INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

TEORÍA Y APLICACIONES DEL CLASIFICADOR ASOCIATIVO GAMMA

T E S I S

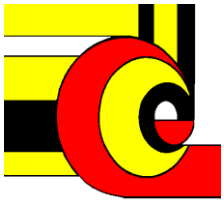
QUE PARA OBTENER EL GRADO DE
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ITZAMÁ LÓPEZ YÁÑEZ

DIRECTORES DE TESIS:

DR. CORNELIO YÁÑEZ MÁRQUEZ
DR. VÍCTOR MANUEL SILVA GARCÍA



MÉXICO, D.F. JUNIO DE 2011

Resumen

En este documento se presentan las bases teóricas que sustentan el funcionamiento del clasificador Gamma, mismo que pertenece al enfoque asociativo de clasificación de patrones y fue introducido por el autor en su tesis de maestría. Asimismo, se presentan diversas aplicaciones de este clasificador a diferentes campos del conocimiento humano.

Se incluye también la caracterización del funcionamiento del clasificador, permitiendo identificar condiciones suficientes para clasificación correcta del conjunto fundamental completo.

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

Índice General

Resumen

En este documento se presentan las bases teóricas que sustentan el funcionamiento del clasificador Gamma, mismo que pertenece al enfoque asociativo de clasificación de patrones y fue introducido por el autor en su tesis de maestría. Asimismo, se presentan diversas aplicaciones de este clasificador a diferentes campos del conocimiento humano.

Se incluye también la caracterización del funcionamiento del clasificador, permitiendo identificar condiciones suficientes para clasificación correcta del conjunto fundamental completo.

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

Abstract

In the current document of thesis, the theoretical concepts which make up the basis of the Gamma classifier are presented. The Gamma classifier is a member of the associative approach to pattern recognition and was introduced by the author in his Master thesis. Several applications of this classifier to different fields of human knowledge are included too.

The characterization of the classifier operation is also presented. This allows the identification of sufficient conditions for correct classification of the whole fundamental set.

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

Introducción

En este documento se presentan las bases teóricas que sustentan el funcionamiento del clasificador Gamma, mismo que pertenece al enfoque asociativo de clasificación de patrones y fue introducido por el autor en su tesis de maestría. Asimismo, se presentan diversas aplicaciones de este clasificador a variados campos del conocimiento humano.

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 [PR63]. 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 [PR48].

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 [PR-patclas18]. 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 [cornelio15].

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 [PR73], [PR-patclas44].

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 [PR48], [PR-patclas18]. Por ello, gran parte de la literatura relacionada con el área de reconocimiento de patrones menciona explícitamente la clasificación de patrones [PR-patclas64], [nuevas09].

Actualmente, entre los principales enfoques de reconocimiento automático de patrones, se encuentran los siguientes:

- *Enfoque estadístico-probabilístico* [PR-patclas18], [PR73], [PR21]-[MA-ab15].
- *Clasificadores basados en métricas* [PR-patclas100]-[nuevas10].
- *Enfoque sintáctico-estructural* [PR21], [PR-patclas69], [MA78], [nuevas11].
- *Enfoque neuronal* [PR21]-[nuevas13], [NN49]-[cornelio22].
- *Enfoque asociativo* [cornelio15], [PR-patclas64], [cornelio22]-[nuevas25].

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 [PR-patclas100], [cornelio19], [nuevas20], [nuevas21]. En el clásico proyecto Statlog, desarrollado en Escocia por Michie y sus colaboradores a inicios de la década de los noventa del siglo pasado, se muestra experimentalmente la superioridad del k -NN respecto de un buen número de clasificadores automáticos de patrones [nuevas13]. 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 [PR-patclas100], [PR-patclas103]-[PR-patclas105].
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 [nuevas12], desarrollada en 2002 en el CIC-IPN. Este enfoque se basa en las memorias asociativas, cuyos modelos matemáticos pioneros [MA71] son contemporáneos a los primeros modelos de redes neuronales [NN58]. 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: el operador alfa y el operador beta, las cuales son equiparables en sencillez a las operaciones básicas de la lógica booleana [MA79]. 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 [cornelio15], [PR-patclas64], [nuevas09], [cornelio22]-[cornelio21], [MA80]-[nuevas25]. Además, estos modelos son más eficientes que el k -NN para conjuntos grandes de patrones de aprendizaje [nuevas17]. Cabe mencionar que las memorias asociativas alfa-beta, cuyos operadores sirven de fundamento al presente documento de tesis, trabajan solamente con patrones binarios [MA79], [nuevas14], [nuevas16], [nuevas18]**.**

Uno de los algoritmos de reciente creación, mismo que toma algunos elementos de las memorias asociativas alfa-beta y que cae dentro del enfoque asociativo, es el clasificador Gamma [itzama01]. Este algoritmo ha mostrado un desempeño competitivo en las aplicaciones en las que ha sido utilizado hasta el momento, con respecto a los algoritmos presentes en el estado del arte de la literatura científica actual. Asimismo, ha mostrado un desempeño aceptable en tareas para las que no fue diseñado, como lo es la regresión de funciones.

De esta manera, la presente propuesta de tesis queda inmersa dentro del área de investigación del enfoque asociativo de reconocimiento automático de patrones, particularmente en el ámbito del clasificador Gamma.

Justificación

Con la introducción del clasificador asociativo Gamma [itzama01], se engrosan las filas del enfoque asociativo de reconocimeinto de patrones. Más aún, este clasificador ofrece resultados competitivos con respecto al 1-NN y otros clasificadores en varias bases de datos. Sin embargo, el funcionamiento del clasificador Gamma no ha sido caracterizado: no se sabe a ciencia cierta por qué funciona, cómo funciona, ni bajo qué condiciones ofrecerá mejores o peores resultados de clasificación que otros clasificadores, ni bajo qué condiciones su operación se vuelve ineficiente. Por otro lado, falta probarlo en otros ámbitos, así como desarrollar aplicaciones del mismo. El fundamento teórico y la caracterización de la operación del clasificador Gamma en ambas fases, aprendizaje y clasificación, justifican el trabajo de tesis propuesto.

Objetivo

Fundamentar teóricamente el clasificador asociativo Gamma y ---partiendo de ese fundamento--- caracterizar su funcionamiento en ambas fases, aprendizaje y clasificación de patrones. Esta fundamentación teórica deberá explicar bajo qué condiciones clasificará correctamente, así como las condiciones necesarias y/o suficientes bajo las cuales operará eficientemente. Adicionalmente, aplicarlo a diversos ámbitos del conocimiento humano.

Contribuciones

- Fundamento teórico del clasificador asociativo Gamma.
- Caracterización del clasificador Gamma, sobre todo en los siguientes aspectos:
  - Condiciones suficientes para clasificación correcta.
  - Condiciones de operación eficiente.
- Aplicaciones.
- Comparación con otros clasificadores de patrones [itzama03]-[itzama07].

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 serán expuestos en el Capítulo 3.

El Capítulo 4 es la parte más relevante de este documento. En el mismo se presenta la caracterización y fundamentación teórica del clasificador Gamma, así como las mejoras y modificaciones propuestas. 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, contribuciones y recomendaciones para trabajo futuro.

Finalmente, se incluyen las referencias bibliográficas.

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 [PR-patclas18], [PR73], [PR21]-[MA-ab15].
- *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 [PR-patclas100]-[nuevas10].
- *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 [PR21], [PR-patclas69], [MA78], [nuevas11].
- *Enfoque neuronal*. Se basa en modelos matemáticos de las neuronas del cerebrohumano y, a diferencia del enfoque estadístico-probabilístico, los sistemas automáticos de reconocimiento de patrones basados en el enfoque neuronal, además de clasificar patrones, también son capaces de recuperarlos [PR21]-[nuevas13], [NN49]-[cornelio22].
- *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 [cornelio15], [PR-patclas64], [cornelio22]-[itzama02], [sossa1]-[nuevas24], [nuevas25].

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 de máxima verosimilitud [PR-patclas18], y el clasificador euclidiano se identifica con el enfoque basado en métricas [octavio24]. A pesar de ello, 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 euclidiano 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) [SVM92], [SVM93]. Cabe mencionar que esta técnica ha sido aplicada exitosamente en diversas áreas de la actividad humana [nuevas04]-[nuevas07].

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 pertenece al enfoque de clasificadores basados en métricas. En el clásico proyecto Statlog se muestra la gran eficacia del k -NN, quedando como uno de los clasificadores a vencer para cualquier nueva propuesta [nuevas13]. Sin embargo, la gran desventaja del k -NN es su poca eficiencia al trabajar con una conjunto amplio de patrones de aprendizaje, ya que calcular las distancias entre todos los patrones y ordenarlas es un proceso computacionalmente caro [PR-patclas100], [PR-patclas103]-[PR-patclas105]. El enfoque asociativo, por su parte, se inició con el CHAT (Clasificador Híbrido Asociativo con Traslación) [nuevas12], desarrollado en 2002 por Raúl Santiago Montero en el CIC-IPN. Este enfoque se basa en utilizar las memorias asociativas ---cuyo estado del arte incluye a las memorias asociativas alfa-beta--- para clasificar patrones. 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 [cornelio15], [PR-patclas64], [nuevas09], [cornelio22]-[cornelio21], [MA80]-[itzama02], [sossa1]-[nuevas24], [nuevas25]. Adicionalmente, es preciso hacer notar que los modelos asociativos son más eficientes que el k -NN para conjuntos grandes de patrones de aprendizaje [nuevas17], 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) [MA79], [nuevas14], [nuevas16], [nuevas18].

Clasificador de Máxima Verosimilitud

El contenido de la presente sección está basado en las referencias [PR-patclas18], [MA-ab15], [Bayes2]-[Bayes10], 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 particular, 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 de máxima verosimilitud, mismo que es un caso particular de la teoría de la decisión bayesiana [PR-patclas18]. 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... ([PR-patclas18], pp. 17).

Theorem ***Teorema de Bayes.*** *Sean los eventos* $A_1, A_2, \ldots, A_n$ *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:

$$posterior = \frac{a\ priori \times verosimilitud}{evidencia}$$

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 forma vista en TeoBayes2. 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 de máxima verosimilitud toma ventaja de las probabilidades condicionales al sustituir, en el proceso de clasificación, el teorema de Bayes en la regla ReglaBayes1:

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

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

$\forall k = 1, 2, \ldots, n$ 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 ReglaBayes1, y tomando en cuenta que las probabilidades siempre son positivas, queda la expresión ReglaBayes2.

$$P(C_i \mid X) > P(C_j \mid X)$$
$$\frac{P(C_i)\, p(X \mid C_i)}{P(X)} > \frac{P(C_j)\, p(X \mid C_j)}{P(X)}$$
$$P(C_i)\, p(X \mid C_i) > P(C_j)\, p(X \mid C_j)$$

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 ReglaBayes2:

$$\ln\big(P(C_i)\, p(X \mid C_i)\big) > \ln\big(P(C_j)\, p(X \mid C_j)\big)$$
$$\ln(P(C_i)) + \ln(p(X \mid C_i)) > \ln(P(C_j)) + \ln(p(X \mid C_j))$$
$$d_i > d_j, \text{ con } d_k = \ln(P(C_k)) + \ln(p(X \mid C_k))$$

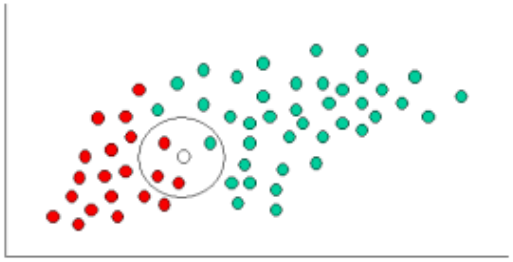
donde d_k define una función discriminante para el clasificador.
Así pues, que un patrón desconocido sea clasificado en una clase C_i en particular, implica tener todo el conocimiento *a priori* de cada clase C_i y su distribución de probabilidad correspondiente, lo que raramente sucede en la práctica [PR-patclas18].
Tomando en cuenta lo anterior, el algoritmo para diseñar un clasificador de máxima verosimilitud queda como sigue:

Algorithm *Algoritmo del clasificador de máxima verosimilitud.*

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.
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 :
6) $X \in C_i$, si $d_i > d_j \ \forall i \neq j$, con $d_k = \ln(P(C_k)) + \ln(p(X \mid C_k))$

■

7)
En la figura FigBay se puede ver un ejemplo de clasificación usando el clasificador de máxima verosimilitud, tomada de [Bayes1].



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 confiables 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 de máxima verosimilitud se ve limitado, forzando a los investigadores en este campo a establecer condiciones artificiales a las probabilidades condicionales de modo que sea funcional su uso.

Clasificador Euclidiano

El funcionamiento básico de los clasificadores basados en métricas, entre los que se encuentra el Clasificador Euclidiano [octavio24], [MA-ab16], [cornelio20], [euclidean1], [euclidean2] se muestra en el siguiente algoritmo:

Algorithm *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:

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

■

5)

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 algoritmo anterior se especifica que la métrica a utilizar es la distancia euclidiana, entonces se tiene el clasificador euclidiano. 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 euclidiana* 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 manera:

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

o en forma vectorial:

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

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

Algorithm *Primer algoritmo del clasificador euclidiano.*

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 euclidiana d_2 , este patrón será clasificado en la clase C_i , si se cumple lo siguiente:

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

■

5)

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

Algorithm *Algoritmo del clasificador euclidiano que usa funciones discriminantes.*

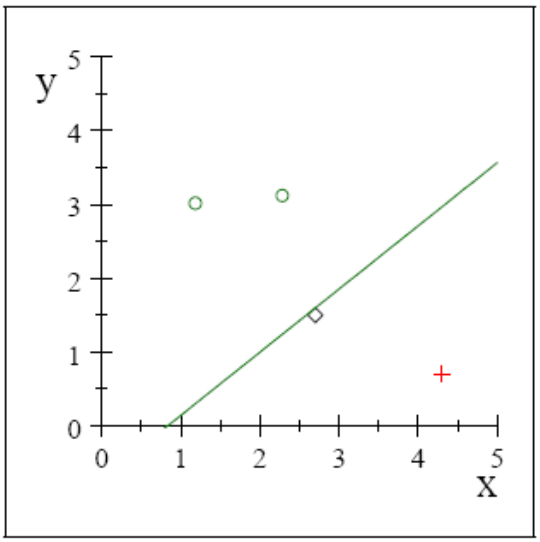
1) Escoger una muestra de patrones clasificada de antemano en p clases $\{C_1, C_2, .., C_p\}$.
2) Con base en la muestra y para cada clase C_i , calcular el patrón representante $\mu_i = \frac{1}{k} \sum_{x_j \in C_i} x_j$, donde k es el número de elementos en la muestra que pertenecen a C_i .
3) Generar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

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

■

6)

Así, el clasificador euclidiano 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 FigEuc2, tomada de [MA-ab16].



Remark *El clasificador euclidiano clasifica correctamente si las clases* C_i *y* C_j *son linealmente separables.*

El clasificador euclidiano, aunque simple de implementar, posee limitaciones fuertes, como el hecho de que para que presente una 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.

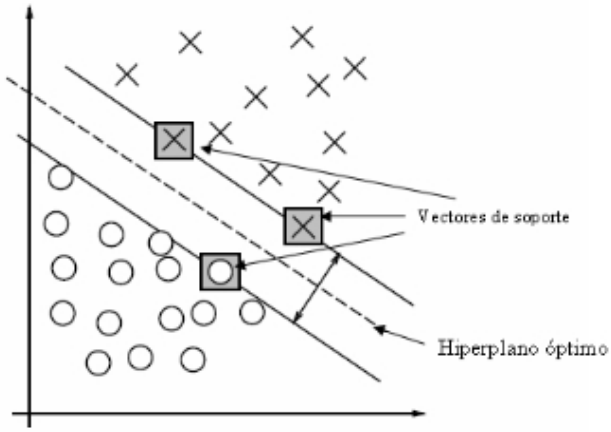
Support vector machines (SVMs)

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 [SVM90], [SVM91]. 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 SVMs durante la década de los noventa, en los Bell Laboratories de AT&T por Vapnik y sus colaboradores [SVM92], [SVM93].

El problema básico que ataca este modelo es el 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 SVMs 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 [PR-patclas18].

Ahora bien, puede darse el caso de que existan patrones, cercanos a la frontera de las

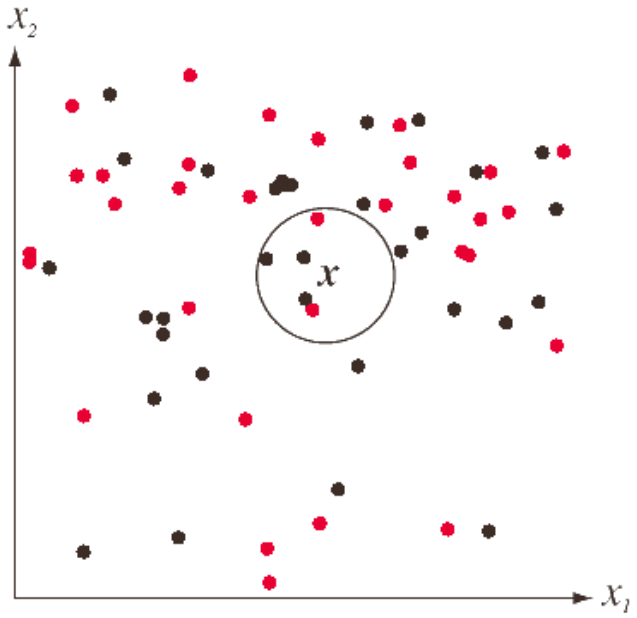
clases, que se comportan más como excepciones. Si dichos patrones se tomaran como vectores de soporte, degradarían la generalización de la clasificación. Para solucionar este problema, se incluye cierto margen de error, con lo cual se admite que ciertos patrones cercanos a la frontera no sean tomados 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 [SVM93], [SVM101], [SVM102]. Un ejemplo de SVM se puede apreciar en la figura FigSVM1.



