25
24

# UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A C A T L A N

"DESARROLLO DE UN ALGORITMO PARA EL ANALISIS DE DATOS Y CLASIFICACION TIPO TIPICIDAD Y CONTRASTE"

# T E S I S

QUE PARA OBTENER EL GRADO DE:

**LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION.**

P R E S E N T A :

**LETICIA VEGA ALVARADO**

MEXICO, D.F. MARZO DE 95


**UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso**

A mis padres:

Celia y Medardo

Con infinita gratitud y cariño por todos sus sacrificios, apoyo y confianza además de su cariño, comprensión y estímulos de superación.

A mis hermanos:

Eduardo, Alfredo y Arturo

Por todo su apoyo, amistad y cariño.

A José Ruiz Shulcloper:

Por su amistad, confianza comprensión y apoyo, asi como por su colaboración para el desarrollo de este trabajo.

Finalmente quiero expresar mi gratitud a todas y cada una de las personas e Instituciones que contribuyeron en la elaboración y presentación de este trabajo, así como, a los que formaron parte de mi desarrollo y formación profesional.

# INDICE


**Introducción** 1

**Capitulo 1 Problemas de clasificación y selección de rasgos en Reconocimiento de Patrones (R.P)**

1.1 ¿Qué es R.P. y cuáles son sus problemas fundamentales? 5

1.2 Planteamiento formal de un problema de clasificación con aprendizaje. 12

1.3 Análisis de datos y determinación de anomalías. 20

**Capitulo 2 Algoritmo de clasificación para el análisis de datos**

2.1 Conceptos básicos. 23

2.2 Descripción del algoritmo. 28

2.3 Limitantes del algoritmo. 35

**Capitulo 3 Planteamiento de un algoritmo tipo tipicidad y contraste**

3.1 Definiciones básicas. 39

3.2 Descripción del algoritmo. 44

3.3 Resultados básicos. 49

**Capitulo 4 Implantación computacional del algoritmo**

4.1 Introduccón. 51

4.2 Estructuras y funciones 56

4.3 Manejo del sistema 62

**Conclusiones** 69

**Bibliografía** 71

**Apendice** 75

# INTRODUCCIÓN

Existen muchas ciencias como la Geología, la Medicina, la Psicología y otras más en las cuales con frecuencia surgen problemas de clasificación como son la determinación de zonas perspectivas para algún mineral, el pronóstico de complicaciones posoperatorias de un paciente, el diagnóstico del estado mental de un paciente, etc. La mayoría de estos problemas presentan características en común. Una de las características más importantes es que en general estos problemas no son representables en espacios métricos ya que involucran diferentes tipos de variables; desde las que denotan la presencia o ausencia de alguna propiedad o característica, como es el caso de la fiebre; las que sólo pueden tomar k diferentes valores; hasta aquellas cuya descripción es lingüística y subjetiva, como son los casos del dolor y el olor, por ejemplo, que están en dependencia de la persona que las describe.

Otra de las características que también aparecen con frecuencia en este tipo de ciencias y más común en el caso de las geociencias, es que no siempre se tiene la información completa, es decir, que hay propiedades o factores que no han sido estudiados o determinados ya sea porque es imposible hacerlo o porque resulte muy costoso. Como por ejemplo en el caso de la prospección geológica si se desea conocer la perspectividad en una zona de un cierto mineral, se deben tomar en cuenta una serie de manifestaciones indirectas del mineral que se encuentra en la tierra como pueden ser el campo magnético, algunas manifestaciones geoquímicas, ciertas propiedades geomorfológicas, etc. Pero algunas veces estas medidas no se pueden obtener, como es el caso del campo magnético en una zona pantanosa, por ejemplo. Asimismo el tratar de proporcionar la información que caracteriza a las zonas no perspectivas para formar una muestra es casi imposible. En el caso de la psicología pudieran desconocerse algunas características de un paciente como por ejemplo, sus antecedentes psicológicos familiares.

Además existen problemas en los cuales las clases no siempre son disjuntas como por ejemplo en el caso del diagnóstico médico en el cual un paciente puede presentar una serie de síntomas y signos que correspondan a más de una enfermedad al mismo tiempo, contrario a la prospección geológica en donde se puede determinar si una zona es perspectiva o no, pero es evidente que no puede ser ambas cosas a la vez. Asimismo hay

# INTRODUCCIÓN

Existen muchas ciencias como la Geología, la Medicina, la Psicología y otras más en las cuales con frecuencia surgen problemas de clasificación como son la determinación de zonas perspectivas para algún mineral, el pronóstico de complicaciones posoperatorias de un paciente, el diagnóstico del estado mental de un paciente, etc. La mayoría de estos problemas presentan características en común. Una de las características más importantes es que en general estos problemas no son representables en espacios métricos ya que involucran diferentes tipos de variables; desde las que denotan la presencia o ausencia de alguna propiedad o característica, como es el caso de la fiebre; las que sólo pueden tomar k diferentes valores; hasta aquellas cuya descripción es lingüística y subjetiva, como son los casos del dolor y el olor, por ejemplo, que están en dependencia de la persona que las describe.

Otra de las características que también aparecen con frecuencia en este tipo de ciencias y más común en el caso de las geociencias, es que no siempre se tiene la información completa, es decir, que hay propiedades o factores que no han sido estudiados o determinados ya sea porque es imposible hacerlo o porque resulte muy costoso. Como por ejemplo en el caso de la prospección geológica si se desea conocer la perspectividad en una zona de un cierto mineral, se deben tomar en cuenta una serie de manifestaciones indirectas del mineral que se encuentra en la tierrra como pueden ser el campo magnético, algunas manifestaciones geoquímicas, ciertas propiedades geomorfológicas, etc. Pero algunas veces estas medidas no se pueden obtener, como es el caso del campo magnético en una zona pantanosa, por ejemplo. Asimismo el tratar de proporcionar la información que caracteriza a las zonas no perspectivas para formar una muestra es casi imposible. En el caso de la psicología pudieran desconocerse algunas características de un paciente como por ejemplo, sus antecedentes psicológicos familiares.

Además existen problemas en los cuales las clases no siempre son disjuntas como por ejemplo en el caso del diagnóstico médico en el cual un paciente puede presentar una serie de síntomas y signos que correspondan a más de una enfermedad al mismo tiempo, contrario a la prospección geológica en donde se puede determinar si una zona es perspectiva o no, pero es evidente que no puede ser ambas cosas a la vez. Asimismo hay

problemas donde las clases se vuelven imprecisas, de modo que no siempre un especialista puede con absoluta certeza afirmar si un objeto pertenece a una cierta clase o no, sino que valora de alguna manera cierto grado de certidumbre para su afirmación.

Por todas las características descritas anteriormente trabajar estos problemas con métodos y modelos que no se ajusten a ellas resulta imposible sin violentar los principios elementales de la modelación matemática. Aunque cabe mencionar que en general a lo largo de los años se han utilizado modelos o métodos para solucionar estos problemas, aún cuando estos no cumplen con las características de los mismos; tal es el caso de los modelos en que se considera que todas las variables toman valores en el conjunto de los números reales y a partir de ahí se incorporan al problema todas las propiedades de los números reales y se hallan normas y distancias, etc. o aquellos modelos en donde se codifica la información, es decir, que cada variable se convierte en una propiedad que se cumple o no se cumple, y entonces se hace uso de la lógica bivalente y de todas sus propiedades.

Tomando en consideración las limitantes que presentan la mayoría de los modelos que se utilizan en la solución de problemas reales nos damos cuenta que surge la necesidad de buscar, extender o incluso crear modelos que se ajusten a las características del problema y no de manera contraria como se ha venido realizando históricamente, en donde el problema se tiene que acoplar a los requerimientos de un modelo determinado. Resulta entonces evidente que la práctica nos impulsa al desarrollo de nuevos modelos que atiendan a las características de los problemas reales.

Por tanto el objetivo y la motivación de este trabajo es fundamentalmente por la necesidad de resolver de una manera más adecuada los problemas que hemos mencionado a lo largo de este epígrafe elaborando un algoritmo de análisis de datos que se ajuste de una mejor manera a las características de los problemas reales, es decir, que considere que las variables que describen a los objetos puedan ser de cualquier tipo, como por ejemplo numéricas, bivalentes e incluso difusas; que pueda trabajar con ausencia de parte de la información y que tome en cuenta que los objetos pueden pertenecer a más de una clase, es decir que las clases en que se divide nuestro problema no necesariamente tengan que ser disjuntas, permitiéndose la multiclasificación. Para el desarrollo de este algoritmo

nos apoyaremos en las ideas básicas de un algoritmo de clasificación y análisis de datos que se presenta en un trabajo previo de Diordenko L. y Vaskosvskii B. V. [ 4 ].

Por lo descrito anteriormente nos damos cuenta que podemos ser capaces de desarrollar nuevas herramientas utilizando de una manera adecuada los conocimientos que hemos adquirido a través de nuestra formación profesional. Y que la aplicación de dichas herramientas no es solamente en el área de las Matemáticas, si no , en una gran cantidad de áreas como las que se describieron al principio de ésta sección.

El presente trabajo está constituido por cuatro capítulos.

En el capítulo 1 se hace una breve descripción histórica sobre el surgimiento y desarrollo del Reconocimiento de Patrones (R.P), la formulación de los problemas fundamentales del R.P. , así como el planteamiento formal del problema de clasificación con aprendizaje.

El capítulo 2 está dedicado a la descripción del algoritmo de análisis de datos propuesto por Diordenko L. y Vaskosvskii B. V. [ 1 ] y se mencionan las limitantes que presenta dicho algoritmo.

En el capítulo 3 se expone un nuevo algoritmo de clasificación y análisis de datos basado en los conceptos básicos del algoritmo presentado en el capítulo 2 y se muestran las ventajas que presenta dicho algoritmo.

En el capítulo 4 se hace la descripción de la implantación computacional del algoritmo presentado en el capítulo 3.

Los resultados obtenidos en este trabajo fueron expuestos en el Segundo Taller Iberoamericano de "Informática y Geociencias", realizado en La Habana, Cuba del 1ro al 4 de Agosto de 1994, y fue publicado en las memorias de dicho evento,así como en las memorias del V Congreso Nacional de Matemática y Computación. que se realizó en la Universidad Central de las Villas en Santa Clara, Cuba.

# CAPITULO 1 Problemas de clasificación y selección de rasgos en Reconocimiento de Patrones (R.P.).

## 1.1 ¿Qué es R.P. y cuáles son sus problemas fundamentales?

El reconocimiento es visto como un atributo básico del ser humano, así como de otros organismos vivos. El ser humano lleva a cabo actos de reconocimiento en cada instante de su vida. Puede reconocer los objetos que están a su alrededor y moverse y actuar sobre la base de ellos. Mucha de la información que está alrededor de nosotros se manifiesta en forma de patrones, un patrón es la descripción de un objeto. El ser humano tiene un sistema de información muy sofisticado ya que posee una capacidad muy grande de reconocer patrones.

Cuando reconocemos patrones hacemos una discriminación (clasificación) de un conjunto de objetos o fenómenos. Estos pueden ser físicos o abstractos. Por lo tanto de acuerdo a su naturaleza podemos dividir el proceso de reconocimiento, desde el punto de vista del ser humano, en dos tipos. El primer tipo es el reconocimiento de objetos concretos, es decir, aquel donde se involucra cualquiera de los cinco sentidos; para determinar las características del patrón u objeto. Por ejemplo cuando reconocemos una pintura, una canción, una persona, una huella digital, el aroma de una flor, etc., utilizamos la vista, el oído o el olfato. A este tipo de reconocimiento se le denomina reconocimiento sensitivo. El segundo tipo es el reconocimiento de objetos abstractos, es decir, aquel donde no se involucran ninguno de los sentidos sino que reconocemos los objetos de una manera conceptual, como por ejemplo, el reconocer la solución de un problema matemático, el argumento de una novela, etc. A este tipo de reconocimiento se le denomina reconocimiento conceptual [ 17 ].

