83] Sossa,Humberto, Barrón, Ricardo , Cortés, Héctor, Sánchez, Flavio: Nuevas memorias asociativas para patrones en niveles de gris. Proceedings of the 4th. International Conference on Control, Virtual Instrumentation and Digital Systems, CICINDI, México, 2002, pp. 27-1 a 27-5

[84] Salgado-Ramírez, J.C.: Estudio estadístico comparativo entre Memorias Asociativas Clásicas, Memorias Morfológicas y Memorias Alfa-Beta para el caso binario. Tesis de Maestría en ClC IPN, México (2005)

[85] Yáñez-Márquez, C., Cruz-Meza, M.E., Sánchez-Garfias, F.A., López-Yáñez, I.: Using Alpha-beta Associative Memories to Learn and Recall RGB Images. En Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (Eds.): Advances in Neural Networks – ISNN 2007. 4th International Symposium on Neural Networks, ISNN 2007, Nanjing, China, June 3-7, 2007. Lecture Notes in Computer Science 4493. Springer-Verlag Berlin Heidelberg (2007) 828-833

[86] Román-Godínez, I., López-Yáñez, I., Yáñez-Márquez, C.: Perfect Recall on the Lernmatrix. En Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (Eds.): Advances in Neural Networks – ISNN 2007. 4th International Symposium on Neural Networks, ISNN 2007, Nanjing, China, June 3-7, 2007. Lecture Notes in Computer Science 4492. Springer-Verlag Berlin Heidelberg (2007) 834-841

[87] González, R. C. , Woods, R. E.: Digital Image Processing.Prentice Hall,USA (2001)

[88] Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifier. In: D. Haussler (ed.), 5th Annual ACM warshop on COLT, Pittsburgh, PA. ACM Press (1992) 144–152

[89] Cortes, C.,Vapnik V.: Support Vector Network. Machine Learning, 20 (1995) 273 – 297

[90] Yap, C.W., Xue, Y., Chen, Y.Z.: Application of Support Vector Machines to In Silico Prediction of Cytochrome P450 Enzyme Substrates and Inhibitors. Current Topics in Medicinal Chemistry, Vol. 6, Number 15. Bentham Science Publishers (2006) 1593-1607

[91] Pochet, N.L.M.M., Suykens, J.A.K.: Support vector machines versus logistic regression: improving prospective performance in clinical decision-making. Ultrasound in Obstetrics and Gynecology, vol. 27, number 6. John Wiley & Sons, Ltd. (2006) 607-608

[92] Wu, K., Yap, K.-H.: Fuzzy SVM for content-based image retrieval: a pseudo-label support vector machine framework. IEEE Computational Intelligence Magazine, Vol. 1, Issue 2 (2006) 10-16

[93] Bahamonde, A., Díez, J., Quevedo, J.R., Luaces, O., del Coz, J.J.: How to learn consumer preferences from the analysis of sensory data by means of support vector machines (SVM). Trends in Food Science and Technology, Vol. 18, Issue 1. Elsevier BV. ISSN 0924-2244 (2007) 20-28

[94] Zhou, D., Wu, L., Liu, G.: Bayesian Classifier Based on Discretized Continuous Feature Space. Fourth International Conference on Signal Processing Proceedings, 1998. ICSP '98 (1998) 1225-1228

[95] Su, Z., Zhang, H., Ma, S.: Using Bayesian Classifier in Relevant Feedback of Image Retrieval. 12th IEEE International Conference on Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. (2000) 258-261

[96] Han, X., Wakabayashi, T., Kimura F.: The Optimum Classifier and the Performance Evaluation by Bayesian Approach. In: F.J. Ferri et al. (Eds.): SSPR&SPR 2000. LNCS Vol. 1876 Springer-Verlag Berlin Heidelberg (2000) 591-600

[97] Garg, A., Roth, D.: Understanding Probabilistic Classifiers. In: L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI Vol. 2167 Springer-Verlag Berlin Heidelberg (2001) 179–191

[98] Kelemen, A., Zhou, H., Lawhead, P., Liang, Y.: Naive Bayesian Classifier for Microarray Data. Proceedings of the International Joint Conference on Neural Networks, 2003. (2003) 1769- 1773

[99] Kotsiantis, S.B., Pintelas, P.E.: Increasing the Classification Accuracy of Simple Bayesian Classifier. In: C. Bussler and D. Fensel (Eds.): AIMSA 2004, LNAI Vol. 3192 Springer-Verlag Berlin Heidelberg (2004) 198–207