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 a un espacio en una dimensión mayor, donde el problema sea linealmente separable [SVM102].

k -*Nearest Neighbor* (k -NN)

Uno de los algoritmos de clasificación de patrones más notables es sin duda el del k -*Nearest Neighbor* [PR-patclas100]-[PR-patclas74], [cornelio12], [cornelio19], [nuevas20], [nuevas21], [knn1], [knn2]; 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 [nuevas13]. En la Figura FigKNN1 se puede ver un ejemplo de 5-NN, tomada de [PR-patclas18]. A continuación se describe el algoritmo k -NN, tanto para el caso de $k = 1$ como para $k > 1$.



Algorithm *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.

Asignar a x la clase del patrón con la mínima distancia.

■

4)

Algorithm *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) Ordenar los datos en orden ascendente.
4) Obtener los k menores valores de distancia.

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

■

5)

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 [PR-patclas100], [PR-patclas103]-[PR-patclas105].

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 [PR-patclas64], [nuevas12]. 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 pertenecientes 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:
Algorithm *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.

Se procede a clasificar los patrones desconocidos.
■
9)

El autor del CHAT describe en su trabajo de tesis [nuevas12] 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 conocidas en el área de reconocimeinto de patrones, con objeto de comparar el desempeño de los clasificadores mostrados en diferentes publicaciones actuales, respecto del clasificador Gamma, tema central de esta tesis.

Materiales y Métodos

En este Capítulo se presentan los conceptos, materiales y métodos que se requieren para describir y fundamentar teóricamente el clasificador Gamma.

En la primera sección se describen los operadores fundamentales de los modelos asociativos alfa-beta: el operador alfa y el operador beta, así como algunas de sus propiedades; estos operadores son la base del clasificador Gamma. La sección 2 incluye el operador u_β, que resulta sumamente útil en la formulación del clasificador Gamma; se presentan ejemplos numéricos. Dos importantes conceptos matemáticos, módulo y congruencia, se incluyen en la sección 3, en virtud de que son también utilizados para formular el algoritmo del clasificador Gamma.

Dado que los vectores con que trabaja el clasificador Gamma serán codificados con el código Johnson-Möbius modificado [nuevas16], 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 [cornelio03]. Finalmente, en la sección 5 se presenta el operador Gamma de similitud generalizado, que resulta de gran importancia para la formulación del algoritmo del clasificador Gamma, mientras que la sección 6 está dedicada al mencionado algoritmo del clasificador Gamma.

Operador alfa y operador beta

En esta sección se presentan los operadores fundamentales de los modelos asociativos alfa-beta, junto con sus propiedades, algunas de las cuales serán usadas intensamente en la propuesta de esta tesis. El material es tomado directamente de [MA79]-[cornelio05], [cornelio08]-[cornelio10], [MA-ab14].

Las memorias asociativas alfa-beta son de dos tipos (*max* y *min*) y pueden operar en dos modos diferentes (autoasociativo y heretoasociativo). El operador alfa 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 [MA79]; a continuación se incluyen las tablas que representan a dichos operadores, dados los conjuntos $A = \{0,1\}$ y $B = \{0,1,2\}$:

Tabla 3.1. Definición del operador alfa (izq.) y el operador beta (der.)

$\alpha : A \times A \to B$		
x	y	$\alpha(x,y)$
0	0	1
0	1	0
1	0	2
1	1	1

$\beta : B \times A \to A$		
x	y	$\beta(x,y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

El operador binario alfa exhibe algunas propiedades algebraicas, expuestas en la tabla 3.2.

Tabla 3.2. Propiedades del operador binario alfa

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

Asimismo, en la tabla 3.3 se muestran algunas propiedades del operador binario beta.

Tabla 3.3. Propiedades del operador binario beta

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

Por otro lado, en la tabla 3.4 se presentan las propiedades de la aplicación combinada de ambos operadores: alfa y beta.

Tabla 3.4. Propiedades de la aplicación combinada de los operadores alfa y beta

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

Operador u_β

El operador aquí descrito y ejemplificado fue creado por el autor y resulta de gran utilidad en el desarrollo, aplicación y prueba del algoritmo del clasificador Gamma.

Definition *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:*

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

■

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

$$u_\beta(\mathbf{x}) = \sum_{i=1}^{5} \beta(\mathbf{x}_i, \mathbf{x}_i)$$

Dada la definición DefUBeta, , 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, $u_\beta(\mathbf{x}) = 2$.

■

Example *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
$u_\beta(\mathbf{y}) = \beta(1,1) + \beta(0,0) + \beta(0,0) + \beta(1,1) + \beta(1,1) + \beta(1,1) + \beta(1,1) + \beta(1,1) = 1 + 0 + 0 + 1 + 1 + 1 +$
. Entonces, $u_\beta(\mathbf{y}) = 6$.

■

Example *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
$u_\beta(\mathbf{y}) = \beta(0,0) + \beta(0,0) + \beta(0,0) + \beta(0,0) = 0 + 0 + 0 + 0 = 0$.

■

Módulo y Congruencia

El contenido de esta sección fue tomado de [nuevas23]. 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 .

Definition *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 $0 \leq r < m$

.

Example *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.

Definition *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 $a \bmod m = b \bmod m$.

Example *Determinar: a) si* 17 *es congruente con* 5 *módulo* 6 *y b) si* 24 *y* 14 *son congruentes módulo* 6 .

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

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

2)

Código binario Johnson-Möbius modificado

El contenido de esta sección está basado en [nuevas16] y [cornelio03]. Dado que los vectores con que trabaja el clasificador Gamma están 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 [nuevas16].

Algorithm *Algoritmo del Código Johnson-Möbius Modificado:*

1) Sea un conjunto de números reales

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

3) donde n es un número entero positivo fijo.

4) 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

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

6) 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.

7) 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.

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

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

10) donde e_m es el número mayor.

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.

■

11)

Example *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

■

Example *Sea el conjunto* $r = \{ 0.2, 1.25, -0.3, 2.147 \}; r \subset \mathbb{R}$.

- 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

■

Operador Gamma de similitud generalizado

El operador Gamma de similitud generalizado deriva su nombre de sus características: en primer lugar, está basado fuertemente en los operadores fundamentales de las memorias asociativas alfa-beta, aunque de una manera novedosa; por ello, se opta por nombrarle Gamma, siendo el nombre de la letra griega que está a continuación de las primeras dos (alfa y beta). Por otro lado, este operador indica si dos vectores son parecidos o no, dado un grado de disimilitud θ ; por ello, este operador es de similitud. En este sentido, el argumento θ indica la tolerancia para que dos vectores, al compararlos, sean considerados similares ---no obstante que son diferentes---. Por otra parte, este operador generaliza al operador Gamma de similitud, permitiéndole operar con vectores binarios de dimensiones diferentes. A continuación se define y ejemplifica el operador Gamma de similitud generalizado, tal como aparece en [itzama01].

Definition *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* θ *un número entero no negativo. Se define el operador Gamma 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* θ *, y la salida es un número binario que se calcula así:*

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

si $m > n$ *, truncar los* $m - n$ *bits izquierdos de* $\mathbf{y}$ *antes de realizar el cálculo.*

■

Example *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* $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$.

En este caso se puede observar que $n = m = 8$. 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 $m = 8$ da como resultado $8 - 2 = 6$; pero $6 > 3$ (o lo que es lo mismo $6 \not\leq 3$), entonces $\gamma_g(\mathbf{x}, \mathbf{y}, \theta) = 0$, ya que este resultado cae en la opción de otro caso de la definición DefGamaSimGen.

■

Example *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* $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$.

$$\alpha(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix},$$

Se puede observar que para este caso $n = m = 10$. Luego, $\begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} \bmod 2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $u_\beta \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 5$, $10 - 5 = 5$, y como $5 = 5$ entonces $5 \leq 5$, por lo que $\gamma_g(\mathbf{x},\mathbf{y},\theta) = 1$.

■

Example *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* $\gamma_g(\mathbf{x},\mathbf{y},\theta)$.

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 $\gamma_g(\mathbf{x},\mathbf{y},\theta) = 1$.

■

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

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 $\gamma_g(\mathbf{x},\mathbf{y},\theta) = 1$.

■

Remark *Nótese que* $\gamma_g(\mathbf{x},\mathbf{y},0) = 1 \leftrightarrow \mathbf{x} = \mathbf{y}$.

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

Para este caso se tiene que $n = 3$ y $m = 4$. Entonces, se trunca $\mathbf{y}$ de tal forma que

ahora $\mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ Así, $\alpha(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \bmod 2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$

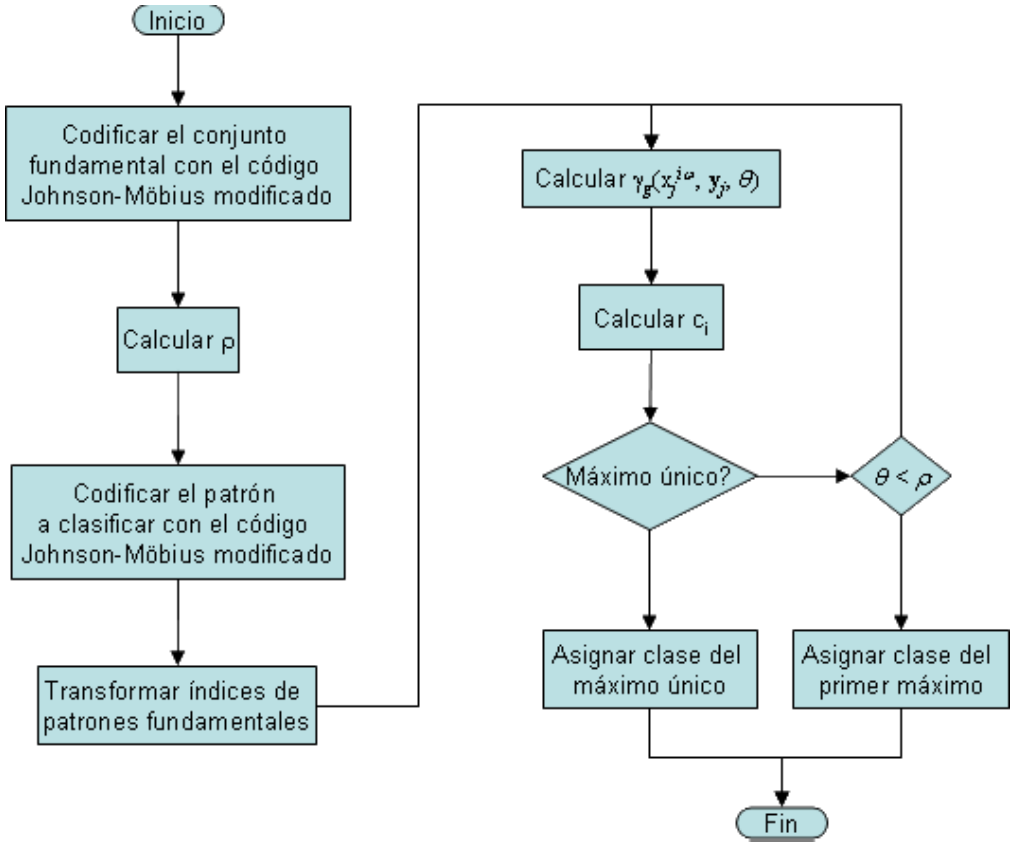
$u_\beta \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 3$, $4 - 3 = 1$, pero como $1 \not\leq 0$ puesto que $1 > 0$, el resultado es $\gamma_g(\mathbf{x},\mathbf{y},\theta) = 0$. Nótese que a pesar de que las secciones de $\mathbf{x}$ y $\mathbf{y}$ que se compararon son idénticas, los dos vectores no lo son. Como $\theta = 0$, el resultado es congruente, pues $\theta = 0$ exige una situación de igualdad para arrojar un resultado de 1.

■

Algoritmo original del clasificador Gamma

El algoritmo del clasificador Gamma hace uso de los operadores fundamentales de las memorias asociativas alfa-beta y sus propiedades, así como del operador Gamma de similitud generalizado (que a su vez requiere del operador u_β 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 clasificador se denomina Gamma. A continuación se presenta el algoritmo del clasificador Gamma, tal y como se propuso originalmente en [itzama01].

Algorithm *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 ω). 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 θ 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_ξ .

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

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

9)

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

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

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

13)

En caso contrario: si λ es el índice más pequeño de clase que corresponde a uno de los máximos, asignar al patrón a clasificar la clase C_λ .

■

14)

Como los patrones con los que se opera $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ en este 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 $mgamma_\xi$ como dimensión de $\mathbf{y}_\xi$.

Example *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.

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

3)

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

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

6)

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

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

9)

10) 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 $\mathbf{x}^1 = \mathbf{x}^{11}$, $\mathbf{x}^2 = \mathbf{x}^{12}$; $\mathbf{x}^3 = \mathbf{x}^{21}$, $\mathbf{x}^4 = \mathbf{x}^{22}$.

11) $\theta = 0$.

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

1) $\gamma_g(\mathbf{x}_1^{11}, \mathbf{y}_1^1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{11}, \mathbf{y}_2^1, 0) = 0$; $\gamma_g(\mathbf{x}_1^{12}, \mathbf{y}_1^1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{12}, \mathbf{y}_2^1, 0) = 0$;
$\gamma_g(\mathbf{x}_1^{21}, \mathbf{y}_1^1, 0) = 1$, $\gamma_g(\mathbf{x}_2^{21}, \mathbf{y}_2^1, 0) = 0$; $\gamma_g(\mathbf{x}_1^{22}, \mathbf{y}_1^1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{22}, \mathbf{y}_2^1, 0) = 0$;

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

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

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

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

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

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

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

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

■

4)

Remark *Los siguientes dos Ejemplos, el EjemAlgClas-2 y EjemAlgClas-3, están tomados de la base de datos Iris Plant del Machine Learning Repository de la UCI [DB1], una base de datos clásica en el área de Reconocimiento de Patrones (misma que es abordada a mayor detalle en [itzama01]). 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 EjemAlgClas-2 pertenece a la clase* C_1 *(Iris-setosa), mientras que los patrones* $\mathbf{y}^1$, $\mathbf{y}^2$ *y* $\mathbf{y}^3$ *del Ejemplo EjemAlgClas-3 pertenecen a las clases* C_2 *(Iris-versicolor),* C_1 *(Iris-setosa) y* C_3 *(Iris-virginica), respectivamente.*

Example *Sean los patrones fundamentales*

$$\mathbf{x}^1 = \begin{pmatrix} 4.8 \\ 3.4 \\ 1.6 \\ 0.2 \end{pmatrix}, \quad \mathbf{x}^2 = \begin{pmatrix} 4.4 \\ 3.0 \\ 1.3 \\ 0.2 \end{pmatrix},$$

$$\mathbf{x}^3 = \begin{pmatrix} 6.4 \\ 2.9 \\ 4.3 \\ 1.3 \end{pmatrix}, \quad \mathbf{x}^4 = \begin{pmatrix} 6.0 \\ 2.7 \\ 5.1 \\ 1.6 \end{pmatrix}, \quad \mathbf{x}^5 = \begin{pmatrix} 6.3 \\ 3.3 \\ 6.0 \\ 2.5 \end{pmatrix}, \quad \mathbf{x}^6 = \begin{pmatrix} 5.8 \\ 2.7 \\ 5.1 \\ 1.9 \end{pmatrix},$$

, agrupados en tres clases

$$C_1 = \{\mathbf{x}^1, \mathbf{x}^2\}, \quad C_2 = \{\mathbf{x}^3, \mathbf{x}^4\} \quad y \quad C_3 = \{\mathbf{x}^5, \mathbf{x}^6\}$$

; 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.

$$\mathbf{x}^1 = \begin{pmatrix} 00000000000000001111111111111111111111111111111111111111111111 \\ 1111111111111111111111111111111 \\ 000000000000000000000000000000000000000000001111111111111111 \\ 000000000000000000000011 \end{pmatrix}$$

$$\mathbf{x}^2 = \begin{pmatrix} 000000000000000000001111111111111111111111111111111111111111111111 \\ 00001111111111111111111111111111 \\ 000000000000000000000000000000000000000000000001111111111111 \\ 000000000000000000000011 \end{pmatrix}$$

$$\mathbf{x}^3 = \begin{pmatrix} 111111111111111111111111111111111111111111111111111111111111111111 \\ 00000111111111111111111111111111 \\ 000000000000000001111111111111111111111111111111111111111111 \\ 000000000001111111111111 \end{pmatrix}$$

$$\mathbf{x}^4 = \begin{pmatrix} 000011111111111111111111111111111111111111111111111111111111111111 \\ 00000001111111111111111111111111 \\ 000000001111111111111111111111111111111111111111111111111111 \\ 00000000111111111111111 \end{pmatrix}$$

$$\mathbf{x}^5 = \begin{pmatrix} 011111111111111111111111111111111111111111111111111111111111111111 \\ 01111111111111111111111111111111 \\ 111111111111111111111111111111111111111111111111111111111111111111 \\ 111111111111111111111111 \end{pmatrix}$$

$$\mathbf{x}^6 = \begin{pmatrix} 000000111111111111111111111111111111111111111111111111111111111111 \\ 00000001111111111111111111111111 \\ 000000000111111111111111111111111111111111111111111111111111 \\ 000000111111111111111111 \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} 00000000000000000011111111111111111111111111111111111111111111 \\ 11111111111111111111111111111111 \\ 000000000000000000000000000000000000000000000011111111111111 \\ 000000000000000000000111 \end{pmatrix}$$