Cuando una persona percibe un patrón, hace una inferencia en su cerebro y asocia lo que percibió con algún concepto general o clave derivado de experiencias pasadas, lo que le permite reconocer dicho patrón. Es por eso que en general podemos decir que el reconocimiento de patrones está basado en conocimientos adquiridos en el pasado, es

decir, en la información que caracteriza a cada uno de los objetos o patrones que conocemos y que nuestro cerebro guarda de alguna manera.

Es importante mencionar que la facilidad con que los seres humanos clasifican y describen patrones no nos puede conducir a pensar que esta capacidad es fácil de automatizar, sin embargo uno de los máximos retos a lo largo de los años ha sido el lograr que una computadora sea capaz de reconocer y percibir patrones tal como lo hace el ser humano.

A principios de los años 50's con el surgimiento y auge de las computadoras digitales se dieron los primeros pasos en el estudio del reconocimiento de patrones automatizado ya que estas computadoras comenzaron a ser una herramienta potencial para el procesamiento de información. Los primeros intentos en reconocimiento de patrones fueron hechos para la toma de decisiones automática y para el desarrollo de hardware especializado para leer patrones de carácteres alfanuméricos impresos. A finales de los años 50's Rosenblatt [ 10 ] introdujo un algoritmo llamado Perceptrón el cual fue de los primeros modelos para almacenar y organizar la información de una manera similar a como lo hace el cerebro; el algoritmo trataba de simular el funcionamiento de las neuronas y con esto ser capaz de clasificar y reconocer patrones. Este trabajo fue el prototipo de docenas de trabajos que eran similares en los conceptos básicos del mismo, por lo tanto dicho trabajo es considerado como uno de los más importantes de la década de los 50's.

La década de los 60's fue la del crecimiento rápido dentro de las investigaciones del reconocimiento de patrones. Durante esta época aparecieron cientos de artículos referentes a la clasificación de patrones, algoritmos para el procesamiento de imágenes y la aplicación de técnicas de reconocimiento de patrones a problemas prácticos, como por ejemplo, el reconocimiento de carácteres, la identificación de huellas digitales, el análisis de glóbulos rojos y blancos en la sangre, etc.

Dado que los conceptos de reconocimiento de patrones han sido un factor importante para el diseño de sistemas de información computarizados el interés en esta área está creciendo rápidamente en estudios interdisciplinarios e investigaciones de diferentes campos como son: Ingeniería, Ciencias de la Computación, Química,

Medicina, las Geociencias, Psicología, Sociología, etc. Cada una de estas áreas o disciplinas de la investigación tienen una gama muy extensa de problemas que tienen que ver con el reconocimiento, la clasificación, el diagnóstico y pronóstico de objetos y fenómenos, etc. Estos problemas están presentes de hecho en toda la actividad humana en formas muy diversas. Ejemplos particulares de esto son el diagnóstico médico o técnico del estado de salud de un paciente o del funcionamiento de un equipo; el pronóstico de fenómenos naturales como los terremotos, la determinación del nivel de productividad de un yacimiento de recursos minerales a partir de ciertos datos geológicos y geofísicos, la evaluación de la eficiencia o la calidad del trabajo de gestión de un cuadro administrativo o de un producto en una fábrica, la lectura diagnóstica de bioseñales como son los electrocardiogramas y electroencefalogramas; la interpretación diferenciante de imágenes como fotos aéreas, placas de rayos X, o imágenes provenientes de equipos de tomografía, el reconocimiento de la voz o de los sonidos en general.

En un lenguaje simple podemos decir que el reconocimiento de patrones puede ser definido como la categorización de los datos de entrada de un problema dado dentro de unas clases determinadas vía la extracción de características o atributos significativos de los datos[ 17 ]. Por ejemplo en el problema de la predicción del tiempo los datos de entrada se reciben en forma de mapas de climas de los cuales se extraen las características más importantes y sobre la base de éstas se hace la predicción, en el caso del diagnóstico médico los datos de entrada son los síntomas de los pacientes con los cuales se determina las enfermedades de dichos pacientes. En el reconocimiento de carácteres los datos de entrada son recibidos en forma de señales ópticas y sobre la base de éstas se identifica el nombre de la letra.

En Reconocimiento de Patrones.aparecen diferentes tipos de problemas. El primero de estos problemas está relacionado con la representación de los datos de entrada. Uno de los pasos importantes del reconocimiento es la determinación adecuada del dominio de definición de los datos de entrada, que no necesariamente tiene que ser igual para todos, es decir, que podemos tener diferentes tipos de características (variables, rasgos o propiedades). Por ejemplo, en el caso del diagnóstico médico los síntomas pueden ser representados mediante variables booleanas, es decir, que sólo tomen dos valores, digamos presencia o ausencia de cierta propiedad, v. gr. el vómito; reales, como es el caso del peso y la estatura e incluso podemos tener características de tipo lingüístico

por ejemplo el dolor. Existen muchas maneras de representar los datos de entrada ya que la información puede ser interpretada de diferentes maneras, i.e., el dolor pudiese ser una característica de tipo booleano en donde sólo se considerara la existencia o ausencia del mismo o pudiese ser una característica de tipo lingüístico en la cual se determinara en qué grado se presenta el dolor (poco, mucho, casi nada, etc.). Cada una de las posibilidades en que pueden ser consideradas las características obviamente plantea determinadas restricciones y facilidades al modelo matemático.

El segundo problema del reconocimiento de patrones es concerniente a la **selección de características** distintivas de los datos de entrada y la reducción de la dimensionalidad, es decir, la reducción de las variables en término de las cuales se describe el problema, así como la determinación de la importancia informacional de cada una de ellas.

El problema de selección de variables ha sido reconocido como un problema muy importante en el diseño de sistemas de reconocimiento de patrones. Si un conjunto completo de variables discriminantes para cada clase pudiera ser determinado por los datos de entrada el reconocimiento y clasificación de patrones presentaría muy poca dificultad, esto sería como decir que sabemos exactamente qué variables caracterizan de una mejor manera a los objetos de una clase y al mismo tiempo los diferencian de los objetos de las demás clases utilizando la mínima cantidad de variables, evitando la información redundante. Sin embargo, en la práctica en la mayoría de los problemas reales la determinación de un conjunto completo de características discriminantes mínimo es extremadamente difícil de determinar o casi imposible. Afortunadamente podemos encontrar algunas de las características discriminantes de los datos de entrada por medio de técnicas de selección de variables. Ellas nos llevan a la posible reducción del espacio de representación con la mínima pérdida de información lo que facilita el proceso de clasificación, dado que se trabaja con un espacio reducido y suficientemente discriminante. Además como ya se mencionó otro subproblema que engloba la selección de variables es el de determinar aquellas características que aportan más información, es decir, aquellas que son más importantes dentro del proceso de clasificación.

Hasta este momento hemos visto a la selección de variables como una herramienta para el proceso de clasificación, sin embargo, es importante mencionar que este puede

constituir un problema por sí mismo, no necesariamente para la clasificación. Existen problemas en la práctica en donde el interés se centra en la determinación de la importancia de las características que describen los objetos, es decir, qué variables inciden más en la aparición del mismo. Como ejemplo podemos citar el trabajo de Ruiz y Fuentes [ 12 ] para el análisis de la delincuencia juvenil, donde no tiene importancia conocer si un joven es delicuente o no, sino que dado un conjunto de características que describen a un grupo de jóvenes delincuentes; por ejemplo el status social, los antecedentes familiares delictivos, el nivel educativo, el sexo, etc; se quiere saber cuáles de éstas son las que tienen una mayor incidencia para que una persona tenga una conducta delictiva.

El tercer problema del reconocimiento de patrones está relacionado con el proceso de **clasificación**. La clasificación es la habilidad de determinar la pertenencia de un elemento u objeto a un conjunto. Por ejemplo en el caso del diagnóstico médico donde las clases están constituidas por pacientes que tienen una misma enfermedad, nos interesa el poder decidir a qué conjunto pertenece un paciente que no se encuentra dentro de los conjuntos determinados, es decir, determinar qué enfermedad tiene dicho paciente.

La Teoría de Conjuntos es una de las bases del Reconocimiento de Patrones, por lo tanto, enfocaremos el problema de clasificación dando algunos conceptos de la misma.

Los conceptos de conjunto, elemento y pertenencia son conceptos primarios de la Teoría de Conjuntos. Los conjuntos no se definen. Existen dos formas de determinarlos: por extensión y por intención.

Se dice que un conjunto es determinado extensionalmente si podemos dar una relación o un listado de los elementos que lo forman. Esto es característico de los conjuntos finitos, en especial de aquellos con cardinalidad relativamente pequeña, como por ejemplo el conjunto formado por las estaciones del año {primavera, verano, otoño, invierno}.

Un conjunto es determinado en forma intencional cuando expresamos las propiedades que caracterizan a dicho conjunto. Por ejemplo el conjunto formado por

todos los animales que son mamíferos y además tienen 4 patas, que se determina de la siguiente forma: {x | x es mamífero ∧ x tiene 4 patas }.

Dado que existen dos maneras de determinar un conjunto podemos decir que existen también dos maneras de determinar si un objeto pertenece o no a un conjunto. La primera es verificar si el elemento que buscamos se encuentra dentro de la lista de elementos del conjunto. La segunda es verificar si cumple con las condiciones que caracterizan al conjunto. Sin embargo, el problema surge cuando no se tiene completa la lista de elementos del conjunto o no se pueden determinar todas las propiedades que caracterizan al conjunto. En muchos casos se conoce la propiedad que caracteriza a los elementos que pertenecen a un conjunto pero esta propiedad es poco precisa, por ejemplo: "las personas que son jóvenes", "x es un animal pequeño".

En muchos problemas de la práctica (en la Medicina , las Geociencas, las Ciencias Sociales, la Economía, etc.) contamos con un listado incompleto de elementos que por ciertas razones son considerados por especialistas del área en cuestión, elementos de un cierto conjunto y otro listado también incompleto, de elementos de un segundo conjunto y así sucesivamente para cada uno de los conjuntos y no conocemos la (o las) propiedad (es) que caracteriza(n) a dichos conjuntos. El problema estriva justamente en, con esa información, con ese nivel de conocimiento, resolver el problema que surge cuando tenemos que decidir en cuanto a la pertenencia de un nuevo elemento a uno de dichos conjuntos. Donde se considera como "nuevo" a un elemento que no se encuentra exactamente en el listado.

Es importante mencionar que el universo sujeto a estudio, como conjunto de elementos puede ser cubierto por una familia de subconjuntos del mismo y que en unos casos serán disjuntos correspondiéndose perfectamente con el concepto de "clases" en teoría de conjuntos. En otros casos estos conjuntos no serán disjuntos y no obstante, le seguiremos llamando "clases". Esto quiere decir que en algunos problemas la pertenencia de un elemento a una clase conlleva a la negación de la pertenecia a las otras clases, es decir, la clasificación de un objeto será a lo sumo en una de las posibles clases. Por ejemplo, en el caso de la determinación de la perspectividad de una zona para determinado mineral, pertenece a la clase perspectiva o pertenece a la clase no perspectiva, pero no a ambas ya que una misma zona no puede ser perspectiva y no

perspectiva o en el caso del reconocimiento de caracteres alfanuméricos donde el patrón a ser reconocido sólo puede corresponder al nombre de una letra y no más. Sin embargo, si nuestro problema es el diagnóstico de enfermedades, nuestros estudios pueden ser tales que dado un conjunto de síntomas y signos un paciente tenga un cuadro clínico tal que corresponda con más de una de las enfermedades de dicho conjunto.