[100] Altinçay, H.: On Naive Bayesian Fusion of Dependent Classifiers. Pattern Recogn. Lett. 26(15). Elsevier Science Inc. (2005) 2463-2473

[101] Pronk, V., Gutta, S.V.R., Verhaegh, W.F.J.: Incorporating Confidence in a Naive Bayesian Classifier. In: L. Ardissono, P. Brna, and A. Mitrovic (Eds.): UM 2005, LNAI Vol. 3538 Springer-Verlag Berlin Heidelberg (2005) 317–326

[102] Yager, R.R.: An Extension of the Naive Bayesian Classifier. Information Sciences 176(5) 6 March 2006. Elsevier Science Inc. (2006) 577-588

[103] StatSoft Inc. Naive Bayes Classifier. Available at http://www.statsoft.com/textbook/stnaiveb.html (2006)

[104] Onat, B.M., McNeill, J.A., Cilingiroglu, U.: Implementation of a Charge-Based Neural Euclidean Classifier for a 3-Bit Flash Analog-to-Digital Converter. IEEE Transactions on Instrumentation and Measurement (1997) 672-677

[105] Cilingiroglu, U., Aksin, D.Y.: A 4-transistor Euclidean Distance Cell for Analog Classifiers. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, 1998. ISCAS '98. (1998) 84-87

[106] Vapnik , Lerner: Patern recognition using generalized pertrait method. Automation and Remote Control, (1963) 24

[107] Vapnik,V.,Chervonenkis, A.: A note on one class of Perceptrons. Automation and Remote Control, (1964) 25

[108] Drucker,H., Burges,C. J. C.,Kaufman, L., Smola,A.,Vapnik, V.: Support vector regression machines. In: M. Mozer, M. Jordan, T. Petsche(eds): Advances in Neural Information. Processing Systems 9, MA, MIT Press, Cambridge (1997 ) 155-161

[109] Burges,C. J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. Knowledge Discovery and Data Mining, 2(2), 1998

[110] Lin, H., Venetsanopoulos, A.N.: A Weighted Minimum Distance Classifier for Pattern Recognition. Canadian Conference on Electrical and Computer Engineering (1993) 904-907

[111] Senda, S., Minoh, M., Katsuo, I.: A Fast Algorithm for the Minimum Distance Classifier and Its Application to Kanji Character Recognition. Third International Conference on Document Analysis and Recognition (ICDAR'95) Vol. 1 (1995) 283

[112] Rosen, K.H.: Discrete Mathematics and Its Applications. McGraw-Hill. Third Edition (1995)

[113] Cruz-Meza, M.E. Aprendizaje y Recuperación de Imágenes en Color Mediante Memorias Asociativas Alfa-Beta. Tesis de Maestría en: Ciencias de la Computación. CIC IPN, México (2007)

[114] Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. (1998) Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html

[115] Patwari, N., Hero III, A.O., Perkins, M., Correal, N., O'Dea, R.J.: Relative Location Estimation in Wireless Sensor Networks. IEEE Transactions on Signal Processing. vol. 51, no. 8 (2003) 2137-2148

[116] Patwari, N.: Wireless Sensor Network Localization Measurement Repository. Available at: http://www.eecs.umich.edu/~hero/localize/

[117] Ou, G., Murphey, Y.L.: Multi-Class Pattern Classification Using Neural Networks. Pattern Recognition Vol. 40(1). Pattern Recognition Society. Elsevier Ltd. (2007) 4–18

[118] Ayaquica-Martínez, I. O.: Algoritmo C-means usando funciones de disimilaridad. Tesis de Maestría en: Ciencias de la Computación. CIC IPN, México (2002)

[119] Komarek, P., Liu, T., Tengli, A., Moore, A., DiFazio, C.: Fast Classifiers. Carnegie Mellon University, Auton Lab (2006). Available at: http://www.autonlab.org/autonweb/16478.html

[120] Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. IEEE Transactions on Evolutionary Computation (1997)

[121] Wolpert, D.H., et al.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation (1997)

[122] Sewell, M.: No Free Lunch Theorems. (2006) Available at: http://www.no-free-lunch.org/

[123] Wang, J., Neskovic, P., Cooper, L.N.: Improving Nearest Neighbor Rule With a Simple Adaptive Distance Measure. Pattern Recognition Letters Vol. 28. Elsevier B.V. (2007) 207–213