$\rho = \bigwedge_{j=1}^{4} e_m(j) = 25$, 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 $\theta = 0$:
$\gamma_g(\mathbf{x}_1^{11}, \mathbf{y}_1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{11}, \mathbf{y}_2, 0) = 1$, $\gamma_g(\mathbf{x}_3^{11}, \mathbf{y}_3, 0) = 0$, $\gamma_g(\mathbf{x}_4^{11}, \mathbf{y}_4, 0) = 0$;
$\gamma_g(\mathbf{x}_1^{12}, \mathbf{y}_1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{12}, \mathbf{y}_2, 0) = 0$, $\gamma_g(\mathbf{x}_3^{12}, \mathbf{y}_3, 0) = 0$, $\gamma_g(\mathbf{x}_4^{12}, \mathbf{y}_4, 0) = 0$;
$\gamma_g(\mathbf{x}_1^{21}, \mathbf{y}_1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{21}, \mathbf{y}_2, 0) = 0$, $\gamma_g(\mathbf{x}_3^{21}, \mathbf{y}_3, 0) = 0$, $\gamma_g(\mathbf{x}_4^{21}, \mathbf{y}_4, 0) = 0$;
$\gamma_g(\mathbf{x}_1^{22}, \mathbf{y}_1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{22}, \mathbf{y}_2, 0) = 0$, $\gamma_g(\mathbf{x}_3^{22}, \mathbf{y}_3, 0) = 0$, $\gamma_g(\mathbf{x}_4^{22}, \mathbf{y}_4, 0) = 0$;
$\gamma_g(\mathbf{x}_1^{31}, \mathbf{y}_1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{31}, \mathbf{y}_2, 0) = 0$, $\gamma_g(\mathbf{x}_3^{31}, \mathbf{y}_3, 0) = 0$, $\gamma_g(\mathbf{x}_4^{31}, \mathbf{y}_4, 0) = 0$;
$\gamma_g(\mathbf{x}_1^{32}, \mathbf{y}_1, 0) = 0$, $\gamma_g(\mathbf{x}_2^{32}, \mathbf{y}_2, 0) = 0$, $\gamma_g(\mathbf{x}_3^{32}, \mathbf{y}_3, 0) = 0$, $\gamma_g(\mathbf{x}_4^{32}, \mathbf{y}_4, 0) = 0$; 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 NotaIris, este resultado es correcto.

■

Example *Usando el conjunto fundamental del Ejemplo EjemAlgClas-2, 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 EjemAlgClas-2, 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:

$$\mathbf{y}^1 = \begin{pmatrix} 00111111111111111111111111111111111111111111111111111111111111 \\ 00000011111111111111111111111111 \\ 000000000000011111111111111111111111111111111111111111111111 \\ 000000000000011111111111 \end{pmatrix}$$

$$\mathbf{y}^2 = \begin{pmatrix} 000000000000000001111111111111111111111111111111111111111111 \\ 00111111111111111111111111111111 \\ 000000000000000000000000000000000000000000011111111111111111 \\ 000000000000000000000011 \end{pmatrix}$$

$$\mathbf{y}^3 = \begin{pmatrix} 000000011111111111111111111111111111111111111111111111111111 \\ 000000000111111111111111111111 \\ 000000000011111111111111111111111111111111111111111111111111 \\ 000001111111111111111111 \end{pmatrix}$$

- 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 EjemAlgClas-2 como $\rho = 25$), se incrementa θ 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 EjemAlgClas-2 como $\rho = 25$), se incrementa θ 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 NotaIris, estos resultados de clasificación son correctos.

■

Example *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:

$$\mathbf{x}^1 = \begin{pmatrix} 0000000000000111 \\ 0000000000000111111 \\ 000000000001111 \\ 00000000111 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0000000000000000 \\ 0000000011111111111 \\ 000000000000000 \\ 00000000000 \end{pmatrix}$$

$$\mathbf{x}^3 = \begin{pmatrix} 0011111111111111 \\ 0001111111111111111 \\ 111111111111111 \\ 11111111111 \end{pmatrix}, \mathbf{x}^4 = \begin{pmatrix} 0000000000001111 \\ 0000000000011111111 \\ 000000000000111 \\ 00000000001 \end{pmatrix}$$

$$\mathbf{x}^5 = \begin{pmatrix} 1111111111111111 \\ 1111111111111111111 \\ 000001111111111 \\ 00000111111 \end{pmatrix}$$

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:

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

Una vez realizada la conversión, queda como sigue.

$$\mathbf{y} = \begin{pmatrix} 0000011111111111 \\ 1111111111111111111 \\ 000000001111111 \\ 00011111111 \end{pmatrix}$$

Nótese que y_3 ha sido truncado a 18, el valor correspondiente de $e_m(2)$; quedando $nmax_2 = 23$.

- Para $\theta = 0$:

$\gamma_g(\mathbf{x}_1^{11},\mathbf{y}_1,0)=0$, $\gamma_g(\mathbf{x}_2^{11},\mathbf{y}_2,0)=0$, $\gamma_g(\mathbf{x}_3^{11},\mathbf{y}_3,0)=0$, $\gamma_g(\mathbf{x}_4^{11},\mathbf{y}_4,0)=0$;
$\gamma_g(\mathbf{x}_1^{12},\mathbf{y}_1,0)=0$, $\gamma_g(\mathbf{x}_2^{12},\mathbf{y}_2,0)=0$, $\gamma_g(\mathbf{x}_3^{12},\mathbf{y}_3,0)=0$, $\gamma_g(\mathbf{x}_4^{12},\mathbf{y}_4,0)=0$;
$\gamma_g(\mathbf{x}_1^{13},\mathbf{y}_1,0)=0$, $\gamma_g(\mathbf{x}_2^{13},\mathbf{y}_2,0)=0$, $\gamma_g(\mathbf{x}_3^{13},\mathbf{y}_3,0)=0$, $\gamma_g(\mathbf{x}_4^{13},\mathbf{y}_4,0)=0$;
$\gamma_g(\mathbf{x}_1^{21},\mathbf{y}_1,0)=0$, $\gamma_g(\mathbf{x}_2^{21},\mathbf{y}_2,0)=0$, $\gamma_g(\mathbf{x}_3^{21},\mathbf{y}_3,0)=0$, $\gamma_g(\mathbf{x}_4^{21},\mathbf{y}_4,0)=0$;
$\gamma_g(\mathbf{x}_1^{22},\mathbf{y}_1,0)=0$, $\gamma_g(\mathbf{x}_2^{22},\mathbf{y}_2,0)=0$, $\gamma_g(\mathbf{x}_3^{22},\mathbf{y}_3,0)=0$, $\gamma_g(\mathbf{x}_4^{22},\mathbf{y}_4,0)=0$; 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 θ y se repiten los pasos 6 y 7.

- Para $\theta=1$:

$\gamma_g(\mathbf{x}_1^{11},\mathbf{y}_1,1)=0$, $\gamma_g(\mathbf{x}_2^{11},\mathbf{y}_2,1)=0$, $\gamma_g(\mathbf{x}_3^{11},\mathbf{y}_3,1)=0$, $\gamma_g(\mathbf{x}_4^{11},\mathbf{y}_4,1)=0$;
$\gamma_g(\mathbf{x}_1^{12},\mathbf{y}_1,1)=0$, $\gamma_g(\mathbf{x}_2^{12},\mathbf{y}_2,1)=0$, $\gamma_g(\mathbf{x}_3^{12},\mathbf{y}_3,1)=0$, $\gamma_g(\mathbf{x}_4^{12},\mathbf{y}_4,1)=0$;
$\gamma_g(\mathbf{x}_1^{13},\mathbf{y}_1,1)=0$, $\gamma_g(\mathbf{x}_2^{13},\mathbf{y}_2,1)=0$, $\gamma_g(\mathbf{x}_3^{13},\mathbf{y}_3,1)=0$, $\gamma_g(\mathbf{x}_4^{13},\mathbf{y}_4,1)=0$;
$\gamma_g(\mathbf{x}_1^{21},\mathbf{y}_1,1)=0$, $\gamma_g(\mathbf{x}_2^{21},\mathbf{y}_2,1)=0$, $\gamma_g(\mathbf{x}_3^{21},\mathbf{y}_3,1)=0$, $\gamma_g(\mathbf{x}_4^{21},\mathbf{y}_4,1)=0$;
$\gamma_g(\mathbf{x}_1^{22},\mathbf{y}_1,1)=0$, $\gamma_g(\mathbf{x}_2^{22},\mathbf{y}_2,1)=0$, $\gamma_g(\mathbf{x}_3^{22},\mathbf{y}_3,1)=0$, $\gamma_g(\mathbf{x}_4^{22},\mathbf{y}_4,1)=0$; 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 θ y se repiten los pasos 6 y 7.

- Para $\theta=2$:

$\gamma_g(\mathbf{x}_1^{11},\mathbf{y}_1,2)=0$, $\gamma_g(\mathbf{x}_2^{11},\mathbf{y}_2,2)=0$, $\gamma_g(\mathbf{x}_3^{11},\mathbf{y}_3,2)=0$, $\gamma_g(\mathbf{x}_4^{11},\mathbf{y}_4,2)=0$;
$\gamma_g(\mathbf{x}_1^{12},\mathbf{y}_1,2)=0$, $\gamma_g(\mathbf{x}_2^{12},\mathbf{y}_2,2)=0$, $\gamma_g(\mathbf{x}_3^{12},\mathbf{y}_3,2)=0$, $\gamma_g(\mathbf{x}_4^{12},\mathbf{y}_4,2)=0$;
$\gamma_g(\mathbf{x}_1^{13},\mathbf{y}_1,2)=0$, $\gamma_g(\mathbf{x}_2^{13},\mathbf{y}_2,2)=0$, $\gamma_g(\mathbf{x}_3^{13},\mathbf{y}_3,2)=0$, $\gamma_g(\mathbf{x}_4^{13},\mathbf{y}_4,2)=0$;
$\gamma_g(\mathbf{x}_1^{21},\mathbf{y}_1,2)=0$, $\gamma_g(\mathbf{x}_2^{21},\mathbf{y}_2,2)=0$, $\gamma_g(\mathbf{x}_3^{21},\mathbf{y}_3,2)=0$, $\gamma_g(\mathbf{x}_4^{21},\mathbf{y}_4,2)=0$;
$\gamma_g(\mathbf{x}_1^{22},\mathbf{y}_1,2)=0$, $\gamma_g(\mathbf{x}_2^{22},\mathbf{y}_2,2)=0$, $\gamma_g(\mathbf{x}_3^{22},\mathbf{y}_3,2)=0$, $\gamma_g(\mathbf{x}_4^{22},\mathbf{y}_4,2)=1$; 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 .

■

Modelo Propuesto

En este capítulo se presentan las principales aportaciones del presente trabajo de tesis. En primer lugar, se presenta la caracterización del clasificador Gamma, con respecto a la forma que toma su conjunto fundamental para el caso base, así como con respecto a las observaciones realizadas durante la experimentación y aplicación del clasificador a diversos problemas ---incluso algunos casos para los cuales no fue diseñado. Posteriormente, se utiliza la caracterización presentada para expresar algunas propiedades que pueden presentar los conjuntos fundamentales y sus efectos en las condiciones suficientes para clasificación correcta. Finalmente, se aborda una característica de los clasificadores de patrones que resulta de particular interés para el Enfoque Asociativo de Reconocimiento de patrones: el factor de olvido.

Caracterización del Clasificador Gamma

La base para caracterizar la operación del clasificador Gamma es un uso extenso del mismo en diversas tareas de reconocimiento de patrones y con diferentes bases de datos. En este sentido, este clasificador ha sido utilizado para atacar tareas como la clasificación de patrones (para la cual fue diseñado), la predicción de datos desconocidos, así como la interpolación y extrapolación de funciones (mismas que en su conjunto se pueden agrupar como aproximación de funciones), tareas éstas últimas para las cuales este algoritmo no fue inicialmente diseñado.

Ahora pasemos a describir cómo se comporta el clasificador al aplicarlo a diversos problemas. Primero, se observa que si los datos inducen una función, es posible esperar resultados favorables. Dicho de otra forma, si a cada patrón de entrada $\mathbf{x}^{\mu}$ la corresponde uno y sólo un patrón de salida o clase $\mathbf{y}^{\mu}$, se puede esperar un buen desempeño: competitivo o incluso superior al de otros algoritmos. Sin embargo, si los datos corresponden a una relación en la cual un patrón de entrada $\mathbf{x}^{\mu}$ puede tener asignadas más de una clase $\mathbf{y}^{\mu}$, se presenta una situación no prevista por el algoritmo original. Dependiendo de cómo se adapte el algoritmo para tratar esta situación, serán favorables o no los resultados.

Un ejemplo de lo anterior sucedió al aplicar el clasificador Gamma a una base de datos de estimación del esfuerzo durante el desarrollo de programas pequeños, cuyo resultado está en proceso de ser publicado. En dicha base de datos, los patrones de entrada $\mathbf{x}^{\mu}$ eran vectores enteros unidimensionales (en un primer experimento) y vectores enteros bidimensionales (en un segundo experimento); esto es, $\mathbf{x} \in \mathbb{Z}$ y $\mathbf{x} \in \mathbb{Z}^2$, respectivamente. Los patrones de salida o clases eran a su vez vectores enteros unidimensionales (en ambos experimentos): las clases o patrones de salida eran números enteros.

El problema fue que en varias instancias, un mismo valor de entrada correspondía a valores de salida diferentes. Para poder abordar este problema, el algoritmo fue modificado para trabajar con un solo patrón de salida que resumiera todos los patrones de salida asociados; en este caso, se usó el promedio de los valores de salida. Nótese que si los valores de salida son muy diferentes, el promedio dará un error relativamente alto. A pesar de lo anterior, los resultados obtenidos fueron competitivos con todos los métodos con los que se ha trabajado

esta base de datos, siendo en algunos casos superiores los exhibidos por el clasificador Gamma. Sin embargo, esta situación evidenció las carencias inherentes del algoritmo para tratar con este tipo de conjuntos de datos.
Cabe señalar que, en general, los algoritmos clasificadores de patrones no suelen ser diseñados para atacar este tipo de problemas. Asimismo, la gran mayoría de las bases de datos utilizadas en Reconocimiento de Patrones y *Machine Learning* inducen una función. Se sugiere entonces asegurarse primero de que los datos a tratar corresponden a una función. En caso de tener que tratar con datos correspondientes a una relación, será necesario diseñar un mecanismo adecuado para manipular dichos datos. A continuación se define el conjunto fundamental que el clasificador Gamma utilizará para el caso base (en otras palabras, el caso ideal).

Definition *Sea el conjunto fundamental del clasificador Gamma el conjunto de patrones asociados a una clase, de la forma* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1, 2, \ldots, p\}$ *; donde* $\mathbf{x}^\mu$ *es un patrón y* $\mathbf{y}^\mu$ *es su clase correspondiente. Además, para este conjunto fundamental se cumplen las siguientes tres afirmaciones:*

$$\mathbf{x}^i \neq \mathbf{x}^j \ \forall i, j \in \{1, 2, \ldots, p\} \text{ tal que } i \neq j$$

Esto implica que no hay patrones repetidos.

$$\mathbf{x}^i = \mathbf{x}^j \rightarrow \mathbf{y}^i = \mathbf{y}^j \ \forall i, j \in \{1, 2, \ldots, p\}$$

Un patrón dado no puede tener asociada más de una clase.

$$\mathbf{y}^i \neq \mathbf{y}^j \rightarrow \mathbf{x}^i \neq \mathbf{x}^j \ \forall i, j \in \{1, 2, \ldots, p\}$$

Clases diferentes tienen asociados patrones diferentes.

Dicho de otra manera, el conjunto fundamental debe inducir una relación entre el conjunto de patrones y el conjunto de clases, de tal manera que dicha relación cumpla con las características de una función.