Los problemas de clasificación pueden ser divididos en tres tipos a saber:

- Problemas de clasificación con aprendizaje
- Problemas de clasificación con aprendizaje parcial
- Problemas de clasificación sin aprendizaje

Denominaremos problemas de clasificación con aprendizaje a aquellos en los cuales se tiene una muestra de objetos para cada una de las clases y además se considera que el número de clases es mayor o igual a 2. Sin embargo, existe una buena cantidad de problemas prácticos en los que si bien el universo de objetos en estudio se agrupa en 2 o más clases, no es posible disponer de la muestra de objetos de una de ellas. Por ello el "aprendizaje" del que puede disponerse no es total de ahí que a este tipo de problemas se les denomine problemas con aprendizaje parcial. Finalmente puede ocurrir que no se pueda dar, por limitaciones del conocimiento, ni siquiera un grupo de descripciones de objetos de una clase ya que el problema fundamental es que no sabemos ni cuántas clases pueden considerarse y sobre todo ni quiénes las integran por eso a este tipo de problemas se le denomina problemas sin aprendizaje. Este problema es estudiado en la literatura bajo el nombre de "cluster analysis" o agrupamientos.

Existen muchas técnicas matemáticas para la solución de los problemas de representación de los datos de entrada, selección de características y clasificación. Estas técnicas a su vez pueden agruparse en tres enfoques generales llamados:

- Enfoque lógico combinatorio para el R. P.

- Enfoque sintáctico estructural para el R.P.

- Enfoque estadístico probabilístico para el R.P.

Cada uno de estos enfoques utiliza algunas teorías y herramientas, para su desarrollo, como son: la Teoría de Conjuntos, la Teoría de Optimización, la Teoría de los Lenguajes Formales, así como la Estadística, la Probabilidad, la Teoría de Grafos, la Combinatoria, la Lógica Polivalente, la Lógica Clásica, la Lógica Difusa, las Ecuaciones Diferenciales, etc.

Dentro de cada uno de los enfoques antes mencionados existen limitantes así como ventajas y desventajas. Por lo tanto dependiendo del tipo de problema a resolver y de sus restricciones se debe elegir el enfoque o enfoques más adecuados para la solución del mismo. Nosotros nos basaremos en el enfoque lógico-combinatorio en el desarrollo de este trabajo, por lo tanto a partir de este momento todas las teorías y definiciones estarán enmarcadas en dicho enfoque.

## 1.2 Planteamiento formal de un problema de clasificación con aprendizaje

Como ya se dijo en la sección anterior existen tres tipos de problemas de clasificación, pero en este trabajo nos enfocaremos únicamente al problema de clasificación con aprendizaje.

El problema de reconocimiento de patrones desde el enfoque lógico combinatorio se describe de la siguiente forma:

Se tiene un universo M de objetos $O_1,\cdots, O_m$ que denominaremos "admisibles" divididos en $l$ clases, digamos que $K_1,\cdots, K_\ell$, denotan dichas clases, cada uno de los objetos está descrito en términos de un conjunto de rasgos $x_1,\cdots, x_n$ y cada rasgo $x_i$ tiene asociado un conjunto $M_i$ que denominaremos "conjunto de valores admisibles del rasgo $x_i$

" para $i$=1, . . . , n. Este conjunto de valores admisibles puede ser de tres tipos: nominal, ordinal o aritmético, por ejemplo, pueden ser de tipo booleano, puede tomar valores dentro de un conjunto { $a_1$, . . . , $a_k$ } donde los $a_i$ no tienen que ser necesariamente números, $i$=1, . . . , k, k≥2 o estar dentro de intervalos [ a, b ], [ a, b ), ( a, b ], ( a, b ) donde a y b son números cualesquiera. Se pueden considerar conjuntos de valores admisibles tan complejos como sea necesario.

Dentro de los valores admisibles se considera también la ausencia de información que denotaremos con el símbolo " * ", esto implicará que la información con respecto a un rasgo dado de un objeto dado no se conoce.

Dado que los objetos $O_1,\cdots, O_m$ no son todos los que aparecen o pueden aparecer en el universo de estudio entonces el problema de clasificación consiste en que dado un nuevo objeto O, el cual estará descrito en términos de las n características $O = (x_1(O),\cdots,x_n(O))$; donde $x_i(O)$ es el valor del rasgo $x_i$ en el objeto O; se quiere definir a cuál de las clases $K_1,\cdots, K_l$ pertenece. En la figura 1 se resume simbólicamente el planteamiento del problema representándose la información en forma de matriz a la cual le daremos el nombre de matriz de aprendizaje en lo sucesivo y que será definida de manera formal más adelante.

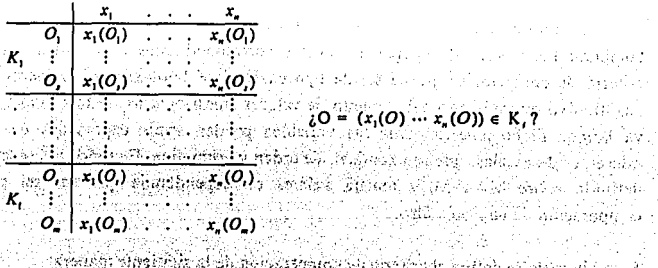



fig 1 Representación simbólica del problema de clasificación con aprendizaje

DEFINICION 1.1. - Una *descripción estándar de un objeto* O será un n-uplo $I(O) = (x_1(O),\cdots, x_n(O))$ donde $x_i(O) \in M_i$, $i = 1, \ldots, n$. Si $x_i(O) \neq *$ para $i = 1, \ldots, n$, diremos que I(O) es una *descripción completa* de O en términos de $x_1, \ldots, x_n$.

DEFINICION 2.1. - Llamaremos *tuplo informacional* de O al $\ell$-uplo $\overline{\alpha}(O) = (\alpha_1(O),\cdots,\alpha_\ell(O))$ donde $\alpha_i(O) \equiv$ "$O \in K_i$" y se cumple que $\alpha_i(O) \in \{0, 1, *\}$, $\alpha_i(O) = 0$ significará que $O \notin K_i$, $\alpha_i(O) = *$, significa que no se conoce si pertenece o no el objeto en cuestión a la clase $K_i$ y $\alpha_i(O) = 1$ que pertenece a $K_i$. Si $\alpha_i(O) \neq *$ para $i = 1, \ldots, 1$ lo llamaremos *tuplo informacional completo* y si se cumple que $\alpha_i(O) \neq *$ implica que $\alpha_i(O) = P_i(O)$ siendo $P_i(O)$ el predicado que describe correctamente la pertenencia de O a $K_i$. $\overline{\alpha}(O)$ se denominará *tuplo informacional correcto*. Finalmente denominaremos *tuplo informacional verdadero* a $\overline{\alpha}(O)$ si este es completo y correcto.

Una vez que se han precisado los conceptos de descripción de los objetos nos referiremos ahora a las clases.

DEFINICION 3.1 - Por *información estándar de las clases* $K_1,\cdots, K_\ell$ entenderemos $I_e(K_1,\cdots,K_\ell) = \{I(O_1),\overline{\alpha}(O_1),\cdots,I(O_m),\overline{\alpha}(O_m)\}$ donde $I(O_i)$ es la descripción de $O_i$ y $\overline{\alpha}(O_i)$ su tuplo informacional. $I_e(K_1,\cdots,K_\ell)$ será una *información estándar correcta (completa verdadera)* en dependencia que todos sus tuplos informacionales sean correctos (completos, verdaderos).

Denominaremos *criterio de comparación* $\delta_i$ a la función con la cual podemos comparar los valores de un par de objetos correspondientes a un mismo rasgo $x_i$. El criterio de comparación puede ser de tipo cualitativo( booleano o k-valente) o de tipo cuantitativo dependiendo del conjunto de valores admisibles $M_i$ de la variable $x_i$. Como ya hemos dicho anteriormente las variables pueden venir expresadas en tres tipos diferentes de escalas, que son nominal, de orden y aritmética. En todos los casos $\delta_i$ estará definido sobre $M_i \times M_i$ y tomará valores en dependencia que sea un criterio de comparación de un tipo u otro.

Formalmente, se define el criterio de comparación de la siguiente manera:

$$\delta_i(x_i(O_i), x_i(O_v)): M_i \times M_i \rightarrow V_i$$

Donde $V_i$ pudiera ser {0,1}, en el caso en que el resultado de la comparación de los valores de la variable $x_i$ sean "coincidentes" o no. $V_i$ puede tomarse como [0, 1] si $\delta_i$ es una métrica o si la respuesta de la comparación de dos valores es por ejemplo de naturaleza difusa.

A continuación daremos algunos ejemplos de criterios de comparación donde $x_i(O_s) \in M_i$ y $x_i(O_t) \in M_i$

$$1)\ \delta_i(x_i(O_s),x_i(O_t)) = \begin{cases} 1 & \text{si } x_i(O_s) = x_i(O_t) \vee x_i(O_s) \neq * \vee x_i(O_t) \neq * \\ 0 & \text{si } x_i(O_s) \neq x_i(O_t) \end{cases}$$

Este criterio de comparación es el de la igualdad, esto quiere decir que para que dos valores sean coincidentes tienen que ser estrictamente iguales. En este caso $V_i$ es de tipo booleano.

$$2)\ \delta_i(x_i(O_s),x_i(O_t)) = \begin{cases} 1 & \text{si } |x_i(O_s) - x_i(O_t)| \leq \varepsilon_i \\ 0 & \text{en otro caso} \end{cases}$$

En este criterio de comparación se dirá que dos valores son "coincidentes" si su diferencia es menor que un número $\varepsilon$.

3) Sea $M_i = \bigcup_{l=1}^{s} [a_l, a_{l+1}),\ a_j \in M_i,\ j = 1,\cdots,s+1$

$$\delta_i(x_i(O_r), x_i(O_t)) = \begin{cases} 1 & \text{si } (x_i(O_r)) \in [a_l, a_{l+1}) \wedge (x_i(O_t)) \in [a_l, a_{l+1}) \\ & \qquad \vee\, x_i(O_r) \neq * \ \vee\, x_i(O_t) \neq * \\ 0 & \text{en otro caso} \end{cases}$$

En este criterio de comparación dos valores son "coincidentes" si caen dentro de un mismo intervalo.

Así como los ejemplos anteriores, existe una cantidad muy grande de criterios de comparación que dependen del problema específico que se esté modelando. En general la mayoría de los criterios de comparación de valores de una variable es del tipo cualitativo booleano o aritmético y en este último caso se usa por lo general una métrica.

Los criterios de comparación forman la base para lo que denominaremos *funciones de semejanza* β las cuales nos permiten hacer comparaciones entre pares de descripciones de objetos. El valor de la función de semejanza se calcula fundamentalmente sobre la base de los criterios de comparación de valores de cada una de las variables que intervienen en la descripción de los objetos. Al igual que los criterios de comparación, las funciones de semejanza pueden ser de tipo cualitativo (booleano o k-valente) o de tipo cuantitativo. En el caso en que β sea booleana ésta denotará si las descripciones de los objetos comparados son semejantes o no. En el caso finito-valente ésta nos dará una gradación de la semejanza entre las descripciones de los dos objetos.