Otra observación interesante es que, a pesar de que el clasificador Gamma fue originalmente diseñado pensando únicamente en la tarea de clasificación de patrones, ha exhibido un desempeño competitivo incluso en tareas como la predicción y extrapolación. Tal fue el caso al aplicar el algoritmo a bases de datos ambientales referentes a la calidad del aire en la Ciudad de México, tomadas del Sistema de Monitoreo Atmosférico de la Ciudad de México (SIMAT), con la intención de predecir el valor de una variable ambiental en un futuro dado.
En este caso, se tomaron las bases de datos publicadas por el SIMAT acerca de los valores históricos de las variables utilizadas para evaluar la calidad ambiental del aire en la Ciudad de México. Luego, se eligió una de estas variables para una estación de monitoreo, procediéndose a formar patrones con una sucesión contigua de datos, de un tamaño dado (por ejemplo, $n = 10$), asignándole como clase el dato siguiente.
Como puede verse, la asignación de clase es un tanto artificial, más por la necesidad de formular el problema como uno de clasificación que por la naturaleza misma de éste. A pesar de lo anterior, los resultados obtenidos fueron sumamente competitivos. Cabe mencionar que en el caso general, puede llegar a suceder algo similar al presentado en la observación anterior; a saber, que los datos representen una relación y no una función. Sin

embargo, por las características de los datos y el hecho de que se trata con cantidades relativamente altas de patrones (365 días $\times$ 24 horas = 8760 muestras por año, tamaño que se utilizó para el conjunto fundamental), es poco común que un mismo patrón (secuencia de n valores consecutivos) se repita, con un valor siguiente diferente. En lugar de ser ese un problema, se observó que para esta tarea ---en particular planteada de la manera antes mencionada--- los puntos más problemáticos son aquellos en los que existe un cambio drástico en los valores. Visto gráficamente, esto sucede donde existe un punto extremo en la gráfica, ya sea un máximo o un mínimo local.

Por otro lado, se observó que los resultados arrojados por el clasificador Gamma son, a final de cuentas, valores conocidos. En caso de que se presente una secuencia conocida, se dará como salida el valor conocido que mejor corresponda a esa secuencia. Sin embargo, es claro suponer que puede darse el caso de que dicha secuencia no sea exactamente igual a una conocida, sino que más bien quede cercana a la frontera entre dos (o más) secuencias conocidas. Entonces, se puede argumentar que la salida también debería ser cercana a la frontera entre la salidas conocidas correspondientes. Si se puede combinar la información de las salidas correspondientes a la secuencia presentada que se encuentran más cercanas, qué tan cercanas están, y los valores de dichas salidas, para así arrojar una salida consensuada aunque desconocida, probablemente este resultado se pueda aplicar en tareas de interpolación y, de manera más general, en tareas de aproximación de funciones.

Otro aspecto de gran relevancia en cuanto a la caracterización del clasificador Gamma es cómo induce las fronteras entre clases, a partir de un conjunto fundamental. Estas fronteras resultan de gran importancia para entender cómo asigna una clase a un patrón dado. Sin embargo, para realizar este análisis es necesario contar con algunas herramientas, como son las definiciones de relaciones entre clases: puntualmente separables, puntualmente segmentables y puntualmente no separables. Estas definiciones se presentan en la siguiente sección, por lo que la caracterización de las fronteras entre clases se verá en dicha sección.

Finalmente, está un aspecto de los clasificadores poco analizado: la capacidad de aprendizaje. En este caso, el clasificador Gamma tiene, en teoría, una capacidad de aprendizaje infinita. Esto se debe a que ---de manera smiliar a los modelos asociativos alfa-beta--- por la forma en que está planteado, no existe límite teórico al tamaño del conjunto fundamental.
Sin embargo, si en efecto se tratara de entrenar al clasificador Gamma con un conjunto fundamental de tamaño infinito, nunca terminaría de aprender los patrones fundamentales, así que nunca llegaría a tratar de clasificar algún patrón. Así que, en la práctica, siempre se tendrán conjuntos fundamentales finitos.
Por otro lado, es claro que si el clasificador no tiene límites impuestos por su fundamentación y planteamiento teórico, sí los tendrá por parte de la implementación. Así, un hardware con capacidades de memoria o de cómputo pequeñas podrá trabajar solamente conjuntos fundamentales pequeños, mientras que una supercomputadora o un cluster podrán trabajar problemas altamente dimensionales, con conjuntos fundamentales de tamaños enormes. Por lo tanto, los límites de los problemas que el clasificador Gamma puede tratar están dados más por la implementación (software) y el equipo (hardware) en el

que se esté ejecutando.
Cabe mencionar que el clasificador Gamma ha sido diseñado pensando en su uso en equipos de escritorio convencionales, utilizando bases de datos típicas del área de Reconocimiento de Patrones ---como las que se encuentran en el *Machine Learning Repository* de la UCI [DB1]---, con conjuntos fundamentales finitos de tamaño menor al millón de patrones, generalmente con dimensionalidades bajas: sin pasar de 1000 rasgos. Sin embargo, si el clasificador Gamma se implementa en una supercomputadora, será posible tratar problemas más complejos.

Clasificación Correcta

Antes de proceder a expresar las condiciones suficientes para clasificación correcta, veamos algunas propiedades que pueden presentar las clases que conforman los conjuntos fundamentales. Su importancia radica en que, dependiendo de cuáles propiedades cumpla, un conjunto fundamental puede garantizar recuperación correcta o no.

Definition *Sean dos clases diferentes* $\mathbf{y}^a$ *y* $\mathbf{y}^b$ *, con patrones asociados* $\mathbf{x}^{a\omega}$ *y* $\mathbf{x}^{b\omega}$ *, respectivamente, con* ω *siendo un índice arbitrario. Se define la relación entre estas clases como feature-wise linealmente separables en la dimensión* D *si se cumple que:*

$$\exists i \in D \text{ tal que } x_D^{a\omega} < i < x_D^{b\omega} \quad i \notin [\bigwedge x_D^{a\omega}, \bigvee x_D^{a\omega}], i \notin \left[\bigwedge x_D^{b\omega}, \bigvee x_D^{b\omega}\right]$$

Remark *Dado que el clasificador Gamma trabaja cada rasgo por separado, es necesario hacer la separabilidad entre clases también rasgo a rasgo: esto es, feature-wise. Es por ello que en lo sucesivo, cualquier comparación que se haga feature-wise linealmente se denotará como puntualmente.*

Definition *Sean dos clases diferentes* $\mathbf{y}^a$ *y* $\mathbf{y}^b$ *; se define la relación entre estas clases como puntualmente separables totalmente si son puntualmente separables para cada una de las* n *dimensiones de los patrones involucrados.*

Definition *Sean dos clases diferentes* $\mathbf{y}^a$ *y* $\mathbf{y}^b$ *; se define la relación entre estas clases como puntualmente separables parcialmente si son puntualmente separables para al menos una de las* n *dimensiones de los patrones involucrados, pero no para todas.*

Definition *Sean dos clases diferentes* $\mathbf{y}^a$ *y* $\mathbf{y}^b$ *; se define la relación entre estas clases como puntualmente segmentables en la dimensión* D *si es posible particionar dicha dimensión en segmentos* $[\delta_0, \delta_f]$ *, dentro de cada uno de los cuales* $\mathbf{y}^a$ *y* $\mathbf{y}^b$ *son puntualmente separables en la dimensión* D *.*

Definition *Sean dos clases diferentes* $\mathbf{y}^a$ *y* $\mathbf{y}^b$ *, con patrones asociados* $\mathbf{x}^{aj}$ *y* $\mathbf{x}^{bk}$ *, respectivamente; se define la relación entre estas clases como puntualmente no separables en la dimensión* D *si se cumple que:*

$$\exists i \in D \text{ tal que } x_D^{aj} = i = x_D^{bk}$$

A continuación se presenta un lema relacionado con el funcionamiento del clasificador Gamma al presentársele un patrón conocido para su clasificación. Este lema será utilizado posteriormente para fundamentar las condiciones suficientes para clasificación correcta.

Lemma *Al presentarse un patrón conocido* $\tilde{\mathbf{x}} = \mathbf{x}^\sigma$ *asociado a la clase* $\mathbf{y}^a$ *al*

clasificador Gamma, durante la fase de clasificación, este patrón aportará n *valores* 1 *a la suma ponderada de la clase* c_a *para cualquier valor de* $\theta \geq 0$ *, donde* $\mathbf{x}^{\sigma} = \mathbf{x}^{ab}$ *.*

Proof *Por la forma de calcular la suma ponderada* c_i *correspondiente a la* i *-ésima clase (como se ve en AlgClasPropuestoGen), se tendrá como resultado la suma de los valores* $\gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right) = 1$ *dividida entre la cardinalidad de la clase* k_i *. Pero por definición tenemos que*

$$\gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right) = 1 \longrightarrow m - u_\beta\left[\alpha\left(x_j^{i\omega}, \tilde{x}_j\right) \bmod 2\right] \leq \theta$$

Es claro que el caso más restrictivo se presenta cuando $\theta = 0$ *, puesto que si la condición se cumple para ese valor de* θ *, también se cumplirá para cualquier valor mayor.*
Pero el hecho de que $\gamma_g\left(x_j^{i\omega}, y_j, 0\right) = 1$ *implica que* $x_j^{i\omega} = y_j$ *como se hace ver en la nota nota-Gama-patronesIguales.*
Ahora bien, como $\tilde{\mathbf{x}}$ *es un patrón conocido* $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma}$ *que a su vez pertenece a la* a *-ésima clase pues* $\mathbf{x}^{\sigma} = \mathbf{x}^{ab}$ *, por transitividad tenemos que* $\tilde{\mathbf{x}} = \mathbf{x}^{ab}$ *. Por otro lado, al calcular* $\gamma_g\left(x_j^{i\omega}, \tilde{x}_j, 0\right)$ *para los índices* $i = a$ *y* $\omega = b$ *tenemos que calcular* $\gamma_g\left(x_j^{ab}, \tilde{x}_j, \theta\right)$ *.*
Pero como $\tilde{\mathbf{x}} = \mathbf{x}^{ab}$ *entonces* $\tilde{x}_j = x_j^{ab}$ *para todas las componentes. Entonces*

$$\gamma_g\left(x_j^{ab}, \tilde{x}_j, 0\right) = 1 \ \forall j \in \{1, 2, \ldots, n\}$$

Así, tendremos n *componentes para las que el operador Gamma de similitud generalizado da como resultado* 1 *. Asimismo, es claro que estos resultados se suman a* c_a *. Por lo tanto, el presentar al clasificador Gamma el patrón conocido* $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma}$ *que pertenece a la* a *-ésima clase afecta con un valor de* $n \times 1 = n$ *en la suma ponderada* c_a *.*
Para que la clase asignada al patrón conocido $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma} = \mathbf{x}^{ab}$ sea efectivamente $\mathbf{y}^a$, esto es, el patrón fundamental sea clasificado correctamente, es necesario que se cumpla lo siguiente:

$$c_a = \bigvee_{i=1}^{m} c_i$$

o lo que es lo mismo, para un valor dado de θ :

$$\frac{\sum_{\omega=1}^{k_a} \sum_{j=1}^{n} \gamma_g\left(x_j^{a\omega}, \tilde{x}_j, \theta\right)}{k_a} = \bigvee_{i=1}^{m} \frac{\sum_{\omega=1}^{k_i} \sum_{j=1}^{n} \gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right)}{k_i}$$

Por otro lado, se sabe por la definición def-CF que existe uno y sólo un patrón fundamental igual a $\tilde{\mathbf{x}}$: $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma}$. Lo anterior garantiza que, aunque pueden existir otros patrones fundamentales con algunas componentes cuyos valores sean iguales a algunas de las componentes de $\tilde{\mathbf{x}}$, sólo $\mathbf{x}^{\sigma}$ tendrá todas las componentes iguales.

A continuación se presenta un teorema que fundamenta una de las condiciones suficientes para clasificación correcta de un patrón fundamental (conocido).

Theorem *Al presentarse un patrón conocido* $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma} = \mathbf{x}^{ab}$ *asociado a la clase* $\mathbf{y}^{a}$ *al clasificador Gamma, durante la fase de clasificación; si la clase* $\mathbf{y}^{a}$ *es puntualmente separable totalmente de todas las otras clases, entonces este patrón será clasificado correctamente.*

Proof *Por el lema lem-RecPatCon, se sabe que* $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma}$ *aportará* n *a la suma ponderada* c_a *. Por otro lado, por hipótesis la clase* $\mathbf{y}^{a}$ *es puntualmente separable del resto de las clases, por lo que* $x_j^{ab} \neq x_j^{h\omega}$ *para todos los patrones pertenencientes a cualquier clase diferente de* $\mathbf{y}^{a}$ *, en la* j *-ésima dimensión. Pero como lo anterior sucede para todas las dimensiones, se puede afirmar que las únicas componentes* $x_j^{i\omega}$ *que podrían ser iguales a* x_j^{ab} *pertenecen también a* $\mathbf{y}^{a}$ *:* $x_j^{a\omega}$ *.*
Sin embargo, por la definición def-CF se sabe que no hay patrones repetidos, por lo que el único patrón con todas las componentes iguales a las de $\mathbf{x}^{ab}$ *es el mismo* $\mathbf{x}^{ab}$ *.*
Ahora bien, para $\theta = 0$ *se requiere que* $x_j^{i\omega} = x_j^{ab}$ *para que* $\gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right) = 1$ *. Esto último sucederá para cada una de las* n *componentes de* $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma} = \mathbf{x}^{ab}$ *, más todas aquellas componentes* $x_j^{ab} = x_j^{a\omega}$ *. Como estas componentes también aportan a* c_a *, se tiene que* $c_a \geq n$ *, mientras que* $c_h = 0$ *para toda suma ponderada* $c_h \neq c_a$ *. Así,*

$$c_a \geq n > 0 = c_h \ \forall h \neq a$$

por lo que

$$c_a = \bigvee_{i=1}^{m} c_i$$

Por lo tanto, $\tilde{\mathbf{x}} = \mathbf{x}^{\sigma} = \mathbf{x}^{ab}$ *es clasificado dentro de* $\mathbf{y}^{a}$ *, lo cual es una clasificación correcta.*

Obsérvese que este teorema fundamenta unas condidciones de clasificación correcta bastante estrictas: es necesario que en todas las dimensiones, los patrones de cada clase estén agrupados en rangos diferentes de la dimensión. Es claro que en la práctica resulta complicado que estas condiciones se cumplan. Más aún, el teorema está planteado para patrones fundamentales, por lo que no se puede aplicar a patrones desconocidos. Dado que en la mayoría de las aplicaciones, el conjunto de mayor interés es el de prueba (integrado normalmente por patrones desconocidos), conocer las condiciones de recuperación correcta (o incorrecta) de los patrones no fundamentales es muy importante.

Fronteras entre Clases

Antes de continuar con las condiciones de clasificación, analicemos cómo induce las fronteras entre clases el clasificador Gamma. Este análisis es relevante ya que nos permitirá fundamentar las condiciones de clasificación.

Dependiendo de las relaciones ente clases del conjunto fundamental, podemos tener varios casos. La frontera entre las clases se verá fuertemente afectada por tales relaciones.

Todas las clases son puntualmente separables totalmente. Este es el caso ideal, como lo indica el teorema teor-ClasPat-PtSepTot. Un ejemplo de esta situación lo podemos ver en la figura FigFront01, en la que se muestra una dimensión de un conjunto fundamental en donde todas las clases son puntualmente separables. Cuando esto sucede, las fronteras quedan en el punto medio entre los valores extremos de las clases, indicado por medio de las líneas negras en la figura. Por su parte, los rectángulos de colores indican el rango de la dimensión que corresponde a cada clase. Recordemos que aunque en la figura sólo se ejemplifica una dimensión, esta situación se repite para todas las dimensiones en el Caso I.

Las clases son puntualmente separables parcialmente. Similar al Caso I (para algunas de las dimensiones), aunque no tan bueno. Existen tres diferentes posibilidades.

En algunas dimensiones puntualmente separables y en otras dimensiones puntualmente segmentables. Al menos una de las dimensiones es puntualmente separable (como en la figura FigFront01), mientras que al menos una dimensión es puntualmente segmentable y no hay dimensiones no separables. Un ejemplo de una dimensión puntualmente segmentable se aprecia en la figura FigFront02. Nótese cómo no es posible separar las clases *Clase 2* y *Clase 3* con una frontera única; pero sí es posible particionar el rango que comparten en varios segmentos, dentro de los cuales es posible establecer una frontera entre las clases. De esta manera, tenemos fronteras que separan sin ambigüedad las clases, aunque ahora tenemos más fronteras que en el Caso I (figura FigFront02).

En algunas dimensiones puntualmente separables y en otras dimensionas puntualmente no separables. Al menos una de las dimensiones es puntualmente separable (como en la figura FigFront01), mientras que al menos una dimensión es puntualmente no separable. Un ejemplo de una dimensión puntualmente no separable se muestra en la figura FigFront03. Las líneas rojas indican fronteras compartidas por ambas clases, debido a que ambas clases tienen valores coincidentes en esa posición.
Si observamos los valores 4.6, 4.8 y 5.1, nos daremos cuenta que tanto la *Clase 2* como la *Clase 3* tienen patrones con esos valores. En el caso de 4.6, la *Clase 2* tiene 7 patrones, contra 1 patrón de la *Clase 3*; aunque hay ambigüedad, incluso con un valor de $\theta = 0$ es posible obtener un máximo único para esta dimensión, correspondietne a la *Clase 2*. Para el valor 5.1 pasa algo similar: la *Clase 2* tiene 1 patrón mientras que la *Clase 3* tiene 7 patrones con ese valor, por lo que ahora el resultado sería a favor de la *Clase 3*. Sin embargo, el valor 4.8 es diferente: ambas clases tienen la misma cantidad de patrones, que es 2 ; para resolver esta ambigüedad es necesario incrementar θ a 1 , con lo que las clases tienen 9 y 5 patrones similares, respectivamente, por lo que la clase favorecida sería la *Clase 2*.

En algunas dimensiones puntualmente separables, en otras dimensiones puntualmente segmentables y en otras dimensionas puntualmente no separables. Al menos una de las dimensiones es puntualmente separable (como en la figura FigFront01), mientras que al menos una dimensión es puntualmente segmentable (como en la figura FigFront02) y al menos una dimensión es puntualmente no separable como en la figura FigFront03). Este caso es una combinación de los dos anteriores. Para el caso general de las aplicaciones reales, éste es el más probable de presentarse.

Las clases son puntualmente segmentables en todas las dimensiones. Aunque las clases no son puntualmente separables como en el Caso I, sí son puntualmente segmentables en todas sus dimensiones. Similarmente a como sucede en el Caso II.1, esta situación no es tan negativa, puesto que es posible establecer fronteras no ambiguas entre las clases. En este caso, se presentaría la situación ejemplificada por la figura FigFront02 en cada una de las dimensiones.

Las clases son puntualmente segmentables en algunas dimensiones y puntualmente no separables en otras. Similar al Caso II.2, sólo que ahora se tienen dimensiones puntualmente segmentables en lugar de puntualmente separables. Así, algunas dimensiones se parecerán a la mostrada en la figura FigFront02, mientras que otras se parecerán más a lo mostrado en la figura FigFront03.

Las clases son puntualmente no separables en todas las dimensiones. Sin lugar a dudas, este es el caso más complicado de los 5. Dado que todas las dimensiones son no separables, en todas las dimensiones habrá coincidencias entre patrones de clases diferentes, lo que genera ambigüedad. Para valores alejados de las fronteras, es posible obtener resultados esperables, de acuerdo con los rangos definidos para cada clase por las fronteras. En cuanto a los valores cercanos a las fronteras, la ambigüedad causada por las coincidencias hace necesario utilizar el algoritmo completo para determinar el resultado de la clasificación.

Nótese que al definirse el valor de la frontera como el punto medio entre los valores extremos entre las clases, hay tres posibilidades para este valor, como se analiza a continuación.

El punto medio es un valor no válido. Situación preferible; la figura FigFront01 nos muestra un ejemplo de este caso en la frontera entre las clases *Clase 2* y *Clase 3*: los valores extremos son 1.8 y 2.1, respectivamente, con el punto medio igual a 1.95. Cuando esto sucede, la frontera queda entre dos valores válidos en la dimensión, por lo que no es posible que algún patrón tome el valor 1.95, igual a la frontera. Así, todos los valores cercanos a la frontera caerán de uno u otro lado, sin ambigüedad. El valor de la frontera queda excluido de pertenecer a alguna clase.

El punto medio es un valor válido desconocido. En el ejemplo representado en la figura FigFront01, sucede entre las clases *Clase 1* y *Clase 2*, en donde los valores extremos son 0.6 y 1.0, respectivamente. El punto medio es 0.8, que es un valor válido en esa dimensión, aunque ningún patrón fundamental toma dicho valor. Cuando esto sucede, la frontera queda en un valor válido para la dimensión. Así, los valores cercanos a la frontera (pero diferentes) caerán de uno u otro lado, sin ambigüedad; pero es posible que un patrón desconocido presente un valor de 0.8, igual al valor de la frontera. En este caso, no es posible determinar a qué clase se asignará este patrón (en esta dimensión) sin conocer la situación de todos los valores del patrón: es necesario ejecutar el algoritmo completo del clasificador. Obsérvese que los rangos de cada clase ---indicados por los rectángulos de colores--- no incluyen el valor de las fronteras de los

casos A y B, como se puede apreciar en las fronteras de las figuras FigFront01 y FigFront02, así como en la frontera entre las clases *Clase 1* y *Clase 2* de la figura FigFront03 (todas ellas indicadas en color negro). En principio, el valor de estas fronteras también está excluido de las clases, pero cuando un patrón toma este valor, la ambigüedad de la frontera hace necesario incluir más información para llegar a una decisión.

El punto medio es un valor válido conocido. Este es el peor caso de los 3; sucede cuando hay coincidencias en los valores de patrones pertenencientes a clases diferentes, como por ejemplo en los valores 4.5 y 4.8 a 5.1 de la figura FigFront03. Ahora no sólo es posible que algún patrón desconocido tome el valor de la frontera, cayendo en una situación de ambigüedad, sino que hay patrones conocidos que toman este valor, introduciendo ambigüedad en el conjunto fundamental. Estas fronteras están indicadas en la figura FigFront03 por medio de líneas rojas. Nótese como el rango de la clase *Clase 2* incluye los valores de las fronteras en 4.5 y 4.8, mientras que el rango de la *Clase 3* incluye las fronteras de 4.9, 5.0 y 5.1. El valor de estas fronteras tiene más ambigüedad que en el Caso B, pero habrá situaciones en las que se pueda asignar directamente a alguna clase, mientras que en otras circunstancias será necesario ejecutar el algoritmo completo para clasificar al patrón.

Del análisis anterior se desprenden algunas conclusiones:

- Desde el punto de vista del conjunto fundamental:
  - El Caso I es el ideal: garantiza la clasificación correcta de los patrones fundamentales, además de ofrecer fronteras entre clases que son claras y con poca ambigüedad.
  - El Caso V es el menos deseable: todas las dimensiones tienen fronteras entre clases que introducen ambigüedad en la decisión.
  - Para propósitos de clasificación correcta, las relaciones entre clases puntualmente separables y puntualmente segmentables son prácticamente equivalentes, con la diferencia de que para las puntualmente segmentables habrá rangos de una clase intercalados con los de otras.
- Desde el punto de vista de las fronteras:
  - El Caso A es el ideal: garantiza que ningún patrón podrá tomar el mismo valor de la frontera, evitando que dicha forntera introduzca más ambigüedad a la decisión.
  - El Caso C es el menos deseable: no es posible saber qué clase tendrá más patrones similares al patrón desconocido sin conocer los valores cercanos a la frontera ---si es posible desambiguar con la información de la misma dimensión--- o, incluso, sin ejecutar el algoritmo completo ---si la información proporcionada por la dimensión no es capaz de desambiguar la decisión.
  - Los valores cercanos a las fronteras corren mayores riesgos de incluir ambigüedad.
  - Los valores iguales a los de las fronteras seguramente agregarán cierta cantidad de ambigüedad a la decisión.

Algunos de los hallazgos antes mencionados son bastante conocidos en el área de Clasificación de Patrones. En particular, se sabe que los patrones cercanos a las fronteras suelen ser más difíciles de clasificar que los patrones alejados de ellas [PR-patclas18]. Cabe mencionar que otra categoría de patrones que suelen ocasionar problemas a los clasificadores (sin importar de cuál se trate) son los patrones no característicos o excepcionales. Estos patrones tienden a ser tan diferentes de los demás patrones que

pertenecen a la misma clase, que pueden incluso llegar a ser más similares a patrones de otras clases.
Por ejemplo, tomemos la conocida base de datos Iris Plant, publicada en el *Machine Learning Repository* de la UCI [DB1]. En la figura FigFront04 podemos apreciar el histograma de la dimensión Longitud del Sépalo. Para el valor 4.9 se observa que las tres clases ---*Iris-setosa*, *Iris-versicolor* e *Iris-virginica*--- presentan patrones con ese valor. Sin embargo, la clase *Iris-setosa* tiene 4 patrones con este valor, mientras que ambas *Iris-versicolor* e *Iris-virginica* tienen sólo 1. Además, la media de los valores de longitud del sépalo para cada clase son: 5.0, 5.9 y 6.6, respectivamente. Es claro ---tanto por la cantidad de patrones, como por la cercanía a la media, además de lo que muestra la figura--- que este valor de 4.9 es bastante normal para la clase *Iris-setosa*. Para la clase *Iris-versicolor* no es tan claro: aun es posible considerarlo un patrón normal, aunque en un extremo de los valores de la clase. En cuanto a la clase *Iris-virginica*, este valor corresponde a un patrón excepcional, puesto que está sumamente alejado de la media de los valores para la clase, aislado entre valores correspondientes a otras clases, y con un rango no despreciable de valores para los que no hay patrones en esta clase entre dicho patrón y el resto de los patrones de la clase. Así, podemos afirmar que en lo que respecta a la variable longitud del sépalo, el patrón (4.9, 2.5, 4.5, 1.7) es una excepción.

Ahora bien, si resulta claro que los patrones más problemáticos son aquellos con valores cercanos a las fronteras (o incluso con valores del otro lado de la frontera ---dentro del rago de otra clase--- como sucede con los patrones excepcionales), ¿acaso se puede hacer algo para mejorar la robustez del clasificador ante ellos? La respuesta es: depende. En el caso de los patrones excepcionales, no es mucho lo que se puede hacer, ya que la causa de la ambigüedad (o el error) en la clasificación radica en el patrón en sí.
Para el caso de los patrones que no son excepcionales, pero sí tienen valores cercanos o iguales a los de las fronteras entre clases, hay esperanzas. Continuando con el ejemplo de la base de datos Iris Plant, veamos las figuras FigFront05 y FigFront06, en las que se muestran los histogramas por clase para los rasgos ancho del sépalo y logitud del pétalo, respectivamente. Nótese que las fronteras entre clases están indicadas por medio de líneas negras, ignorándose las fronteras causadas por coincidencias.

En la figura FigFront05 podemos apreciar cómo más de la mitad de los valores que los patrones fundamentales toman, presentan coincidencias en patrones de diferentes clases. Por supuesto, lo anterior genera ambigüedad y error en la clasificación. Así, si sólo se toma en cuenta esta dimensión, todos los patrones que pertenecen a las clases *Iris-setosa* e *Iris-virginica* cuyos valores de ancho del sépalo sean menores a 2.8 cm serán clasificados como pertenecientes a la clase *Iris-versicolor*. Esta situación se repite en todos los rangos de clase. En total, se tendrían 62 patrones clasificados incorrectamente y 88 clasificados correctamente, lo que equivale a un desempeño de 58.67% de clasificación correcta. Esto, por supuesto, es un desempeño muy bajo.
Por otro lado, en el caso de la longitud del pétalo (figura FigFront06), sólo

10.42%--- de los valores tomados por patrones fundamentales tienen coincidencias de diferentes clases. De hecho, como la clase *Iris-setosa* es puntualmente separable de las otras dos, todos los patrones de esta clase se clasifican correctamente. Incluso los patrones complicados de las clases *Iris-versicolor* e *Iris-virginica* son pocos: sólo 7 patrones se clasificarían incorrectamente si sólo se toma en cuenta esta dimensión; lo anterior equivale a 95.33% de clasificación correcta. Esto, como era de esperarse, es un buen desempeño. Como queda evidenciado en el análisis anterior, una dimensión en la que todas las clases sean puntualmente separables es preferible a una en la que las clases (o al menos algunas clases) sean puntualmente no separables. Dicho de otra manera, es preferible una dimensión que introduce poca ambigüedad y poco error a la clasificación, sobre otra dimensión que introduce mucha ambigüedad o mucho error. Una manera de expresar esta preferencia y lograr que el error generado por una dimensión en la que las clases son puntualmente no separables, es incluir un factor de pesos a la suma ponderada del algoritmo de clasificación. La idea es premiar aquellos rasgos que sean preferibles por introducir poca ambigüedad y poco error, mientras se castiga a los rasgos que agregan ambigüedad o error. Para ilustrar esta idea, veamos algunos ejemplos, tomados de la base de datos Iris Plant [DB1].

Example *Sea el conjunto fundamental integrado por todos los patrones de la base de datos Iris Plant, excepto el patrón* $\mathbf{x}^{135} = \begin{pmatrix} 6.1 \\ 2.6 \\ 5.6 \\ 1.4 \end{pmatrix}$ *que pertenece a la clase Iris-virginica (* C_3 *); esto es,* $\mathbf{y}^{135} = C_3$ *. Clasificar el patrón* $\tilde{\mathbf{x}} = \mathbf{x}^{135} = \begin{pmatrix} 6.1 \\ 2.6 \\ 5.6 \\ 1.4 \end{pmatrix}$.

Siguiendo el algoritmo original del clasificador Gamma (ver AlgClasPropuestoGen), el resultado de calcular las sumas ponderadas de las similitudes del patrón a clasificar con respecto a los patrones fundamentales es como sigue:

Para $\theta = 0$:

$$c_1 = \frac{0+0+0+0}{50} = \frac{0}{50} = 0.000$$
$$c_2 = \frac{4+3+0+7}{50} = \frac{14}{50} = 0.280$$
$$c_3 = \frac{1+1+5+0}{49} = \frac{7}{49} = 0.143$$

Dado que $\bigvee_{i=1}^{3} c_i = c_2 = \frac{14}{50} =$ 0.280, se asigna la clase C_2 *Iris-versicolor*. Pero $\mathbf{y}^{135} = C_3 \neq C_2 = \tilde{\mathbf{y}}$, por lo que la clasificación es incorrecta.

Sin embargo, es posible obtener una clasificación correcta si utilizamos diferentes pesos para cada rasgo. Por ejemplo, si se utilizan los siguientes pesos: 0, 0, 1, y 0.5,

respectivamente, se obtienen los siguientes valores para las sumas ponderadas.
Para $\theta = 0$:

$$c_1 = \frac{(0*0)+(0*0)+(0*1)+\left(0*0.5\right)}{50} = \frac{0}{50} = 0.000$$
$$c_2 = \frac{(4*0)+(3*0)+(0*1)+\left(7*0.5\right)}{50} = \frac{3.5}{50} = 0.070$$
$$c_3 = \frac{(1*0)+(1*0)+(5*1)+\left(0*0.5\right)}{49} = \frac{5}{49} = 0.102$$

Como ahora $\bigvee_{i=1}^{3} c_i = c_3 = \frac{5}{50} =$ 0.102, se asigna la clase C_3 *Iris-virginica*: $\mathbf{y}^{135} = C_3$, $C_3 = \tilde{\mathbf{y}}$, por lo que la clasificación es correcta.
■

Example *Sea el conjunto fundamental integrado por todos los patrones de la base de datos Iris Plant. Clasificar el patrón* $\tilde{\mathbf{x}} = \mathbf{x}^{135} = \begin{pmatrix} 6.1 \\ 2.6 \\ 5.6 \\ 1.4 \end{pmatrix}$ *que pertenece a la clase Iris-virginica (* C_3 *); esto es,* $\mathbf{y}^{135} = C_3$.
En este caso, el patrón $\mathbf{x}^{135}$ es un patrón fundamental. Siguiende el algoritmo original, obtenemos los siguientes valores para las sumas ponderadas.
Para $\theta = 0$:

$$c_1 = \frac{0+0+0+0}{50} = \frac{0}{50} = 0.000$$
$$c_2 = \frac{4+3+0+7}{50} = \frac{14}{50} = 0.280$$
$$c_3 = \frac{2+2+6+1}{50} = \frac{11}{50} = 0.220$$

Dado que $\bigvee_{i=1}^{3} c_i = c_2 = \frac{14}{50} =$ 0.280, se asigna la clase C_2 *Iris-versicolor*. Pero $\mathbf{y}^{135} = C_3 \neq C_2 = \tilde{\mathbf{y}}$, por lo que la clasificación es incorrecta otra vez.
Aplicando la idea de los pesos, pero ahora usando los valores 0.5, 0.5, 1, y 0.5, respectivamente, se obtienen los siguientes valores para las sumas ponderadas.
Para $\theta = 0$:

$$c_1 = \frac{(0 * 0.5) + (0 * 0.5) + (0 * 1) + (0 * 0.5)}{50} = \frac{0}{50} = 0.000$$
$$c_2 = \frac{(4 * 0.5) + (3 * 0.5) + (0 * 1) + (7 * 0.5)}{50} = \frac{7}{50} = 0.140$$
$$c_3 = \frac{(2 * 0.5) + (2 * 0.5) + (6 * 1) + (1 * 0.5)}{50} = \frac{8.5}{50} = 0.170$$

Como ahora $\bigvee_{i=1}^{3} c_i = c_3 = \frac{8.5}{50} =$ 0.170, se asigna la clase C_3 *Iris-virginica*: $\mathbf{y}^{135} = C_3$ y $C_3 = \tilde{\mathbf{y}}$, por lo que la clasificación es correcta.

Utilizando los pesos del ejemplo ej-pesos1 --- 0, 0, 1, y 0.5, respectivamente--- se obtienen los siguientes resultados.

Para $\theta = 0$:

$$c_1 = \frac{(0 * 0) + (0 * 0) + (0 * 1) + (0 * 0.5)}{50} = \frac{0}{50} = 0.000$$
$$c_2 = \frac{(4 * 0) + (3 * 0) + (0 * 1) + (7 * 0.5)}{50} = \frac{3.5}{50} = 0.070$$
$$c_3 = \frac{(2 * 0) + (2 * 0) + (6 * 1) + (1 * 0.5)}{50} = \frac{6.5}{50} = 0.130$$

En este caso, $\bigvee_{i=1}^{3} c_i = c_3 = \frac{6.5}{50} =$ 0.130, con lo que se asigna la clase C_3: $\mathbf{y}^{135} = C_3 = \tilde{\mathbf{y}}$, por lo que la clasificación también es correcta.