Formalmente denotamos a las funciones de semejanza como:

$$\beta: \prod_{i=1}^{n} M_i \times \prod_{i=1}^{n} M_i \rightarrow V$$

V se define de igual manera que $V_i$ en los criterios de comparación, es decir que V puediera ser {0, 1} si el resultado de la comparación entre un par de objetos es expresado en dos posibles respuestas (semejantes o no semejantes). Puede tomarse como [0, 1 ] en caso de que β sea una métrica o el resultado de la comparación entre un par de objetos sea de naturaleza difusa.

Algunos ejemplos de funciones de semejanza son los siguientes:

$$1)\ \beta(I(O_s),I(O_t)) = \begin{cases} 1 & \text{si } \delta_i(x_i(O_s), x_i(O_t)) = 1 \quad \forall i = 1,\cdots,n \\ 0 & \text{en otro caso} \end{cases}$$

Con esta función de semejanza diremos que dos objetos son semejantes si todos los valores correspondientes a sus descripciones son coincidentes dependiendo del criterio de comparación $\delta_i$ que se utilice para cada variable $x_i$. Evidentemente el criterio de comparación debe de ser de tipo booleano.

$$2)\ \beta(I(O_s),I(O_t)) = \begin{cases} 1 & \text{si } |\delta_i(x_i(O_s), x_i(O_t)) = 0| \le \varepsilon \quad \forall i = 1,\cdots,n \\ 0 & \text{en otro caso} \end{cases}$$

En este caso diremos que dos objetos son semejantes si los valores no coincidentes en sus descripciones, en dependencia del criterio de comparación $\delta_i$ que se utilice para la

variable $x_i$, son menores o igual que un umbral ε determinado. En este caso también los criterios de comparación deben de ser de tipo booleano.

$$3)\ \beta(I(O_s),I(O_t)) = \begin{cases} 1 & \text{si } |\delta_i(x_i(O_s),x_i(O_t)) = 0| \le \varepsilon_1 \quad \forall i = 1,\cdots,n \\ & \text{si } |\delta_i(x_i(O_s),x_i(O_t)) = 1| \ge \varepsilon_2 \\ 0 & \text{en otro caso} \end{cases}$$

Esta función de semejanza es similar a la anterior nada más que en ésta se ponen dos umbrales, es decir, que dos objetos serán semejantes si el número de valores coincidentes entre sus descripciones es mayor o igual que un $\varepsilon_2$ y si además el número de valores no coincidentes es menor o igual que un $\varepsilon_1$. Como en los casos anteriores esto depende del criterio de comparación que se utilice para cada $x_i$, siendo estos de tipo booleano.

$$4)\ \beta(I(O_s),I(O_t)) = \frac{1}{n}\sum_{i=1}^{n}\delta_i(x_i(O_s),x_i(O_t))$$

Esta función de semejanza nos da una gradación de la semejanza entre las descripciones de dos objetos, ya que se suman los valores de los criterios de comparación correspondientes a cada variable $x_i$ y se divide toda esta suma entre n que es el número de rasgos. en este caso los criterios de comparación $\delta_i$ no necesariamente tienen que ser de tipo booleano sino que pueden ser de cualquier tipo.

Como en los criterios de comparación existen decenas de funciones de semejanza pero en general estas funciones se derivan de un proceso de modelación matemática de

los problemas particulares que se quieren resolver, es decir, que se debe utilizar la función de semejanza que mejor modele nuestro fenómeno. Quizás encontrar dicha función no sea fácil ya que como se dijo anteriormente esto conlleva a un proceso de modelación matemática del problema. Es por eso que en la Teoría de Reconocimiento de Patrones las funciones de semejanza son sin lugar a dudas uno de los conceptos esenciales y más importantes, sin ellas el reconocimiento de patrones no tendría sentido.

Una vez que se han dado los conceptos básicos dentro de los problemas de reconocimiento de patrones haremos la formalización del problema de clasificación con aprendizaje.

## PLANTEAMIENTO FORMAL DEL PROBLEMA DE CLASIFICACION CON APRENDIZAJE

Sea M un conjunto de objetos admisibles los cuales están descritos en términos de $x_1,\cdots, x_n$ rasgos o características cuyos conjuntos de valores admisibles son $M_1,\cdots, M_n$ y en los que se han definido criterios de comparación $\delta_1, \ldots, \delta_n$ respectivamente. Sea $K_1,\cdots,K_\ell$ un cubrimiento finito de subconjuntos propios de M. Denominaremos *muestra o matriz de aprendizaje* a la información estándar verdadera $I_0(K_1',\cdots,K_\ell')$ de las clases $K_1,\cdots,K_\ell$ donde $K_i' \subset K_i$ , $i=1, \ldots, \ell$ y denominaremos *muestra o matriz de control* al conjunto $I(O_1'),\cdots,I(O_q')$ de descripciones estándar , que pudieran no ser completas, de los objetos admisibles $O_1',\cdots,O_q'$.

El problema consiste en hallar un algoritmo A tal que:

$$A(I_0(K_1',\cdots,K_\ell'),I(O_1'),\cdots,I(O_q')) = \left\|\alpha_j^A(O_i')\right\|_{q\times\ell}$$

siendo $\alpha_j^A(O_i') \in \{0, 1, *\}$ la respuesta del algoritmo A en cuanto a la pertenencia de $O_i'$ a la clase $K_j$, $\left\|\alpha_j^A(O_i')\right\|_{q\times l}$ denota una matriz de q filas y l columnas, * denota la abstención del algoritmo A a clasificar $O_i$.

Existen algunas observaciones que podemos hacerle a este planteamiento, la primera es que no tiene que ocurrir que $\alpha_j^A(O_i') \in \{0, 1, *\}$, es decir, que las clases no necesariamente tienen que ser disjuntas sino que pueden existir elementos multiclasificados, o quizá se pudiera hablar de clases difusas donde todos los objetos pertenecen a todas las clases en cierto grado. Puede ocurrir también que se carezca de información de los objetos de alguna de las clases o que no existan dichas clases. En segundo lugar el algoritmo A puede equivocarse, es decir, $\alpha_j^A(O_i') \neq P_j(O_i')$. En otras palabras los tuplos informacionales que produce A para cada objeto a clasificar $O_i'$ pudieran no ser completos e incluso no ser correctos. Esto significa que de alguna manera la eficiencia del algoritmo no necesariamente es del 100 por ciento al momento de la clasificación, lo cual nos lleva a pensar que la eficiencia de clasificación de un algoritmo puede ser medida sobre la base de los objetos que clasificó mal y de los objetos en los cuales se abstuvo de hacer una clasificación.

## 1.3 Análisis de datos y la determinación de anomalías

Generalmente cuando se habla de análisis de datos, se habla de determinar cómo se comportan ycómo están relacionados los datos de una muestra dada para tomar ciertas decisiones con respecto a dichos datos. Dentro del análisis de datos encontramos una serie de medidas y técnicas que nos sirven de herramientas para determinar dicho comportamiento. Muchas de esta medidas son de tipo estadístico como son: la varianza, la media, la moda, la desviación estándar, la frecuencia, etc., con las cuales podemos establecer con que frecuencia aparecen los datos, qué datos tienen frecuencia máxima, es decir, cuál es la media de los datos, qué tan semejantes son los datos con respecto al dato(s) más frecuente(s), cómo se distribuyen los mismos, etc. Asimismo existen otras medidas que nos proporcionan la relación entre las variables por medio de las cuales describimos a los objetos de nuestra muestra, como es la medida de correlación. Sin embargo, existen otros enfoques y métodos para hacer análisis de datos en los cuales no se utizan las herramientas estadísticas, como es el caso del enfoque lógico-combinatorio para el Reconocimiento de Patrones.

Como vemos se puede hacer análisis de datos con diferentes propósitos, utilizando conceptos que no necesariamente tienen que ser siempre los mismos. Por ejemplo se

pueden analizar los objetos de una matriz dada que no está dividida en clases, precisamente para determinar cómo se agrupan dichos objetos, es decir, definir cuáles son las clases en las que está dividida dicha matriz. En este caso el proceso de agrupamiento se puede llevar a cabo verificando cuáles son las características que cumplen los objetos y agrupar aquellos que se asemejan más con respecto a dichas características. Ejemplos de esto son los algoritmos de Holotipo y Class [ 11 ]. Otro tipo de análisis es el determinar qué tan importante es un objeto dentro de una clase dada, i.e., qué tanta información aporta o qué tan característico es el objeto para la clase. Este proceso se puede realizar de diferentes maneras, una de ellas es la de verificar qué tan frecuente es la aparición de los datos que describen a dicho objeto dentro de la clase y apartir de ahí definir su importancia. O se puede obtener por medio de la semejanza con respecto a los objetos de la clase. Esto nos lleva a pensar en otra aplicación como sería la formación de matrices de aprendizaje y control apartir de una matriz inicial.

En general en los problemas de clasificación sería ideal contar con una muestra o matriz de aprendizaje en la cual todos sus objetos fueran representativos de sus respectivas clases, es decir, que los objetos fueran lo suficientemente discriminantes entre las clases dadas para que el proceso de clasificación fuera más rápido y que se cometiera el mínimo error en la clasificación, sin embargo, la construcción de una matriz de aprendizaje, capaz de respaldar al clasificador de manera adecuada, constituye una tarea difícil. En general en la mayoría de las aplicaciones reales no se cuenta con una muestra lo suficientemente representativa. Es por eso que en muchos problemas de clasificación surge la necesidad práctica de contar con un procedimiento para seleccionar los datos más importantes o que nos proporcionan una mayor discriminación entre las clases, eliminando todos aquellos objetos que no son representativos y por tanto no nos proporcionan una buena discriminación.

Hay que considerar que la construcción de las matrices de aprendizaje o de control no son la única aplicación del análisis de datos, aunque constituye una de las más importantes. Existen algunos problemas, la mayoría dentro de las Geociencias, es decir, Geología, la Geofísica, la Geoquímica y otras, en los cuales se quiere saber cuál es el comportamiento de los datos ya que la aparición de valores anómalos nos pudiera reflejar la presencia de fenómenos raros que nos dieran indicios, por ejemplo, de la existencia de yacimientos de un determinado mineral.

Se puede hablar de análisis de datos donde en lugar de considerar a todas las variables que describen a los objetos para hacer agrupamientos, calcular la medida de información de un objeto, etc., se consideren subconjutos de dichas variables para hacer el análisis.

En general podemos decir que el proceso de análisis de datos puede ser utilizado como una herramienta dentro de otros problemas tanto en el caso de la Estadística, como en el caso del Reconocimiento de patrones, sin embargo, es importante mencionar que por sí solo forma un problema.

# CAPITULO 2 Algoritmo de Clasificación para el análisis de datos

## 2.1 Conceptos Básicos

El algoritmo que se describirá en la siguiente sección está enmarcado dentro de los problemas de clasificación con aprendizaje. Este algoritmo es para el análisis y clasificación de datos que como ya mencionamos en el capítulo 1 constituye una parte muy importante para el Reconocimiento de Patrones. En especial este algoritmo fue elaborado y ha sido utilizado en problemas de la Geología para la detección de anomalías. En este trabajo lo expondremos en una forma más general ya que consideramos que puede ser utilizado en cualquier otra área.

En los párrafos siguientes se darán los conceptos fundamentales del algoritmo, algunas definiciones de tipo estadístico y finalmente se hará el planteamiento formal del problema de análisis de datos.

El algoritmo trabaja en torno a dos ideas o conceptos básicos que son los de "tipicidad" y "contraste". Específicamente en este algoritmo el concepto de tipicidad está altamente ligado al concepto de *frecuencia*, es decir, nosotros diremos que un objeto es típico de una clase determinada si los valores que describen a dicho objeto son muy frecuentes con respecto a los valores de los objetos de dicha clase. Por lo tanto mientras más frecuente sea la coincidencia de los valores que describen a un objeto con respecto a los objetos de una clase determinada, significará que el objeto es más típico; de manera contraria mientras menos frecuente sea la coincidencia de los valores de un objeto con respecto a los objetos de una clase éste será menos típico. Como podemos observar la tipicidad puede presentarse en gradaciones diferentes dependiendo de qué tan frecuentes sean los valores que describen a los objetos. Cabe mencionar que el concepto de tipicidad no necesariamente tiene que estar relacionado con la frecuencia, aunque en este caso como ya se dijo al principio de este párrafo, únicamente nos basaremos en la frecuencia para obtener la tipicidad de los objetos.

El concepto de contraste está basado en el concepto de tipicidad. El contraste nos da una medida de qué tan diferente es la tipicidad de un objeto de una clase determinada

con respecto a la tipicidad del mismo objeto correspondiente a cada una de las clases restantes. Si el grado de tipicidad de un objeto para una clase es "muy grande" y a su vez el grado de tipicidad de dicho objeto para las clases restantes es "pequeño" diremos que el objeto es muy contrastante. De manera contraria, si el grado de tipicidad del objeto para cada una de las clases es similar, diremos que el objeto no es muy contrastante. Como vemos, al igual que la tipicidad el contraste puede presentarse en gradaciones diferentes dependiendo de qué tan típico sea un objeto con respecto a cada una de las clases. Asimismo el contraste no sólo puede ser definido sobre la base de la tipicidad sino que pudieran presentarse otras definiciones de contraste, aunque en este caso está definido sobre la base de ella y por consiguiente con el concepto de frecuencia.

Es claro que el contraste y la tipicidad forman la esencia del algoritmo. Una vez que se obtuvieron la tipicidad y el contraste de cada uno de los objetos se analizan estas medidas y se hace la clasificación de cada uno de los objetos.

A continuación daremos algunas definiciones estadísticas que servirán de herramienta en la descripción del algoritmo.

Cuando se dispone de un gran número de datos, es útil el distribuirlos en clases o categorías y determinar el número de individuos pertenecientes a cada clase, que es la *frecuencia* de la clase.

DEFINICION 2.1.- Una ordenación tabular de los datos en clases, con las frecuencias correspondientes a cada una, se conoce como una *distribución de frecuencias* o tabla de frecuencias.

DEFINICION 2.2.- Un símbolo que define una clase tal como $[x, y]$ , se conoce como *intervalo de la clase*. Los números extremos $x$ y $y$, son los límites de clase. El número menor $x$ el límite inferior de la clase y el mayor $y$ es el límite superior. Los términos clase e intervalo de clase se utilizan a menudo indistintamente, aunque el intervalo de clase es un símbolo para la clase. Un intervalo de clase que, al menos teóricamente, no tiene límite superior o inferior, se conoce como intervalo de clase abierto. Por ejemplo al referirse a la edad de grupos de individuos el intervalo de clase, < mayores de 65> es un intervalo de clase abierto.

DEFINICION 2.3.- La *marca de clase* es el punto medio del intervalo de clase y se obtiene sumando los límites inferior y superior de la clase y dividiendo entre 2. La marca de clase se denomina también punto medio de clase.

Las reglas generales para formar las distribuciones de frecuencia son:

1.- Determinar el mayor y el menor entre los datos registrados y así encontrar el rango (diferencia entre el mayor y el menor de los datos).

2.-Dividir el rango en un número conveniente de intervalos de clase del mismo tamaño. Si esto no es posible, utilizar intervalos de clase de diferentes tamaños o intervalos de clase abiertos. El número de intervalos de clase se toma generalmente entre 5 y 20 dependiendo de los datos. Los intervalos de clase se eligen también de forma que las marcas de clase o puntos medios coincidan con datos realmente observados. Esto tiende a aminorar el llamado error de agrupamiento.

3.- Determinar el número de observaciones que caen dentro de cada intervalo de clase, es decir, encontrar las frecuencias de clase.

DEFINICION 2.4.- Denominaremos *histograma de frecuencias* a la representación gráfica de las distribuciones de frecuencia. Un histograma de frecuencia consiste en una serie de rectángulos que tienen

(a) Sus bases sobre un eje horizontal (el eje X) con centros en las marcas de clase y longitud igual al tamaño de los intervalos de clase.

(b) Superficies proporcionales a las frecuencias de clase. Si los intervalos de clase tienen todos igual tamaño, las alturas de los rectángulos son proporcionales a las frecuencias de clase y se acostumbra en tal caso a tomar las alturas numéricamente iguales a las frecuencias de clase. Si los intervalos de clase no son de igual tamaño, estas alturas deberán ser calculadas.

La figura (1.a) muestra la gráfica de un histograma de frecuencias donde (a, b), (b, c), (c, d), (d, e), (e, f) representan los intervalos de frecuencia.

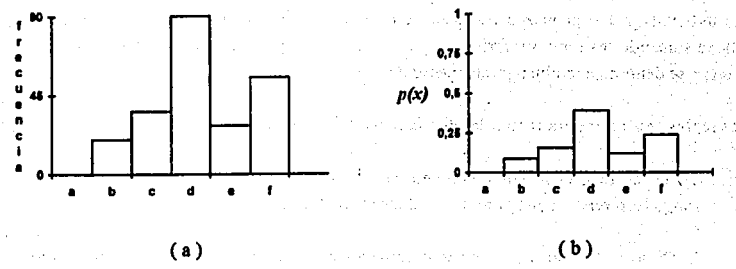



fig 1.

DEFINICION 2.3.- Si una variable $X$ puede tomar una serie de valores $X_1, X_2, \ldots, X_k$ con probabilidades respectivas $p_1, p_2, \ldots p_k$ donde $p_1 + p_2 + \ldots + p_k = 1$, se dice que ha sido definida para X una *distribución de probabilidad discreta.* La función $p(X)$ que toma los valores respectivos $p_1, p_2, \ldots p_k$ para $X = X_1, X_2, \ldots, X_k$ se llama *función de probabilidad o función de frecuencia* de $X$.

DEFINICION 2.4.- Denominaremos *histograma de probabilidades* a la representación gráfica de las distribuciones de probabilidad. El histograma de probabilidades al igual que el de frecuencia consiste en una serie de rectángulos sólo que en este caso la suma de las áreas de los rectángulos es 1, como se muestra en la figura ( 1.b )

Una vez que hemos dado los conceptos y definiciones importantes del algoritmo, haremos la formalización del problema de clasificación y análisis de datos.

Se tiene un problema como el que se planteó en la sección 1.2 del capítulo 1, que está representado simbólicamenten en la matriz de aprendizaje(MA). En este caso los valores admisibles de cada uno de los rasgos son de tipo real y dentro de ellos no se considera la ausencia de información.

Cada una de las clase de MA puede estar constituido por objetos de 4 tipos dependiendo de qué tan típico y contrastante sea cada uno de los objetos con respecto a

los objetos de su clase y a los objetos de las clases restantes. Así podemos tener los siguientes tipos de objetos:

I.- Objetos "*Propios*"

Son objetos característicos solamente para la clase a la que los mismos están asociados, es decir, estos objetos presentan ***semejanza "suficientemente alta"con los objetos de su clase y se diferencian "significativamente" de los objetos de otras clases,*** por lo tanto, se dice que tienen un alto grado de tipicidad dentro de su clase y a su vez también tienen un alto grado de contraste.

II.- Objetos "*Generales*"

Estos objetos son ***característicos para todas o la mayoría de las clases,*** es decir, presentan semejanza simultánea con los objetos de algunas o todas las clases y por tanto fijan o representan procesos y fenómenos ampliamente distribuidos en la región objeto de estudio. En este caso el grado de tipicidad del objeto es similar para la mayoría de las clases y por consiguiente el grado de contraste es bajo.

III.- Objetos "*Atípicos*"

Estos objetos ***no son característicos para ninguna de las clases,*** es decir que el grado de tipicidad con respecto a cada una de las clases es muy bajo. Los objetos de este tipo representan fenómenos raros en los límites del territorio objeto de estudio y por tanto presentan interés particular para la búsqueda de yacimientos de minerales útiles.

IV.- Objetos "*No propios*"

Son objetos ***no característicos para la clase a la que los mismos están asociados.*** Estos objetos se diferencian significativamente de los objetos de su clase. Los mismos pueden representar:

- Fenómenos no reflejados en los mapas correspondientes al criterio de agrupamiento utilizado.

- Fenómenos de difícil pronóstico.

- Presencia de variedades escasamente distribuidas en los límites de las clases consideradas

En general, en las Geociencias, la presencia de gran número de objetos de este tipo se explica por un limitado grado de estudio de la región donde se ubican las anomalías sometidas al procesamiento

## 2.2 Descripción del algoritmo

El algoritmo de clasificación que a continuación se describe fue propuesto por Diordenko L. V. y por Vaskosvskii B. V. [ 4 ]

Como ya hemos mencionado, el objetivo fundamental del algoritmo es clasificar los objetos de una matriz de aprendizaje, dentro de los cuatro tipos de objetos que se describieron en la sección 2.1

El algoritmo consta de dos etapas.

PRIMERA ETAPA

Por su esencia corresponde al proceso de aprendizaje en los algoritmos tradicionales de reconocimiento de patrones.

Los pasos a seguir son:

Paso 1).- Confección de histogramas de los atributos seleccionados.

En este paso se construyen los histogramas de frecuencia de todos los atributos seleccionados para cada una de las clases, tomando en consideración y respetando las reglas generales para formar las distribuciones de frecuencia y por consiguiente los

histogramas de frecuencia. En la figura 2 se muestra cómo deben de quedar los histogramas de frecuencia para cada uno de los atributos.

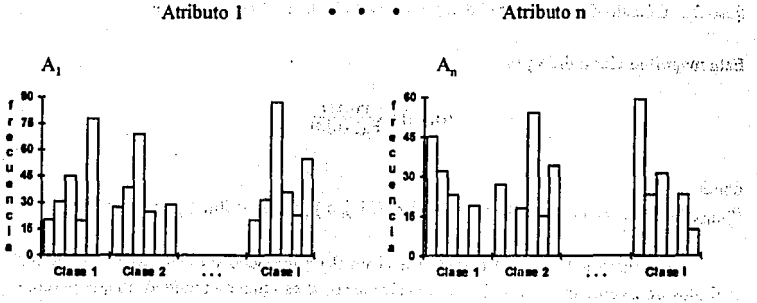



fig. 2 Histogramas de frecuencia de los atributos para cada una de las clases

Después de construir los histogramas de frecuencia, éstos se convierten en histogramas de probabilidad utilizando la expresión:

$$P(i,j,s) = \frac{F(i,j,s)}{N(j,s)}$$

donde:

P( i, j, s): Probabilidad del atributo j en el intervalo i para la clase $K_s$.