■

Como se ha visto en los ejemplos ej-pesos1 y ej-pesos2, es posible mejorar el desempeño del clasificador Gamma al modificar las sumas ponderadas, añadiendo no sólo la cardinalidad de cada clase, sino también un peso a cada rasgo, permitiendo premiar a las dimensiones que ayuden más a la clasificación y castigar a las que entorpezcan la decisión. Nótese que un valor de peso de 1 equivale a no modificar la suma ponderada para esa dimensión; se puede decir que el algoritmo original incluye pesos de 1 en todas las dimensiones. Por otro lado, un peso igual a 0 tiene el mismo efecto que ignorar esa dimensión; esta es una situación un tanto radical, por lo que se recomienda procurar no llegar a estos extremos. Finalmente, no existen límites estrictos a los valores que pueden tomar los pesos, aunque asignar valores negativos puede ser contraproducente: ahora no sólo se estaría dando menor peso a la información aportada por dicha dimensión, sino que esa información se le restaría al valor de la suma ponderada. En cuanto al límite superior, probablemente no convenga alejarse mucho hacia un extremo solamente, por lo que se recomienda mantener los pesos dentro del rango $(0, 2]$.

Cabe mencionar que, a final de cuentas, los valores óptimos que tomen los pesos de cada rasgo dependen estrechamente del problema de clasificación y del conjunto fundamental. En principio, se sugiere asignar valores de acuerdo con los siguientes casos.

- Dentro del rango $[1.5, 2]$ a las dimensiones que sean puntualmente separables para todas las clases.
- Dentro del rango $[1, 1.5]$ a las dimensiones que sean puntualmente separables para algunas clases o bien que sean puntualmente segmentables para todas las clases.
- Dentro del rango $[0.8, 1.2]$ a las dimensiones que sean puntualmente segmentables para todas o algunas clases.
- Dentro del rango $(0, 0.5]$ a las dimensiones que sean puntualmente no separables.

Eficiencia de Clasificación

Utilizando los hallazgos encontrados acerca de las fronteras entre clases, es posible analizar qué sucede con los patrones que son clasificados ambiguamente. Así, se observa que los patrones que introducen mayores cantidades de ambigüedad a la clasificación son de dos tipos:

- Aquellos patrones que tienen valores muy cercanos o coincidentes con las fronteras entre clases.
- Aquellos patrones que tienen valores muy diferentes de los valores de los patrones fundamentales.

Para los patrones que se encuentran en el primer caso, la solución es iterar con valores mayores de θ, con la esperanza de encontrar información suficiente para desambiguar en patrones que se encuentren un poco más alejados. Dependiendo de qué tan densamente ---y equilibradamente--- esté poblada la región cercana a la frontera, así como qué tan ambiguos o no sean los valores del patrón en otras dimensiones, es posible que se logre la desambiguación con pocas iteraciones (valores pequeños de θ) o que sea necesario iterar durante una cantidad considerable de ciclos.

Aunque en términos de clasificación es preferible una búsqueda más amplia de valores que permitan desambiguar, en términos de eficiencia conviene limitar la profundidad de la búsqueda a valores razonables. La idea es que si se llega a dichos límites y aún existe ambigüedad entre varias clases, sea posible asignar al patrón cualquiera de tales clases, puesto que cualquier patrón capaz de desambiguar ya sea suficientemente diferente para no ser relevante. Estos límites están dados por el parámetro de paro ρ.

En el algoritmo original (ver AlgClasPropuestoGen), se define el parámetro de paro como

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

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

$$\rho = \bigwedge_{j=1}^{n} \left(\bigvee_{i=1}^{p} x_j^i \right)$$

Esta definición se basa en la asunción de que para desambiguar, basta con buscar información hasta una distancia igual al menor de los mayores valores en cada dimensión;

pero no es necesario ir más allá, puesto que no vale la pena. Sin embargo, para que esta asunción tenga sentido, es necesario que la ditribución de los valores en cada dimensión sea similar. Por ejemplo, supongamos un problema de dos dimensiones, como el siguiente.

Example *Sea un conjunto fundamental* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1, 2, \ldots, p\}$ *integrado por patrones* $\mathbf{x}^\mu$ *bidimensionales* $\forall \mu \ \mathbf{x}^\mu \in \mathbb{R}^2$ *asociados a una de dos clases posibles* $\forall \mu \ \mathbf{y}^\mu \in \{C_1, C_2\}$ *. Los valores máximos y mínimos de cada dimensión son los siguientes:*

$$\bigwedge_{i=1}^{p} x_1^i = 0.186 \quad \bigvee_{i=1}^{p} x_1^i = 0.395$$

$$\bigwedge_{i=1}^{p} x_2^i = 32.601 \quad \bigvee_{i=1}^{p} x_2^i = 85.715$$

Clasificar el patrón $\tilde{\mathbf{x}} = \begin{pmatrix} 0.923 \\ 29.891 \end{pmatrix}$ *, considerando que* $\rho = \bigwedge_{j=1}^{n} \left(\bigvee_{i=1}^{p} x_j^i \right)$.

Es claro que existe una diferencia de magnitudes entre las dos dimensiones bastante considerable, así que al presentarse $\tilde{\mathbf{x}} = \begin{pmatrix} 0.923 \\ 29.891 \end{pmatrix}$ el algoritmo se detendrá al llegar a comparar valores en los rangos $[0.528, 1.318]$ para la primera dimensión y $[29.496, 30.286]$ para la segunda dimensión. Es claro que no habrá patrones similares, puesto que los valores que el patrón aceptaría como similares se encuentran fuera de los rangos de valores de los patrones fundamentales. Así, las sumas ponderadas serán igual a 0 para todas las clases:

$$c_i = 0 \ \forall i$$

Esto es un resultado totalmente ambiguo: ¡No se encontró información con la cual tomar una decisión!

■

Otra posibilidad consiste en calcular ρ como el máximo en lugar del mínimo:

$$\rho = \bigvee_{j=1}^{n} \left(\bigvee_{i=1}^{p} x_j^i \right)$$

El riesgo de hacer esto es que una de las dimensiones sea desproporcionadamente mayor a las otras, de tal manera que al presentarse un patrón extremadamente ambiguo ---por ejemplo, uno que coincida con la frontera en varias dimensiones--- el algoritmo continúe iterando más allá de los rangos de valores en los que tiene sentido buscar. En este caso, también se terminaría con una situación de paro con ambigüedad entre las clases, pero se tardaría más en llegar a dicha conclusión.

Una tercera opción, que pretende ser un punto medio entre las dos mencionadas

anteriormente, es calcular ρ como la media entre los $e_m(j)$:

$$\rho = \frac{1}{n}\sum_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$$

A final de cuentas, un buen valor para ρ depende de las características del problema de clasificación a tratar y del conjunto fundamental.

Por otro lado, el caso de los patrones que son muy diferentes de los fundamentales es bastante interesante.
Supongamos que estamos clasificando pescados de salmón y robalo en la banda transportadora de una planta empacadora (ejemplo clásico presentado en [PR-patclas18]); ¿cómo deberíamos clasificar un pulpo? ¿o una langosta? ¿o una tortuga? ¿o un tiburón? Es claro que un pulpo, una langosta o una tortuga son muy diferentes de un salmón y a la vez de un robalo; tanto que seguramente presentarían valores muy alejados de los rangos de los patrones conocidos en prácticamente cualquier dimensión que utilicemos. El caso del tiburón es un poco diferente: a pesar de ser también totalmente diferente de las dos clases de interés, al menos se parece un poco en la forma del cuerpo (aunque también en tamaños diferentes).
Para el problema de clasificación, ¿tendrá algún sentido forzar la decisión por alguna de las dos clases? Lo más probable es que no tenga sentido, sobre todo si la finalidad del problema de clasificación es decidir cómo se va a empacar el pescado para su venta: ya sea como salmón o como robalo.
Entonces, ¿qué se puede hacer con patrones que son tan diferentes? En una aplicación como la del ejemplo, un observador externo probablemente retiraría estos patrones excepcionales. En términos del problema de clasificación, eso sería como asignar una clase nula a dichos patrones: la clase *desconocida* C_0 .
Una manera en la que se puede determinar si un patrón es totalmente desconocido es utilizando un parámetro similar al de paro (llamémosle *parámetro de pausa* ρ_0): si al iterar ese número de veces no hay suficiente similitud con patrones conocidos, entonces se considerará al patrón como perteneciente a la clase desconocida. Para visualizar lo anterior, veamos un ejemplo.

Example *Sea un conjunto fundamental* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu)|\mu = 1, 2, \ldots, p\}$ *integrado por patrones* $\mathbf{x}^\mu$ *bidimensionales* $\forall\mu\ \mathbf{x}^\mu \in \mathbb{R}^2$ *asociados a una de dos clases posibles* $\forall\mu\ \mathbf{y}^\mu \in \{C_1, C_2\}$ *. Los valores máximos y mínimos de cada dimensión son los siguientes:*

$$\bigwedge_{i=1}^{p} x_1^i = 0.186 \quad \bigvee_{i=1}^{p} x_1^i = 1.395$$

$$\bigwedge_{i=1}^{p} x_2^i = 3.601 \quad \bigvee_{i=1}^{p} x_2^i = 8.715$$

Clasificar el patrón $\tilde{\mathbf{x}} = \begin{pmatrix} 4.923 \\ 29.891 \end{pmatrix}$ *, considerando que el parámetro de paro* $\rho = \bigvee_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$ *y el parámetro de pausa* $\rho_0 = \bigwedge_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$.

Con la nueva definición de ρ, es seguro que el patrón será asignado a alguna de las clases (idealmente sin ambigüedad), ya que los rangos de búsqueda de cada dimensión serán $\left[-3.792, 13.638\right]$ y $\left[21.176, 38.606\right]$, respectivamente: el rango de la dimensión 1 incluye el rango de valores de los patrones fundamentales para esa dimensión. Pero, ¿tendría sentido buscar patrones tan diferentes del patrón a clasificar?

Cuando se cumpla que $\theta = \rho_0$, se habrán cubierto los rangos $\left[3.528, 6.318\right]$ y $\left[28.496, 31.286\right]$, respectivamente. Dado que estos rangos no se intersectan con los rangos de valores de los patrones fundamentales, las sumas ponderadas en ese momento serán

$$c_1 = \frac{0}{k_1} = 0$$
$$c_2 = \frac{0}{k_2} = 0$$
$$c_3 = \frac{0}{k_3} = 0$$

por lo que no habrá un máximo único; incluso, no habrá un solo patrón fundamental similar. Así que podemos detenernos aquí y decir que el patrón pertenece a la clase desconocida: $\tilde{\mathbf{y}} = C_0$.

■

Existen todavía dos puntos que cabe aclarar. Primero, la manera de calcular el parámetro de pausa depende en gran medida del problema a tratar, así de cómo se relacionen los valores de los patrones fundamentales de una dimensión a otra. Si todas las dimensión tienen rangos de valores similares, es probable que unos valores buenos para ρ y ρ_0 sean los siguientes:

$$\rho = \bigvee_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right) \quad \text{y} \quad \rho_0 = \bigwedge_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$$

Sin embargo, puede ser que haya dimensiones con valores muy diferentes, como sucedió en el ejemplo ej-desconocidos1. En estos casos, es necesario tener cuidado al momento de definir los valores de ρ y ρ_0. Incluso, puede suceder que hasta patrones muy diferentes a los fundamentales deban ser asociados a alguna de las clases conocidas; entonces se deberá ignorar el parámetro de pausa, lo que se puede hacer dándole un valor mayor que el del parámetro de paro: $\rho < \rho_0$.

Segundo, es necesario determinar qué tanto se permite que haya similitud entre el patrón a clasificar $\tilde{\mathbf{x}}$ y los patrones fundamentales $\mathbf{x}^{\mu}$ y aún así asignar el patrón $\tilde{\mathbf{x}}$ a la clase

desconocida C_0 . Una primera opción es exigir que no haya patrones similares; esto es, las sumas ponderadas son todas iguales a 0 : $c_i = 0 \ \forall i$. Sin embargo, recordemos que los valores extremos en todas las dimensiones suelen pertenecer a patrones no muy representativos. De lo anterior surge una segunda opción: establecer un umbral ---que puede depender de la cantidad de componentes de patrones fundamentales similares, porcentaje de componentes de patrones fundamentales similares, o bien del valor de la suma ponderada máxima--- que delimite qué tan similar debe ser $\tilde{\mathbf{x}}$ con respecto de los $\mathbf{x}^{\mu}$ para asignarle una clase conocida.

Factor de Olvido

En uno de los libros de texto más influyentes en el área de Clasificación de Patrones en todo el mundo, Duda y Hart explican que:

el objetivo central de diseñar un clasificador es sugerir acciones al presentársele patrones nuevos [...]. Este es el aspecto de la generalización. ([PR-patclas18], pp. 20).

Esta afirmación refleja un punto de vista generalizado en la comunidad científica: es preferible contar con cierta cantidad de error en la clasificación del cunjunto fundamental, con tal de obtener una mejor generalización, lo que nos lleve a un mejor desempeño en la clasificación de patrones desconocidos: los patrones de prueba.

Esta predilección del desempeño en el conjunto de prueba sobre el del conjunto fundamental ha llevado al fenómeno que se presenta cuando un clasificador es incapaz de clasificar correctamente un patrón fundamental... ¡a pesar de haber sido entrenado previamente con dicho patrón! A esta característica de los clasificadores de olvidar patrones fundamentales se le conoce como *factor de olvido* y suele variar su impacto de un algoritmo clasificador a otro.

Dado que esta característica de los algoritmos es negativa e indeseable, pero es mucho más deseable y de mayor interés el desempeño exhibido frente a patrones desconocidos, el factor de olvido no suele discutirse mucho. Otra razón que hace de este factor una variable de comparación poco popular es que prácticamente todos los algoritmos clasificadores de patrones lo presentan, en menor o mayor medida.

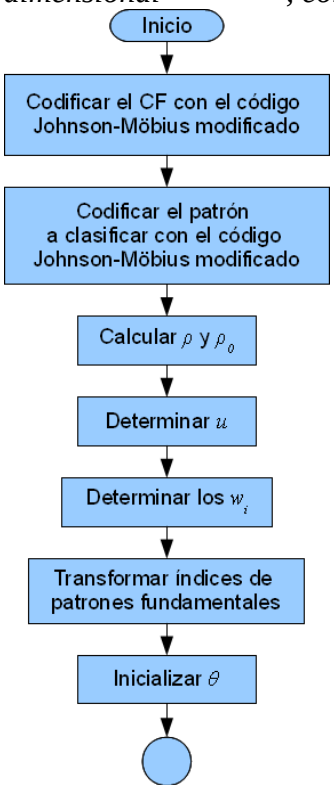
Sin embargo, el tener un desempeño competitivo ante el conjunto de prueba no necesariamente obliga a que un clasificador dado presente factor de olvido. Lo anterior se hace patente al observar que todos los modelos y algoritmos de reconocimiento de patrones pertenecientes al Enfoque Asociativo (conocidos actualmente) son capaces de recuperar o clasificar correctamente todos y cada uno de los patrones fundamentales, sin condición alguna, siempre y cuando se cumplan las asunciones iniciales con respecto al conjunto fundamental. Dicho de otra manera: los modelos asociativos de reconocimiento de patrones no presentan factor de olvido.

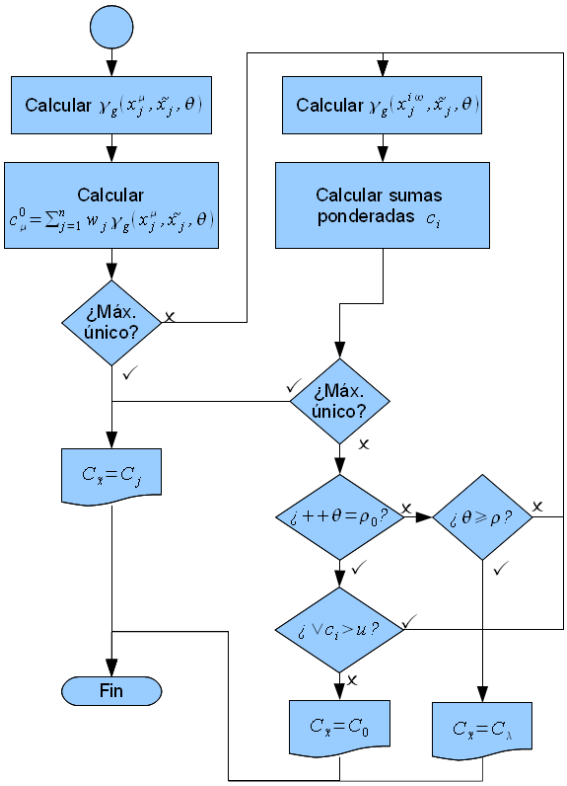
La única excepción es el clasificador Gamma original, tal y como se describe en AlgClasPropuestoGen. En efecto, como se explica en la discusión del lema lem-RecPatCon, un patrón conocido aportará n a la suma ponderada de su clase, siendo el único patrón que aporte esa cantidad, además de que si se cumple con la definición del conjunto fundamental, ningún otro patrón aportará tanto a la suma ponderada de su respectiva clase. Sin embargo, esto no garantiza al sumar las aportaciones de todos los patrones de todas las clases, las aportaciones más pequeñas de varios patrones pertencientes a otra clase sean menores a n .

Lo anterior se puede apreciar claramente en la figura FigFront04, donde el patrón (4.9, 2.5,

4.5, 1.7) sería asignado a la clase *Iris-setosa*, a pesar de pertenecer a la clase *Iris-virginica* (siempre y cuando se tomara en cuenta sólo la dimensión Longitud del sépalo, como por ejemplo si se aplicaran los pesos 1, 0, 0 y 0, respectivamente).
Para corregir este problema, basta con agregar un paso cero al algoritmo del clasificador, justo antes de iniciar con las iteraciones. En este paso, se verificará si el patrón a clasificar $\tilde{\mathbf{x}}$ es un patrón fundamental, al calcular una variante de sumas ponderadas, esta vez por patrón en lugar de hacerlo por clase. Así, e incluyendo todas las modificaciones y extensiones introducidas en estre trabajo de tesis, el algoritmo final del clasificador Gamma propuesto queda como sigue. Las figuras AlgClasMod1 y AlgClasMod2 muestran el diagrama a bloques de dicho algoritmo, donde $\checkmark$ indica un resultado verdadero en una condicional, mientras que $\times$ indica un resultado falso.

Algorithm *Sea el cunjunto fundamental del clasificador Gamma de acuerdo con la difinición def-CF. Al presentarse un patrón a clasificar* $\tilde{\mathbf{x}}$*, donde* $\tilde{\mathbf{x}}$ *es un vector real* n*-dimensional* $\tilde{\mathbf{x}} \in \mathbb{R}^n$*, con* $n \in \mathbb{Z}^+$*, se realiza lo siguiente:*





- Codificar cada componente de cada patrón del conjunto fundamental con el código Johnson-Möbius modificado, restando el valor menor a todos los valores por cada componente; asimismo, se obtiene un valor $e_m = \bigvee_{i=1}^{p} x_j^i$ por cada componente. Con lo anterior se desplaza el rango de cada componente para que vaya de 0 a e_m. Así, la componente x_j^i se transforma en un vector binario de dimensión $e_m(j)$.
- 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 ($\tilde{\mathbf{x}}_\xi > e_m(\xi)$), igualar esa componente a $e_m(\xi)$ y guardar su valor anterior en la variable $mgamma_\xi$. Por otro lado, si alguna componente da un valor negativo una vez desplazada, igualar esa componente a 0 y asignar el valor $e_m(\xi) + |\tilde{\mathbf{x}}_\xi|$ a $mgamma_\xi$.
- Calcular el parámetro de paro ρ y el parámetro de pausa ρ_0. Dependiendo del problema a tratar, algunas posibilidades sugeridas para estos parámetros son las siguientes:
  - $\rho = \bigwedge_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$.
  - $\rho = \frac{1}{n}\sum_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$.

- $\rho = \bigvee_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$.
- $\rho_0 = \bigwedge_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$, sobre todo si $\rho = \bigvee_{j=1}^{n}\left(\bigvee_{i=1}^{p} x_j^i\right)$.
- $\rho_0 > \rho$, cuando se desea asignar forzosamente una clase conocida a los patrones desconocidos.

- Determinar el umbral de pausa u. Considerando que el valor de este umbral depende fuertemente de las caracteristicas del problema y las propiedades del conjunto fundamental, se ofrecen las siguientes sugerencias como valores iniciales:
  - $u = 0$.
  - $u = n$.
- Determinar los pesos de cada dimension $w_i \in \mathbb{R}^+ | i = 1, 2, \ldots, n$. A falta de afinar estos pesos a las características del problema y las propiedades del conjunto fundamental, se sugieren los siguientes rangos como valores iniciales empiricos:
  - Dentro del rango $[1.5, 2]$ a las dimensiones que sean puntualmente separables para todas las clases.
  - Dentro del rango $[1, 1.5]$ a las dimensiones que sean puntualmente separables para algunas clases o bien que sean puntualmente segmentables para todas las clases.
  - Dentro del rango $[0.8, 1.2]$ a las dimensiones que sean puntualmente segmentables para todas o algunas clases.
  - Dentro del rango $(0, 0.5]$ a las dimensiones que sean puntualmente no separables.
- 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 ω). 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.
- Inicializar θ a 0.
- Realizar la operación $\gamma_g\left(x_j^\mu, \tilde{x}_j, \theta\right)$ para cada componente de cada uno de los patrones fundamentales y del patrón a clasificar, considerándose $mgamma_\xi$ como la dimensión del patrón binario $\tilde{\mathbf{x}}_\xi$.
- Calcular la suma ponderada inicial c_i^0 de los resultados obtenidos en el paso 7, para cada patrón fundamental $\mu = 1, 2, \ldots, p$:

$$c_\mu^0 = \sum_{j=1}^{n} w_j \cdot \gamma_g\left(x_j^\mu, \tilde{x}_j, \theta\right)$$

-

- Si existe un máximo único, cuyo valor es además igual a n , asignar al patrón a clasificar la clase correspondiente a ese máximo:

$$C_{\tilde{\mathbf{x}}} = C_j \text{ tal que } \bigvee_{\mu=1}^{p} c_\mu^0 = c_j^0 = n$$

- De lo contrario, continuar.
- Realizar la operación $\gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right)$ 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 $mgamma_\xi$ como la dimensión del patrón binario $\tilde{\mathbf{x}}_\xi$.
- Calcular la suma ponderada c_i de los resultados obtenidos en el paso 11, para cada clase $i = 1, 2, \ldots, m$:

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

- Si existe más de un máximo entre las sumas ponderadas por clase, incrementar θ en 1 y repetir los pasos 11 y 12 hasta que exista un máximo único; o se cumpla con la condición de pausa: $\theta = \rho_0$; o se cumpla con la condición de paro: $\theta \geq \rho$.
- Si se cumple con la condición de pausa $\theta = \rho_0$, se compara el valor máximo de las sumas ponderadas con el umbral de pausa.
  - Si $\bigvee_{i=1}^{m} c_i \leq u$ entonces se asigna la clase desconocida al patrón a clasificar:
    - $C_{\tilde{\mathbf{x}}} = C_0$
  - Si $\bigvee_{i=1}^{m} c_i > u$ entonces se continua en el paso 11.
- Si existe un máximo único, asignar al patrón a clasificar la clase correspondiente a ese máximo:

$$C_{\tilde{\mathbf{x}}} = C_j \text{ tal que } \bigvee_{i=1}^{m} c_i = c_j$$

En caso contrario: si λ es el índice más pequeño de clase que corresponde a uno de los máximos, asignar al patrón a clasificar la clase C_λ .

■

Resultados Experimentales y Discusión

En el presente capítulo se analizan algunos de los experimentos realizados con el clasificador Gamma, tanto en la etapa de caracterización, como en la etapa de pruebas del modelo propuesto.

Los experimentos realizados hasta la fecha incluyen tres áreas en particular. Primero tenemos el problema de la predicción de datos ambientales; por otro lado está la estimación de esfuerzo en el desarrollo de software; y finalmente tenemos la estimación de propiedades del concreto.

Predicción de Datos Ambientales

Estos experimentos se han realizado con datos tomados de las bases de datos del Sistema de Monitoreo Atmosférico de la Ciudad de México (SIMAT); en particular de la Red Automática de Monitoreo Atmosférico (RAMA) [SIMAT24]. El problema consiste en predecir la concentración de un contaminante dado, tomando como base las concentraciones registradas en una estación en particular a lo largo de un año. Una explicación a fondo y los resultados obtenidos están publicados en [SIMAT1], [SIMAT-cerma] y [SIMAT-INTECH].

Para esta aplicación se conformó el conjunto fundamental con los datos tomados en una estación durante un año, mientras que el conjunto de prueba está formado por los datos tomados en la misma estación durante un mes posterior al periodo de tiempo que corresponde al conjunto fundamental. El mes elegido ha sido no contiguo al año que comprende el conjunto fundamental, principalmente por decisión propia (arbitraria). A continuación se muestran las figuras Fig1 y Fig2 (tomadas tal cual de [SIMAT1]), en donde se comparan los resultados de la predicción hecha con el clasificador Gamma contra los datos reales del mes de febrero de 2002. La figura Fig1 muestra los resultados de concentración de SO_2, mientras que la figura Fig2 muestra los resultados para el valor del IMECA. Nótese que en ambos casos, las gráficas tanto de los datos predichos como de los datos observados son muy similares.

En las tablas 5.1 y 5.2 (tomadas de [SIMAT1]) se presenta una comparación con respecto a trabajos relacionados. En el caso de la tabla 5.1 se compara con resultados experimentales obtenidos con datos de la misma base de datos del SIMAT, mientras que la tabla 5.2 contiene resultados obtenidos con datos tomados de diversas bases de datos. En ambos casos, se utiliza el *Rooted Mean Square Error* (RMSE, raíz del error cuadrático medio en inglés) como medida de desempeño.

Remark *El RMSE se calcula por medio de la siguiente expresión.*

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - O_i)^2}$$

Cabe aclarar que no es posible realizar una comparación directa, ya que los datos con los cuales se han hecho los experimentos son diferentes. Incluso, en el caso de la tabla 5.2, hay resultados de experimentos hechos con datos de contaminantes distintos a SO_2, y algunos

además expresados en unidades diferentes. Nótese que a pesar de lo anterior, los resultados obtenidos por el clasificador Gamma son bastante competitivos.

Tabla 5.1. Comparación de resultados relacionados (base de datos SIMAT) en RMSE, dados para valor de IMECA / nivel de IMECA; ND indica un valor no disponible

Algoritmo Usado	Contaminantes Considerados	Tamaño del Conjunto Fundamental / de Prueba	Desempeño (RMSE)
Red bayesiana cite: 1997	O_3 (ppm)	400 / 200	26.8 / 10
Red neuronal cite: 1997	O_3 (ppm)	400 / 200	19.4 / NA
C4.5 cite: 1997	O_3 (ppm)	400 / 200	21.4 / NA
Clasificador Gamma	SO_2 (ppm)	8040 / 672	7.09 / 0.12

Tabla 5.2. Comparación de resultados relacionados (bases de datos diversas) en RMSE, dados para concentración; ND indica un valor no disponible, ppb denota partes por billón

Algoritmo Used	Contaminantes Considerados	Tamaño del Conjunto Fundamental / de Prueba	Desempeño (RMSE)
Red neuronal cite: ex07	O_3 (μg/m^3)	613 / 105	15
Red neuronal cite: ex06	O_3 (ppb)	NA / 1343	9.43
		NA /2367	13.79
Online SVM cite: 2008	SO_2 (μg/m^3)	240 / 168	12.96, 10.90
CALINE3 cite: ex01	PM_{10}, $PM_{2.5}$ (μg/m^3)	~120	88, 55
Clasificador Gamma	SO_2 (ppm)	8040 / 672	0.009218

Estimación de Esfuerzo en el Desarrollo de *Software*

Estos resultados se encuentran actualmente en proceso de publicación. Sin embargo, están muy relacionados con lo presentado en [nuevasCuau], pues ahí se publicó la base de datos utilizada. El problema consiste en estimar el tiempo que se tarda un programador en desarrollar una pieza de software (un programa pequeño), dadas la cantidad de líneas de código nuevas y modificadas, así como la cantidad de líneas reutilizadas. Cabe mencionar que este problema resultó ser complicado para al calsificador Gamma, puesto que el conjunto fundamental presenta irregularidades: varios patrones tienen asociadas más de un clase. A pesar de lo anterior, los resultados obtenidos son bastante competitivos, como se puede apreciar en la tabla 5.3, que es una adaptación del material que está en proceso de publicación.

Tabla 5.3. Comparación de resultados en MMER

	Ecuación de Regresión Lineal Múltiple	Modelo Basado en Lógica Difusa	Clasificador Gamma
Verificación	0.27	0.25	0.22
Validación	0.28	0.28	0.28

A continuación se presenta la manera de calcular la *Mean* MER (MMER), que es la métrica utilizada para comparar los resultados experimentales en este problema, tal como se puede ver en [nuevasCuau].

$$MER_i = \frac{\left|\text{Esfuerzo real}_i - \text{Esfuerzo predicho}_i\right|}{\text{Esfuerzo predicho}_i}$$

$$MMER = \frac{1}{N}\sum_{1}^{N} MER_i$$

Estimación de Propiedades del Concreto

Los experimentos descritos en esta sección se llevaron a cabo utilizando una base de datos tomada del *Machine Learning Repository* de la UCI [DB1]: el *Concrete Compressive Strength Data Set* [DB6]. Esta base de datos, conformada por 1030 muestras y 9 variables, describe la resistencia a la compresión (*compressive strength*) de varias muestras de concreto u hormigón, dadas la composición y edad de mezclada cada muestra. La relevancia de esta base de datos, según su descripción en [DB1], radica en que el concreto es el material más importante en ingeniería civil. En la tabla 5.4 se describen las variables que integran el *Concrete Compressive Strength Data Set*. Cabe mencionar que no hay datos perdidos: todas las instancias cuentan con todos sus valores para todas las variables.

Tabla 5.4. Descripción de la base de datos *Compressive Strength Data Set*

Variable	Valores	Rango	Unidad
Cemento	Reales	102-540	kg/m^3
Escorias de altos hornos	Reales	0-359.4	kg/m^3
Ceniza suelta	Reales	0-200.1	kg/m^3
Agua	Reales	121.75-247	kg/m^3
Superplastificante	Reales	0-32.2	kg/m^3
Árido grueso	Reales	801-1145	kg/m^3
Árido fino	Reales	594-992.6	kg/m^3
Edad	Enteros	1-365	días
Resistencia a la compresión	Reales	2.331-82.599	MPa

Como se puede apreciar, 8 de las 9 variables toman valores reales, mientras que la variable Edad toma valores enteros: se refiere a los días que tiene de hecha la mezcla de la cual se tomó la muestra. Por otro lado, todos los componentes de la mezcla se miden en kilogramos por metro cúbico (kg/m^3), mientras que la edad se mide en días y la resistencia a la compresión se mide en mega Pascales (MPa).

Para los experimentos, se plantearon dos situaciones. En la primera, la propuesta por el autor de la base de datos, se toman en cuenta la composición y edad de cada mezcla con el propósito de estimar la resistencia a la compresión de la misma. En la segunda propuesta, se predice la edad de la mezcla dada su composición y resistencia a la compresión. Para ambos conjuntos de experimentos, se utilizó la conocida metodología de validación *leave-one-out*.

Por otro lado, se utilizó el software WEKA (versión 3.6.1) [ex10] para llevar a cabo los experimentos sobre otros clasificadores, con el fin de comparar el desempeño de los mismos con el que muestra el clasificador Gamma. Se decidió utilizar este ambiente de experimentación dada la amplia colección de algoritmos de clasificación, *machine learning* y minería de datos con que cuenta, así como su creciente aceptación como herramienta de comparación por grupos de investigación científica en todo el mundo.

En la tabla 5.5 se muestran los resultados obtenidos por el clasificador Gamma y otros cuatro clasificadores incluidos en WEKA, usando de nuevo el RMSE como medida de comparación. En dicha tabla se ha utilizado la nomenclatura usada por WEKA, aunque tanto lazy.IBk1 como lazy.IBk1 son implementaciones de un clasificador muy famoso y utilizado, mejor conocido como k-NN: el primero corresponde al caso $k = 1$ mientras que el segundo es el caso de $k = 3$.

Tabla 5.5. Comparación de resultados con el *Concrete Compressive Strength Data Set*, estimación de la resistencia a la compresión

Modelo	Desempeño (RMSE)
lazy.IBk1	8.5512
lazy.IBk3	8.6904
Gamma	**13.3477**
trees.DecisionStump	14.5052
rules.ZeroR	16.7139

Como se puede ver, el desempeño del clasificador Gamma es competitivo, aunque puede mejorar bastante. A pesar de ser superado por el k -NN, es factible pensar que al afinar los parámetros del clasificador Gamma (en particular los pesos de los rasgos) se puedan obtener mejores resultados.
Por su parte, la tabla 5.6 presenta los resultados experimentales obtenidos al estimar la edad de las mezclas. En este caso, se utilizó una medida de comparación además del RMSE: el porcentaje de clasificación correcta. Dependiendo de cómo se plantee el problema, esta medida puede ser más o menos importante. Dado que la variable edad está dada en días (valores enteros), se puede atacar este problema como una tarea de clasificación.

Tabla 5.6. Comparación de resultados con el *Concrete Compressive Strength Data Set*, estimación de la edad

Modelo	Desempeño	
	(% Clasificación)	(RMSE)
Gamma	**49.32**	**44.56**
lazy.IBk1	44.66	42.01
lazy.IBk3	28.35	46.69
rules.ZeroR	0.00	63.20
trees.DecisionStump	0.00	56.60
trees.M5P	0.00	44.07

En cuanto al RMSE, el clasificador Gamma es superado por el 1 -NN y, marginalmente, por el M5P, que es un clasificador basado en árboles de decisión; mientras que deja atrás al 3 -NN y más aun a los otros clasificadores. Sin embargo, el porcentaje de clasificación

correcta presenta un panorama muy diferente: en primer lugar tenemos al clasificador Gamma con 49%, seguido de cerca por el k-NN (44% el 1-NN y 28% el 3-NN), mientras que los otros modelos tienen 0% de clasificación correcta. Esta diferencia en el desempeño relativo del clasificador Gamma y su superioridad en la tarea de clasificación sobre la de aproximación de funciones se debe, muy probablemente, a que la primera tarea es para la que fue diseñado, mientras la segunda no.