**F( i, j, s ):** Frecuencia del atributo j en el intervalo i para la clase $K'_s$.

**N( j, s )** : Número total de valores del atributo j en la clase $K'_s$.

**con s=1,. . . , $\ell$**

**Paso 2).- Cálculo de la magnitud t( i, j, s ) para cada intervalo en cada clase.**

**Esta magnitud viene dada por:**

$$t(i,j,s) = \frac{P(i,j,s)}{P_{jmax}(i,j,s)}$$

**donde:**
**$P_{jmax}$(s): Valor máximo de la probabilidad P( i, j, s ) para el atributo j en la clase $K'_s$.**

**La magnitud t( i, j, s ) caracteriza el nivel de tipicidad en cada intervalo de los atributos para cada una de las clases. Su valor máximo es igual a la unidad, lo que permite comparar entre sí los distintos intervalos por el nivel de su tipicidad para distintas clases.**

**En otras palabras t( i, j, s) caracteriza la probabilidad de aparición del atributo j en el intervalo i en cada clase, expresada en fracciones de probabilidad máxima. Esta magnitud puede ser considerada como una expresión normalizada de la probabilidad.**

**El cambio de P( i, j, s) por t(i, j, s) permite comparar las clases independientemente de las funciones de distribución de cada atributo y de los volúmenes desiguales de las clases.**

**Paso 3).- Determinación del nivel de tipicidad de cada objeto para cada atributo.**

**En este paso se comparan los valores de los atributos que caracterizan al objeto que se analiza con los histogramas obtenidos para dichos atributos en cada clase.**

Esta comparación permite definir a qué intervalos de cada histograma corresponde el valor de cada uno de los atributos en términos de los cuales está descrito el objeto, sustituyendo el valor de cada atributo por los valores de t( i, j, s ) correspondientes.

En este caso la magnitud t( i, j, s ) también constituirá una medida de la tipicidad del objeto para el cual el valor del atributo j "cae" en el intervalo i del histograma construido para el mismo. El índice del número del intervalo ( i ) cuando se habla de objetos puede ser omitido y por tanto la tipicidad de cada objeto concreto para uno de los atributos en cada una de las clases en lo adelante se designará como $t_j( s )$ donde:

j: índice correspondiente al atributo.
s: índice correspondiente a la clase.

Como resultado de la comparación mencionada se sustituye el valor de cada atributo para cada clase por los valores de $t_j( s )$.

En este caso, la expresión inicial del objeto en función de los atributos originales

$$O = (x_1(O), x_2(O), \cdots, x_n(O))$$

se convierte en una expresión matricial del tipo:

$$O = f\left\{t_j(s)\right\}_{n,\ell}$$

donde:

n: Cantidad de atributos
$\ell$: Cantidad de clases

O sea, cada objeto viene expresado en términos de un arreglo bidimensional. como se muestra en la tabla de la figura 3.

| | $x_1$ | $x_2$ | ... | $x_n$ |
|---|---|---|---|---|
| Clase $K_1'$ | $t_1(1)$ | $t_2(1)$ | ... | $t_n(1)$ |
| Clase $K_2'$ | $t_1(2)$ | $t_2(2)$ | ... | $t_n(2)$ |
| ⋮ | ⋮ | ⋮ | ... | ⋮ |
| Clase $K_\ell'$ | $t_1(\ell)$ | $t_2(\ell)$ | ... | $t_n(\ell)$ |

fig. 3

Con este paso concluye la primer etapa del algoritmo.

SEGUNDA ETAPA

La segunda etapa de trabajo del algoritmo comienza con:

Paso 1).- Cálculo de la magnitud de la tipicidad T para cada uno de los objetos de la matriz en cada una de las clases con respecto a los atributos seleccionados.

$$T(s) = \sum_{j=1}^{n} t_j(s)$$

Como es evidente, para cada objeto se calculan $\ell$ valores de T( s ) en correspondencia con la cantidad de clases. Este parámetro constituye una medida que muestra en cuánto un objeto dado es típico entre todos los objetos de la clase s.

El objeto será más típico para aquella clase donde T( s ) sea máximo por tanto el nivel de tipicidad del objeto ( T ) vendrá dado por :

$$T = \max\{T(1), T(2), \cdots, T(\ell)\}$$

Para cada objeto $O_i$ obtendrá por tanto un valor de T que denominaremos ( $T_i$ ).

Resulta también importante determinar la clase asociada a $T_i$ , o sea, el máximo de los valores de T(s ). Esta clase la denominaremos $CM_i$ .

Paso 2).- Cálculo de la magnitud del contraste ( C ) del objeto.

Esta magnitud viene dada por :

$$C = \left[ \frac{\sum_{s=1}^{\ell} (T_{max} - T(s))}{\ell - 1} \right]^{\frac{1}{2}}$$

La magnitud C constituye una medida del nivel de diferenciación del objeto en relación con el resto de las clases, para las cuales el valor de T( s ) no es máximo. Para cada objeto se determinará un valor de C ($C_i$ ).

Paso 3).- Determinación de las magnitudes tipicidad ($T_i$ ) y contraste ($C_i$ ) para cada uno de los objetos.

En este paso es necesario dividir el recorrido de las magnitudes de tipicidad ($T_i$ ) y contraste ($C_i$ ) para todos los objetos en dos intervalos. Para realizar esta tarea se definirá por el usuario una magnitud que denominamos PORCIENTO DE DELIMITACION

ALTO - BAJO (POR; POR = 1, 2, . . . , 100). Los valores de umbral para $T_i$ y $C_i$ se calculan por medio de las siguientes expresiones:

$$T_o = \frac{POR}{100} \times (T_{max} - T_{min})$$

$$C_o = \frac{POR}{100} \times (C_{max} - C_{min})$$

Tmax, Cmax : Valores máximos de $T_i$ y $C_i$ para el conjunto de objetos analizados.

Tmin, Cmin : Valores mínimos de $T_i$ y $C_i$ para el conjunto de objetos analizados.

Paso 4).- Clasificación de los objetos en los tipos I, II, III, IV.

TIPO I Objetos Propios
$T_i$ : alto, o sea, $T_i > T_o$
$C_i$ : alto, o sea, $C_i > C_o$
$CM_i = K_i$

TIPO II Objetos Generales
$T_i$ : alto, o sea, $T_i > T_o$
$C_i$ : bajo, o sea, $C_i \leq C_o$

TIPO III Objetos Atípicos
$T_i$ : bajo para cualquier valor de $C_i$ ( $T_i \leq T_o$ )

TIPO IV Objetos No Propios
$T_i$ : alto, o sea, $T_i > T_o$
$C_i$ : alto, o sea, $C_i$
$CM_i \neq K_i$

Una vez que todos los objetos de MA han sido clasificados dentro de los cuatro tipos de objetos, el algoritmo finaliza.

## 2.3 Limitantes del algoritmo

Como ya hemos repetido muchas veces a lo largo de este trabajo, el Reconocimiento de Patrones está relacionado con ciencias o disciplinas como la Medicina, Geología, Pedagogía, Sociología, etc. en las cuales frecuentemente surgen problemas de clasificación, como por ejemplo, el diagnóstico médico, la determinación del nivel de productividad de un yacimiento de recursos minerales a partir de ciertos datos geológicos y geofísicos, etc. En general la mayoría de las veces estos problemas no son solucionados de una manera adecuada, ya que para su solución se hace uso de modelos y métodos que no se ajustan de la mejor manera a las características de dichos problemas, sino por el contrario tratan de que los problemas se ajusten a las condiciones que impone el modelo. Estas condicionantes hacen casi imposible la utilización del modelo o método, sin violentar los principios elementales de la modelación matemática. Tal es el caso del algoritmo que se presentó en la sección 2.2 de este capítulo, el cual tiene un conjunto de condiciones o limitantes que hacen que no se deba aplicar a la mayoría de los problemas de clasificación citados. Aunque cabe mencionar que la esencia del algoritmo es buena y que este algoritmo pudiera ser de mucha utilidad si se eliminaran dichas limitantes. A continuación enlistaremos las limitantes del problema para después dar las razones por las cuales se considera que éstas constituyen un problema.

Limitantes:

1) Unicamente trabaja con variables de tipo real.

2) No considera la posibilidad de ausencia de información.

3) Considera que todas las variables proporcionan la misma información.

4) No permite utilizar diferentes criterios de comparación.

5) Utiliza como función de semejanza la igualdad estricta.

6) No clasifica objetos nuevos, es decir, objetos que no están en la matriz de aprendizaje.

Además de estas limitantes existen algunas consideraciones y herramientas que se utilizan en el desarrollo del algoritmo que no se ajustan a las características de los problemas en general. Como es el caso de las técnicas estadísticas utilizadas cuyos presupuestos en general es muy raro que se cumplan en la práctica.

1) Sucede que en muchos de los problemas de clasificación de las ciencias antes citadas se involucran variables tanto cuantitativas como cualitativas y con frecuencia en forma simultánea. Como es el caso del diagnóstico médico en el cual se pueden considerar variables que tomen dos valores únicos, como el sexo, variables que tomen valores numéricos, como el peso o la estatura , variables de tipo booleano, como la ausencia o presencia de fiebre, hasta variables de tipo lingüístico como es el caso del dolor. Así como este problema existen muchos otros en los cuales se utilizan diferentes tipos de variables. Por lo tanto no se puede pedir que los problemas sean representables en el espacio de los números reales.

2) En muchas ocasiones en los problemas no contamos con la información completa para todos los objetos, es decir, que existen propiedades o variables que no han sido estudiadas para todos los objetos ya sea porque esto resulte imposible o porque sea muy costoso hacerlo. Como por ejemplo, en el caso de la Geología donde para determinar si una zona es perspectiva se consideran características o variables como el campo magnético, algunas propiedades geomorfológicas y geoquímicas, pero algunas veces esta información no puede ser proporcionada como es el caso del campo magnético de una zona pantanosa. En el caso de la Medicina existen algunas características de algunos pacientes que se desconocen como pudieran ser un antecedente familiar de tipo genético. Como vemos es muy probable que en muchos problemas exista falta de información, sin embargo, en este algoritmo no se toma en cuenta la ausencia de información y por consiguiente no se sabe qué hacer cuando aparecen este tipo de problemas. En algunos algoritmos el problema es peor ya que se hace una estimación de la información faltante.

3) Las variables que describen a los objetos no siempre nos proporcionan la misma información, es decir, que existen algunas variables que son más importantes que otras y por eso mismo éstas resultan ser más discriminantes cuando se lleva a cabo el proceso de clasificación. Muchas veces los expertos de antemano proporcionan una gradación de la importancia informacional de la aparición de cada una de las variables. Evidentemente las variables que son casi imprescindibles dentro del proceso de clasificación tienen un peso informacional muy grande y aquellas que son importantes pero no impresindibles tienen un peso informacional no muy grande. Un ejemplo de esto sería en el diagnóstico del estado de salud de un paciente donde la presencia de la variable fiebre sea más importante que la presencia de la variable vómito o fumar.

4) El criterio de comparación que se maneja en el algoritmo es un criterio por intervalos y este criterio es el mismo para todas las variables. Pero esto constituye un gran problema ya que como se ha dicho, en los problemas reales las variables pueden ser de cualquier tipo y por lo tanto los criterios de comparación para cada una de las variables debe depender del tipo de variable que se tenga y del concepto de analogía o semejanza que se haya determinado en el momento de la modelación del problema, para cada variable. Incluso en el caso que se considere por el especialista sólo variables reales, digamos por caso, no siempre se encuentran fundamentos en la disciplina en particular (Geociencias, Medicina, etc.) para justificar el tratamiento uniforme de las variables. Además al construir los intervalos se pide que se cumplan una serie de reglas que no tienen sentido en los problemas reales, como es el exigir que el número de intervalos sea mínimo de 5 y máximo 20 y que la cantidad de valores que exista en cada intervalo sea mínimo de 5.

5) De la misma manera que los criterios de comparación, las funciones de semejanza no tienen porqué limitarse a un sólo tipo. En este caso la función de semejanza es la igualdad estricta, es decir, que para que dos objetos sean semejantes todos los valores de sus descripciones tienen que ser iguales, pero existen problemas en los cuales no es necesario que las descripciones de los objetos sean iguales sino que siendo algunos de los valores de sus descripciones iguales es suficiente para considerarlos semejantes. Por supuesto que las funciones de semejanza que se utilicen dependen de la modelación matemática del problema en cada caso. Por lo tanto no tiene porqué forzarse al problema a ajustarse a una función determinada.

**6) El algoritmo sólo clasifica los objetos que se encuentran en la matriz de aprendizaje, sin embargo, en muchos problemas de la realidad lo que se quiere es clasificar objetos que no se encuentren dentro de la matriz de aprendizaje.**

# CAPITULO 3 Planteamiento de un algoritmo tipo tipicidad y contraste

## 3.1 Definiciones básicas

Como vimos el algoritmo que se presentó en el capítulo 2 tiene una serie de limitantes que hacen que no se pueda aplicar a muchos de los problemas reales, es por eso que en este capítulo se proponen algunas modificaciones que elimimen las limitantes del mismo haciéndolo más flexible, es decir, permitiendo que se ajuste a las características que presentan los problemas reales, como son el trabajar con diferentes tipos de variables simultáneamente y por consiguiente el establecer criterios de comparación diferentes para cada una de las variables, así como diferentes funciones de semejanza, considerar la importancia informacional de cada una de las variables y algo que es muy importante, eliminar la obligatoria utilización de técnicas estadísticas, como son la construcción de histogramas de frecuencia y por consecuencia la utilización de intervalos de frecuencia. Por otro lado se permitirá la clasificación de objetos que no se encuentren dentro de la matriz de aprendizaje. Para el desarrollo de este algoritmo se hace necesario establecer una serie de conceptos y definiciones nuevas.

Este algoritmo seguirá girando en torno a los conceptos de tipicidad y contraste, sólo que la manera de obtener la medida de la tipicidad para cada uno de los objetos será diferente. La tipicidad estará basada en los pesos informacionales de los rasgos y objetos, así como en el peso diferenciante de los objetos, de los cuales daremos la definición más adelante.

En este algoritmo el concepto de tipicidad está definido en términos de los conceptos de *peso informacional* y *peso diferenciante* de los objetos; los cuales a su vez involucran el término de *peso informacional de los rasgos*; así como de la función de semejanza que se establezca para decir cuándo dos objetos son semejantes o análogos. De una manera rápida diremos que un objeto es típico de una clase determinada si es semejante a los objetos de dicha clase y al mismo tiempo es diferenciante de los objetos de las clases restantes. Por lo tanto mientras más se asemeje a los objetos de una clase y más se diferencie de los objetos de las clases restantes, significará que el objeto es más típico de dicha clase; de manera contraria en la medida en que menos se asemeje a los

objetos de una clase dada y a su vez se asemeje más a los objetos de las clases restantes significará que es menos típico de dicha clase. Como vemos al igual que en el algoritmo anterior la tipicidad puede venir expresada en diferentes gradaciones.

El concepto de contraste seguirá manteniéndose de la misma manera que en el algoritmo anterior , es decir, que será expresado en términos de la tipicidad de los objetos para cada una de las clases, representando de igual manera la medida en que se diferencian las tipicidades de un objeto para cada una de las clases, es decir, que nos dará en forma graduada el contraste de un objeto dependiendo de qué tan típico sea dicho objeto para cada una de las clases.

Como se mencionó en el capítulo anterior existen muchas maneras de definir los conceptos de tipicidad y contraste. En si esto se debe a que la definición de lo que es típico y contrastante es muy subjetiva. En el diccionario encontramos que la definición de típico es: " Simbólico, que incluye en sí la representación de otra cosa siendo emblema de ella. | Que presenta los caracteres distintivos de un tipo.". Sin embargo, el problema estriba en determinar qué es lo representativo de algo o cómo determinar quién presenta los caracteres distintivos de algo. Lo mismo sucede con la definición de contraste. En el diccionario se define el contraste como: " Acción de contrastar, es decir, mostrar condiciones opuestas, dos personas o cosas cuando se comparan. Mostrar notable diferencia con otra cosa.". En este caso el problema es definir qué es " notable diferencia". Por tanto reitero que en este trabajo se utilizará una interpretación específica de tipicidad y contraste que tratan de ajustarse a las definiciones antes mencionadas, pero que no excluyen la posibilidad de encontrar otras formas de expresar dichos conceptos, como es el caso del algoritmo presentado en el capítulo anterior donde la tipicidad era interpretada de una manera distinta a la que se propone en este capítulo.

El plantemiento del problema de clasificación se hará de la misma manera que en la sección 1.2 del capítulo 1. Donde toda la información se representa simbólicamente por medio de la matriz de aprendizaje..

En este caso la diferencia entre el planteamiento que se hizo en la sección 2.1 del capítulo 2 y este es que aquí las variables pueden ser de cualquier tipo y se considera la posibilidad de que exista ausencia de información.

Al principio de esta sección se dijo que el concepto de tipicidad estaba relacionado con el peso informacional de los rasgos y objetos, a continuación daremos los conceptos de peso informacional de los objetos y los rasgos, así como del concepto de peso diferenciante.

Podemos decir que los rasgos que describen a los objetos no proporcionan siempre la misma información en el proceso de clasificación, es decir, que existen variables que resultan más discriminantes que otras. Por ejemplo, el color de la piel de las personas no es tan importante para determinar el tipo de enfermedad que puedan tener, sin embargo, por el contrario la presencia de un determinado síntoma pudiera ser más importante en el proceso de clasificación. Así la variable síntoma x será más importante que la variable color de piel. De esta forma nosotros podemos graduar la importancia de cada uno de los rasgos que describen a los objetos asociándoles a cada uno de ellos una medida que denominaremos *peso informacional del rasgo* $x_i$ y que denotaremos como $p(x_i)$ para i=1, . . ., n. Existen varios enfoques para determinar el peso informacional de los rasgos. Por ejemplo en problemas como los hasta aquí estudiados, en los que los objetos son representados por n-uplos, la importancia de un rasgo pudiera darse por medio de una función monótona de la frecuencia de aparición y del grado en que éste aparece en los objetos estudiados. Esta medida puede ser de utilidad en algunos problemas concretos de Reconocimiento de Patrones, sin embargo, hay otros en los que un rasgo aparece poco, pero cuando aparece determina unívocamente la pertenencia de un objeto a una clase dada. En algunos de los problemas los pesos informacionales de las variables son determinados por los expertos, ya que de alguna manera ellos pueden decidir sobre la base de la experiencia, cuáles son los rasgos que son más importantes en el proceso de clasificación, y dar una gradación de su importancia.

De la misma manera que el peso informacional de los rasgos, en muchos problemas de Reconocimiento de Patrones se hace necesario establecer una diferenciación entre la información proveniente de uno u otro objeto, es decir, de asociarle parámetros externos, magnitudes que nos digan qué tan importante es un objeto. A estas medidas las denominaremos *pesos informacionales de los objetos*. Existen muchas maneras de determinar el peso informacional de los objetos. En este trabajo nos basaremos en la siguiente definición del peso informacional de un objeto.

Sean $p(x_1)$, . . ., $p(x_n)$ los pesos informacionales de los rasgos $x_1$, . . ., $x_n$ respectivamente, $\delta_i : M_i \times M_i \longrightarrow Vi$ el criterio de comparación asociado al rasgo $x_i$. $D_i$ un subconjunto de $V_i$ tal que $\delta_i(O_r, O_v) \in D_i$ significa que $O_r$ y $O_v$ son semejantes atendiendo únicamente al rasgo $x_i$, por lo tanto los criterios de comparación que se utilicen sólo pueden ser de tipo booleano.

DEFINICION 3.1; Llamaremos *peso informacional del objeto* O *con respecto a la clase* $K_t$ a la magnitud

$$PI_j(O) = \frac{1}{\left|K_j^t\right| \sum_{i=1}^{n} p(x_i)} \sum_{i=1}^{n} \alpha_j^i(O)\, p(x_i)$$

siendo $\alpha_j^i(O) = \left|\left\{O_t \in K_j^t \;/\; \delta_i(O, O_t) \in D_i\right\}\right|$

El peso informacional de un objeto O con respecto a una clase será mayor en la medida en que lo sea su semejanza por cada rasgo con los objetos de esa clase en MA; especialmente para aquellos rasgos de mayor relevancia.

Como hemos dicho el peso informacional de un objeto nos proporciona la medida de semejanza por cada rasgo de dicho objeto con respecto a una clase dada, pero qué pasa con las clases restantes, es decir, qué información nos puede proporcionar ese objeto con respecto a las demás clases, a las cuales evidentemente se esperaría que no se pareciera mucho, pero ¿ qué tan cierto es esto?. Tratando de contestar esta pregunta surge el concepto de peso diferenciante de un objeto, con el cual podemos obtener una medida de qué tan no semejante por cada rasgo es un objeto dado con respecto a las clases con la que dicho objeto no fue asociado, es decir, a las clases que son diferentes de las clases que se tomó como base para obtener el peso informacional del objeto.

DEFINICION 3.2 Llamaremos *peso diferenciante del objeto* O *con respecto a la clase* $K_j$ a la magnitud

$$PD_j(O) = \frac{1}{\left|MA - K_j^{'}\right| \sum_{i=1}^{n} \rho(x_i)} \sum_{i=1}^{n} \vartheta_j^{i}(O)\rho(x_i)$$

siendo $\vartheta_j^{i}(O) = \left|\left\{O_i \in (MA - K_j^{'}) \;/\; \delta_i(O,O_i) \in D_i\right\}\right|$

El peso diferenciante de O con respecto a $K_j$ es mayor entonces en la medida en que sea mayor su no-semejanza por rasgo con los objetos que están en las demás clases, teniendo en cuenta los pesos de cada rasgo.

Estas dos magnitudes pretenden caracterizar de alguna manera cuánto O se parece a los objetos que están en $K_j^{'}$ ($PI_j(O)$) y cuánto se diferencia de los que no están en $K_j^{'}$ ($PD_j(O)$). Estas magnitudes pueden calcularse para cada objeto O del universo y pudieran utilizarse directamente para decidir acerca de la pertenencia de un objeto a clasificar, el clasificador que se propone considera únicamente los pesos de los objetos de MA.

De modo que si $O \in K_j^{'}$ le asociaremos dos medidas que son el peso informacional $PI_j(O)$, y el peso diferenciante $PD_j(O)$.

Hasta aquí hemos dado los conceptos básicos para el algoritmo que será descrito en la siguiente sección aunque en realidad falta mencionar los tipos de objetos que forman cada una de las clases. Si recordamos en el capítulo anterior se dijo que los objetos que constituían a cada una de las clases podían estar divididos en cuatro tipos que eran:

- Objetos propios.

- Objetos generales.

- Objetos atípicos

- Objetos no propios

Esta división se hacía dependiendo de las medidas de tipicidad y contraste que presentaran cada uno de los objetos. Pero recordemos también que esta división se hizo para la clasificación de anomalías AGE, que fue la aplicación que se le dió al algoritmo dentro de la Geología. Sin embargo, ahora el algoritmo se presenta de una manera más general ya que ahora los tipos de objetos que tendremos dependerán del problema que se quiera resolver. En el algoritmo simplemente se considera que tenemos s tipos de objetos donde s≥2 los cuales serán definidos durante el proceso de la modelación matemática del problema al igual que las reglas para determinar cuándo un objeto es de un tipo u otro las cuales estarán representadas por $T_{0,s}$, $K_{0,s}$. Aunque en principio diremos que los objetos están definidos dentro de los cuatro tipos mencionados anteriormente y que las reglas $T_{0,s}$, $K_{0,s}$ se obtienen de la misma manera que en el algoritmo presentado en el capítulo 2.

Hecha esta aclaración nos encontramos ya en condiciones de realizar la descripción del algoritmo.