Cabe mencionar que no todos los clasificadores incluidos en WEKA se pueden utilizar siempre. Por ejemplo, el modelo trees.J48, que es una implementación del famosísimo clasificador C4.5, no es capaz de tratar valores numéricos para la clase, por lo que no puede procesar esta base de datos. Esta es una limitación que no presenta el clasificador Gamma.

Conclusiones, Contribuciones y Trabajo Futuro

En el presente capítulo se enumeran las conclusiones obtenidas durante el desarrollo de este trabajo de tesis, así como las aportaciones derivadas del mismo. Después, se exploran algunos de los posibles trabajos futuros que se desprenden de este trabajo, mismos que pueden ser aprovechados por otros investigadores para continuar ampliando el área de investigación en Reconocimiento de Patrones.

Conclusiones

- Se ha caracterizado la operación del clasificador Gamma.
  - En particular, se observó que los conjuntos fundamentales que inducen relaciones que no son funciones, ocasionan problemas.
  - Por lo anterior, se define el conjunto fundamental del clasificador Gamma como uno que induce una función; esto es, un patrón fundamental pertenece a sólo una clase y los patrones fundamentales no se repiten.
- También se establece un teorema que garantiza condiciones suficientes para clasificación correcta del conjunto fundamental completo.
- Por otro lado, se ha caracterizado cómo el clasificador Gamma induce las fronteras entre clases a partir del conjunto fundamental, identificándose patrones que causan error o ambigüedad en la clasificación, lo que a su vez se traduce en clasificación incorrecta o ineficiente, respectivamente.
  - Con la intención de subsanar tales problemas, se proponen dos adecuaciones al clasificador: asignar pesos a las dimensiones para premiar aquellos rasgos que facilitan la clasificación y castigar aquellos que introducen ambigüedad o error; y considerar una clase desconocida para aquellos patrones que sean demasiado diferentes de los patrones fundamentales.
- Asimismo, se introduce un paso inicial en el algoritmo de clasificación que otorga al clasificador Gamma un factor de olvido nulo.
- Se ha aplicado el clasificador Gamma en diversas bases de datos, inmersas en diferentes dominios del quehacer humano.
  - Por un lado, la base de datos RAMA del SIMAT, dedicada al monitoreo de la contaminación atmosférica en la Ciudad de México.
  - Por otro lado, se usó una base de datos recopilada ex profeso, misma que está inmersa en el problema de la estimación del esfuerzo en el desarrollo de programas pequeños; en estos dos experimentos se aplicó la tarea de aproximación de funciones.
  - Por último, se utilizó el *Concrete Compressive Strength Data Set* del *Machine Learning Repository* de la UCI (uno de los repositorios de bases de datos más prestigiosos y utilizados en las áreas de reconocimiento de patrones y *machine learning*); los problemas atacados con esta base de datos fueron, por un lado, la predicción de la resistencia a la compresión de diversas mezclas de concreto y, por el otro, la estimación de la edad de dichas mezclas. Con esta última base de datos se aplicaron las tareas de clasificación de patrones y de aproximación de funciones.
- Los resultados experimentales fueron sumamente alentadores, puesto que en todos ellos

el clasificador Gamma exhibe un desempeño competitivo.

- Con todos los experimentos realizados, se ha ganado un bagaje de conocimeinto y experiencia que se resumen en las extensiones hechas al clasificador original, permitiendo mejorar su desempeño tanto en cuanto a eficacia como en lo tocante a eficiencia.

Contribuciones

- Un teorema que garantiza condiciones suficientes para clasificación correcta del conjunto fundamental completo.
- Caracterización de cómo el clasificador Gamma induce las fronteras entre clases.
- Modificación al algoritmo del clasificador Gamma:
  - Se asignaron pesos a las dimensiones.
  - Se añade una clase desconocida.
  - Se introduce un paso inicial que otorga al clasificador Gamma un factor de olvido nulo.

Trabajo Futuro

- Algunos problemas presentan conjuntos fundamentales que necesitan un análisis más profundo que el aquí mostrado para poder afinar los parámetros del mismo: en particular los pesos asignados a los rasgos.
  - Por ejemplo, el *Concrete Compressive Strength Data Set*, que no permite diferenciar fácilmente un rasgo de otro: todos son puntualmente no separables. Entonces, para poder asignarles pesos que mejoren el desempeño del clasificador, es necesario desarrollar (o adoptar) una metodología de aprendizaje de pesos; tal vez similar al algoritmo *back propagation* utilizado en redes neuronales.
- Asimismo, los valores sugeridos para calcular ρ, ρ_0 y u son resultados empíricos, decantados de la experiencia acumulada durante la aplicación y experimentación con el clasificador Gamma propuesto. Esto no garantiza que funcionen tal cual para todos los problemas; antes al contrario, lo más probable es que no ofrezcan los mejores resultados posibles, por lo que se sugieren únicamente como puntos de partida.
  - Un trabajo futuro consiste en desarrollar métodos específicos para determinar los valores de estos parámetros, dado un problema.
- Por otro lado, queda abierto el camino para utilizar el clasificador Gamma con muchas otras bases de datos conocidas y problemas de aplicación.

Referencias

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)

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

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

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 (2006) 151-160. ISBN: 0-7695-2722

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

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

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

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)

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

Sarünas, R.: Statistical and Neural Classifiers, An integrated Approach to design. MIT Press, England (2001)

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

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

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

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

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

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

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)

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

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

Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NNpattern Classification Techniques. IEEE Computer Society Press, USA (1991)
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
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
Huang, Y.S., Chiang, C.C., Shieh, J. W., Grimson, E.: Prototype optimization for nearest-neighbor classification. Pattern Recognition, 35 (2002) 1237-1245
Wu, Y., Ianekiev, K., Govindaraju, V.: Improved k-nearest neighbor classification. Pattern Recognition, vol. 35 (2002) 2311-2318
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)
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) 191-202. ISSN 1665-9899
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) 304-315. ISBN: 970-36-0267-3
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) 316-326. ISBN: 970-36-0267-3
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
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)
Gonzalez, R.C., Thomason, M.G.: Syntactic Pattern Recognition: an Introduction. Addison Wesley (1982)
McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, vol. 5 (1943) 115-133
Rosenblatt, F.: The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, vol. 65 (1958) 386-408
Pal, S.: Neuro-Fuzzy, Pattern Recognition: Methods in Soft Computing. USA, John Wiley & Sons (1999)
Pandya, A.S.: Pattern recognition with neural networks in C++. Springer-Verlag, Great Britain (1996)
Abe, S.: Pattern classification, Neuro-Fuzzy Methods and their Comparison. Springer-Verlag, Grat Britain (2001)
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
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

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
Anderson, J.A., Rosenfeld, E(eds.): Neurocomputing: Fundations of Research. MIT Press, Cambridge (1990)
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
Haykin, S.: Neural Networks, A Comprehensive Foundation. Prentice Hall, USA (1999)
Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge (1995)
Kishan, M., Chilukuri, K.M., Sanjay, R.: Elements of Artificial Neural Networks. MIT Press, USA (1997)
Minsky, M., Papert, S.: Perceptrons. MIT Press, Cambridge (1969)
Rumelhart, D.E., Hinton, G. E., Williams, R.J.: Learning internal representation by Backpropagating errors. Nature, 323 (1986) 533--536
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
Krauth, W., Mezard, M.: Learning algorithms with optimal stability in neural networks. J. Phys. A: Math. Gen, vol. 20 (1987) L745-L752
F. Rosenblatt.: Principles of Neurodynamics. Spartam Books, New York (1992)
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
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, USA (2005) 247-254. ISBN: 0-8194-5921-6, ISSN: 0277-786X, DOI: 10.1117/12.621783
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)
Yáñez-Márquez, C.: Memorias Asociativas Basadas en Relaciones de Orden y Operadores Binarios. Tesis doctoral, Centro de Investigación en Computación, IPN, México (2002)
Acevedo-Mosqueda, M.E.: Memorias Asociativas Bidireccionales Alfa-Beta. Tesis doctoral, Centro de Investigación en Computación, IPN, México (2006)
Acevedo-Mosqueda, M.E., Yáñez-Márquez, C., López-Yáñez, I.: Complexity of Alpha-Beta Bidirectional Associative Memories. Lecture Notes in Computer Science, LNCS 4293, Springer-Verlag Berlin Heidelberg (2006) 357-366. ISSN: 0302-9743
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)
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, LNCS 4225, Springer-Verlag, Berlin Heidelberg (2006) 529-538. ISSN: 0302-9743
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, LNCS 4263, Springer-Verlag Berlin Heidelberg (2006) 286-295. ISSN: 0302-9743
Sánchez Garfias, F.A., Díaz-de-León Santiago, J.L.,Yáñez Márquez, C: Lernmatrix de Steinbuch: avances teóricos, Computación y Sistemas, Vol. 7, No. 3, México (2004) 175-189. ISSN 1405-5546
Yáñez Márquez, C., Díaz-de-León Santiago, J.L.: Memorias Asociativas Basadas en Relaciones de Orden y Operaciones Binarias. Computación y Sistemas, Vol. 6, No. 4, México (2003) 300-311. ISSN 1405-5546
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) 105-110. ISSN: 0973-1873
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) 190-194. ISSN: 1738-7906
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) 11-20. ISSN 1870-4069
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) 91-102. ISSN 1665-9899
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) 55-59. ISBN: 0-7695-2708-6
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) 64-69. ISBN: 970-36-0355-6
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, USA (2005) 233-241. ISBN: 0-8194-5921-6, ISSN: 0277-786X, DOI: 10.1117/12.621551
Hassoun, M.H.(ed.): Associative Neural Memories. Oxford University Press, New York (1993)
Kohonen, T.: Self-Organization and Associative Memory. Springer-Verlag, Berlin (1989)
Sossa, H., Barrón, R., Vázquez, R.: New Associative Memories to Recall Real-Valued Patterns. CIARP 2004, LNCS 3287 (2004) 195-202
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
López Yáñez, I.: Clasificador Automático de Alto Desempeño. Tesis de Maestría de la Maestría en Ciencias de la Computación. Centro de Investigación en Computación del Instituto Politécnico Nacional, México (2007)
Román Godínez, I.: Aplicación de los modelos asociativos Alfa-Beta a la Bioinformática. Tesis de Maestría de la Maestría en Ciencias de la Computación. Centro de Investigación en Computación del Instituto Politécnico Nacional, México (2007)

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

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) 206-212

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, LNCS 3315 (2004) 646-655

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

Sossa, Humberto, Barrón, Ricardo, A. Vázquez, 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, LNCS 3315 (2004) 687-696

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) 27-1 a 27-5

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)

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. In 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

Román-Godínez, I., López-Yáñez, I., Yáñez-Márquez, C.: Perfect Recall on the Lernmatrix. In 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

Argüelles-Cruz, A.J., López-Yáñez, I., Aldape-Pérez, M., Conde-Gaxiola, N.: Alpha-Beta Weightless Neural Networks. Lecture Notes in Computer Science, LNCS 5197, Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-72394-3 (2008) 496-503

Brier, G.W. : Verification of forecasts expressed in terms of probabilities. Monthly Weather Review vol. 78 (1950) 1--3

Kononenko, I.:Estimating attributes: analysis and extensions of RELIEF. Proc. European Conference on Machine Learning 1994 (ECML 94) (1994) 171--182

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

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

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
Steinbuch, K.: Die Lernmatrix. Kybernetik, vol. 1, num. 1 (1961) 36-45
Maji, P., Chaudhuri, P.P.: RBFFCA: A hybrid pattern classifier using radial basis function and fuzzy cellular automata. Fundamenta Informaticae, Vol. 78, Iss. 3 (2007) 369-396
Chakrabartty, S., Cauwenberghs, G.: Sub-microwatt analog VLSI trainable pattern classifier. IEEE Journal of Solid-State Circuits Vol. 42, Iss. 5 (2007) 1169-1179
Craig, D.A., Nguyen, H.T., Burchey, H.A.: Two channel EEG thought pattern classifier. Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings (2006) 1291-1294
Kostin, A.: A simple and fast multi-class piecewise linear pattern classifier. Pattern Recognition Vol. 39, Iss. 11 (2006) 1949-1962
Xu, L., Guang, L., Le, W., Freeman, W.J.: A study on a bionic pattern classifier based on olfactory neural system. International Journal of Bifurcation and Chaos Vol. 16, Iss. 8 (2006) 2425-2434
González, R. C. , Woods, R. E.: Digital Image Processing.Prentice Hall, USA (2001)
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
Cortes, C.,Vapnik V.: Support Vector Network. Machine Learning, 20 (1995) 273--297
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
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
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
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
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
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
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
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--

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
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
Altinçay, H.: On Naive Bayesian Fusion of Dependent Classifiers. Pattern Recogn. Lett. 26(15). Elsevier Science Inc. (2005) 2463-2473
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
Yager, R.R.: An Extension of the Naive Bayesian Classifier. Information Sciences 176(5) 6 March 2006. Elsevier Science Inc. (2006) 577-588
StatSoft Inc. Naive Bayes Classifier (2006) Available at: http://www.statsoft.com/textbook/stnaiveb.html
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
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
Vapnik, V., Lerner, A.: Pattern recognition using generalized portrait method.Automation and Remote Control, Vol. 24, Is. 6 (1963) 774-780
Vapnik,V., Chervonenkis, A.: A note on one class of Perceptrons. Automation and Remote Control, 25 (1964)
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
Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. Knowledge Discovery and Data Mining, 2(2) (1998)
Lin, H., Venetsanopoulos, A.N.: A Weighted Minimum Distance Classifier for Pattern Recognition. Canadian Conference on Electrical and Computer Engineering (1993) 904-907
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
Rosen, K.H.: Discrete Mathematics and Its Applications. McGraw-Hill. Third Edition (1995)
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)
Asuncion, A. & Newman, D.J.: UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. (2007) Available at: http://archive.ics.uci.edu/ml/
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
Patwari, N.: Wireless Sensor Network Localization Measurement Repository. Available at: http://www.eecs.umich.edu/~hero/localize/
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
Ayaquica-Martínez, I. O.: Algoritmo C-means usando funciones de disimilaridad. Tesis de Maestría en: Ciencias de la Computación. CIC IPN, México (2002)
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
Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. IEEE Transactions on Evolutionary Computation (1997)
Wolpert, D.H., et al.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation (1997)
Sewell, M.: No Free Lunch Theorems. (2006) Available at: http://www.no-free-lunch.org/
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
Sistema de Monitoreo Atmosférico de la Ciudad de México: IMECA (2007) Disponible en: http://www.sma.df.gob.mx/simat/pnimeca.htm
Yáñez-Márquez, C., López-Yáñez, I., Sáenz Morales, G. de la L.: Analysis and Prediction of Air Quality Data with the Gamma Classifier. Lecture Notes in Computer Science, LNCS 5197, Springer-Verlag Berlin Heidelberg (2008) 651-658. ISBN: 978-3-540-72394-3
López-Yáñez, I., Yáñez-Márquez, C., Sáenz Morales, G. de la L.: Application of the Gamma Classifier to Environmental Data Prediction. Proc. Electronics, Robotics and Automotive Mechanics Conference CERMA 2008, IEEE Computer Society (2008) 80-84. ISBN: 978-0-7695-3320
López-Yáñez, I., Yáñez-Márquez, C., Silva-García, V.M.: Forecasting Air Quality Data with the Gamma Classifier, in Peng-Yeng Yin (Ed.): Pattern Recognition, ISBN: 978-953-307-014-8, INTECH (2009) 499-512, Available from: http://sciyo.com/articles/show/title/forecasting-air-quality-data-with-the-gamma-classifier
Sucar, L.E., Pérez-Brito, J., Ruiz-Suárez, J.C., Morales, E.: Learning Structure from Data and Its Application to Ozone Prediction. Applied Intelligence, vol. 7 (4) (1997) 327-338
Dutot, A.-, Rynkiewicz, J., Steiner, F.E., Rude, J.: A 24-h forecast of ozone peaks and exceedance levels using neural classifiers and weather predictions. Environmental Modelling and Software, vol. 22(9) (2007) 1261-1269
Salazar-Ruiz, E. *et al.*: Development and comparative analysis of tropospheric ozone prediction models using linear and artificial intelligence-based models in Mexicali, Baja California (Mexico) and Calexico, California (US). Environmental Modelling and Software, vol. 23 (8) (2008) 1056-1069
Wang, W., Men, C., Lu, W.: Online prediction model based on support vector machine. Neurocomputing, vol. 71 (4-6) (2008) 550-558
Gokhale, S., Raokhande, N.: Performance evaluation of air quality models for predicting PM10 and PM2.5 concentrations at urban traffic intersection during winter period. Science of the Total Environment, vol. 394 (1) (2008) 9-24

López-Martín, C., Yáñez-Márquez, Cornelio, Gutiérrez-Tornés, A.: Predictive Accuracy Comparison of Fuzzy Models for Software Development Effort of Small Programs, Journal of Systems and Software, Vol. 81, No. 6, Elsevier (2008) 949-960. ISSN: 0164-1212. DOI: 10.1016/j.jss.2007.08.027. Ms. Ref. No.: JSS-D-07-00144R1.

I-Cheng Yeh: Modeling of strength of high performance concrete using artificial neural networks. Cement and Concrete Research, Vol. 28, No. 12 (1998) 1797-1808

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten: The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1 (2009)