## 3.2 Descripción del algoritmo tipo Tipicidad y Contraste

Paso 1) Determinación de los criterios de comparación, pesos informacionales de los rasgos y función de semejanza.

En este paso se determinan los criterios de comparación $\delta i$ y los pesos informacionales $p(x_i)$ para cada una de las variables, así como la función de semejanza $\beta$ para la comparación entre objetos.

Paso 2) Determinación de los tipos de objetos.

En este paso se determinan los s tipos de objetos, donde s≥2, al igual que las reglas $T_{o,s}$, $K_{o,s}$ para determinar cuándo un objeto es de un tipo u otro. Los tipos de objetos son establecidos durante el proceso de modelación matemática por medio de los expertos. En principio podemos tomar los 4 tipos de objetos determinados para el algoritmo del capítulo 2.

Paso 3) Obtención de pesos informacionales y diferenciantes de los objetos.

En este paso se obtienen los pesos informacionales $PI_j(O)$ y diferenciantes $PD_j(O)$ de cada uno de los objetos con respecto a la clase a la que los mismos pertenecen por medio de las expresiones que se definieron anteriormente. Obteniendose la tabla que se muestra en la fig.1.

| | | PI | PD |
|---|---|---|---|
| | $O_1$ | $PI_1(O_1)$ | $PD_1(O_1)$ |
| $K_1'$ | ⋮ | ⋮ | |
| | $O_s$ | $PI_1(O_s)$ | $PD_1(O_s)$ |
| | ⋮ | ⋮ | ⋮ |
| | $O_t$ | $PI_\ell(O_t)$ | $PD_\ell(O_t)$ |
| $K_\ell'$ | ⋮ | ⋮ | ⋮ |
| | $O_m$ | $PI_\ell(O_m)$ | $PD_\ell(O_m)$ |

fig. 1 Tabla de pesos informacionales y diferenciantes de los objetos de MA

Paso 5) Cálculo de la tipicidad.

En este paso se lleva a cabo el cálculo de la magnitud de la tipicidad T para cada uno de los objetos en cada una de las clases. Por medio de la siguiente expresión, la cual está basada en las medidas de semejanza y diferencia por objeto.

$$T_j(O_h) = \frac{T_{j,1}(O_h) + \sum_{\substack{i=1 \\ i \neq j}}^{n} T_{i,0}(O_h)}{T_{j,0}(O_h) + \sum_{\substack{i=1 \\ i \neq j}}^{n} T_{j,1}(O_h)} \quad \text{para } h=1, \ldots, m$$

De la misma manera que le asociamos a cada objeto un peso informacional y un peso diferenciante basados en la comparación por rasgo, les asociaremos ahora otras magnitudes similares, sólo que éstas estarán basadas en la semejanza por objeto y en el peso informacional y diferenciante de los objetos.

$$T_{j,1}(O_h) = \frac{1}{\left|K'_j\right|} \sum_{\substack{O_i \in K'_j \\ \beta(O_h, O_i) \in D}} PI_j(O_i) \quad \text{para } h=1, \ldots, m$$

$T_{j,1}(O_h)$ es entonces el promedio de los pesos informacionales de los objetos que, estando en $K'_j$, son semejantes a $O_h$ y como $PI_j(O_i)$ es una medida de cuánto $O_i$ se asemeja a los objetos que están en $K'_j$; entonces un valor alto de $T_{j,1}(O_h)$ nos está refiriendo una medida alta de semejanza con objetos de la clase $K_j$.

**Sea**

$$T_{j,0}(O_h) = \frac{1}{|K'_j|} \sum_{\substack{Oi \in K'_j \\ \beta(O_h,Oi) \in D}} PD_j(O_i) \qquad \text{para } h=1, \ldots, m$$

Esta expresión nos da el promedio de los pesos diferenciantes de los objetos que, estando en $K'_j$ no son semejantes a O; y como $PD_j(O_i)$ es una medida de cuánto $O_i$ no se asemeja a los objetos que no están en $K'_j$, entonces un valor alto de $T_{j,0}(O)$ nos está refiriendo una medida alta de no-semejanza con objetos de la clase $K_j$, que no se asemejan a los objetos que no están en $K'_j$.

De modo que podemos considerar $T_{j,1}(O)$ como un valor "a favor" de la pertenencia de O a $K_j$ y $T_{j,0}(O)$ un valor "en contra" de la pertenencia de O a $K_j$.

Como es evidente para cada objeto se calculan $l$ valores de la tipicidad T en correspondencia con la cantidad de clases como se muestra en la figura 3. Este parámetro constituye una medida que muestra en cuánto un objeto dado, es típico entre todos los objetos de la clase $l$.

| | $K'_1$ | ... | $K'_l$ |
|---|---|---|---|
| $O_1$ | $T_1(O_1)$ | ... | $T_l(O_1)$ |
| $O_2$ | $T_1(O_2)$ | ... | $T_l(O_2)$ |
| ⋮ | ... | ... | ... |
| $O_m$ | $T_1(O_m)$ | ... | $T_l(O_m)$ |

fig. 3

El objeto será más típico para aquella clase donde $T_j(O_h)$ es máximo, por tanto el nivel de tipicidad del objeto $T(O_h)$ vendrá dado por:

$$T(O_h) = \max(T_1(O_h), T_2(O_h), \cdots, T_\ell(O_h))$$

Donde h=1, . . ., m lo que implica que para cada objeto se obtendrán r valores de la tipicidad T, donde $r \geq 1$ y $r \leq 1$, ya que el máximo se puede alcanzar en más de una clase. Por lo tanto denotaremos $K_i^p$ a la clase p donde se obtenga la tipicidad máxima.

Paso 5) Cálculo del contraste

En este paso se hará el cálculo de la magnitud del contraste para cada uno de los objetos. Esta magnitud vendrá dada por:

$$C(O_h) = \left[ \frac{\sum_{\substack{j=1 \\ j \neq K_i^p}}^{\ell} (T(O_h) - T_j(O_h))}{\ell - 1} \right]^{\frac{1}{2}}$$

La magnitud $C(O_h)$ constituye una medida del nivel de diferenciación del objeto en relación con el resto de las clases, para las cuales el valor de $T_j(O_h)$ no es máximo. Como vemos para cada objeto se determinará un valor de C ya que h=1, . . ., m.

Paso 6) Clasificación de Objetos de MA en los diferentes tipo.

En este paso se hará la clasificación de los objetos de MA en los diferentes tipos definidos. Como ya mencionamos se cuenta con s tipos de objetos donde $s \geq 2$ y asimismo con s magnitudes $T_{0,s}$, $C_{0,s}$ las cuales nos servirán para decidir de qué tipo es cada uno de los objetos. Con este paso se finaliza el algoritmo.

Hasta el paso 5 podemos hacer un análisis de la estructura interna de la matriz de aprendizaje que nos permite incluso decidir en cuanto a la posible reubicación de los objetos clasificados en MA, ya que existe la posibilidad de que algunos de los objetos de la MA sean más tipicos para una clase distinta a la que los mismos estén asociados.

Paso 7) Clasificación de nuevos objetos.

En este paso se clasifican los objetos que no se encuentran dentro de MA, para lo cual se obtiene la tipicidad con respecto a cada una de las clases que conforman a MA así como el contraste para cada uno de los objetos a ser clasificados.

Los objetos serán clasificados en aquellas clases donde se haya obtenido la tipicidad máxima. y digo clases porque quizá la tipicidad máxima no es única lo que permite la multiclasificación de los objetos.

## 3.3 Resultados básicos

Con el desarrollo de este nuevo algoritmo se ha obtenido una serie de resultados importantes que permitirán que pueda ser aplicado a una gama muy extensa de problemas reales dentro de las diferentes áreas o disciplinas que se mencionaron tanto en la introducción de este trabajo, como en el primer capítulo del mismo. Ya que como vimos dichos problemas presentan ciertas características que hacen que muchos algoritmos de clasificación no puedan ser aplicados a dichos problemas.

Los resultados obtenidos son los siguientes:

- Poder utilizar cualquier tipo de variables para describir a los objetos de la matrz de aprendizaje.

- Dado que se puede utilizar cualquier tipo de variables, podemos también tener diferentes criterios de comparación booleanos para cada una de las variables.

- Poder considerar la ausencia de información. En los criterios de comparación se puede definir qué hacer cuando existe ausencia de información, cosa que no estaba contemplada anteriormente.

- Como se pueden tener diferentes criterios de comparación entre variables también podemos considerar diferentes tipos de funciones de semejanza booleanas. La semejanza no está ligada estrictamente a la igualdad.

- Ahora se puede considerar la importancia informacional de los rasgos y objetos, dando esto ventajas dentro de la clasificación.

- Se pueden clasificar objetos que no se encuentren dentro de la matriz de aprendizaje, es decir, que el algoritmo no es sólo un analizador de datos, sino también un clasificador.

- Podemos tener s diferentes tipos de objetos con $s \geq 2$, permitiendo esto que la cantidad de objetos sea determinada dependiendo de las necesidades del problema.

Es importante mencionar que si bien el algoritmo desarrollado resuelve algunas de las limitantes planteadas al inicio de este trabajo, subsisten algunas que deben ser objeto de trabajos posteriores, como es el considerar que la tipicidad y el contraste puedan ser calculados haciendo un análisis multivariado de los datos y no univariado como se desarrolló en este trabajo, así como el desarrollar también los conceptos de tipicidad y contraste cuando las clases sean difusas, es decir, cuando los objetos pertenezcan a cada una de las clases en cierto grado, ya que en este trabajo sólo se considera que las funciones de semejanza sean de tipo booleano al igual que los criterios de comparación.

# CAPITULO 4 Implantación computacional.

## 4.1 Introducción

Una de las partes más importantes en el desarrollo de un sistema computacional es la selección del lenguaje en el cual se hará la implantación de dicho sistema.

El proceso de selección del lenguaje de implantación puede dividirse en dos puntos importantes[ 18 ] que son: (1) establecer un criterio para la selección del lenguaje (2) determinar cuál o cuáles lenguajes se acoplan de una mejor manera a las características del problema a ser desarrollado computacionalmente, sobre la base del criterio establecido.

Una primera preselección de un lenguaje de implantación es ubicar de una manera rápida aquellos que pudieran acoplarse mejor a las características de nuestro problema y eliminar todos los que no están relacionados con nuestro problema. Por ejemplo, si hablamos del desarrollo computacional de una aplicación de inteligencia artificial, lo más factible sería utilizar lenguajes como PROLOG o LISP, pero si nuestro problema está relacionado con base de datos tendremos que seleccionar lenguajes como DBASE, FOXPLUS, etc. y evidentemente descartaremos como PROLOG y LISP. El descartar los lenguajes que no tienen nada que ver con las características del problema es de gran ayuda ya que de alguna manera se reducen las posibilidades y resulta más fácil hacer la selección. Sin embargo, en muchos casos no podemos hacer esta preselección de lenguajes.

Una vez que se hizo la preselección entonces se establecen los criterios de selección del lenguaje de implantación.

Los criterios para la selección pueden ser determinados considerando los siguientes puntos:

1) Criterio técnico

a) Portabilidad

b) Modularidad

c) Factores humanos

d) Factores técnicos

e) Rapidez

2) Criterio de costo

Uno de los factores más importantes para la seleción del lenguaje de implantación es el criterio técnico, el cual vimos que está dividido en cinco puntos importantes, los cuales explicaremos a continuación.

***Portabilidad***

La portabilidad puede ser definida fácilmente como la capacidad de mover un programa de un ambiente a otro haciendo ninguna o pocas modificaciones. Esta es una característica muy importante, ya que hace que los programas o sistemas que se desarrollan puedan ser utilizados en diferentes equipos y medios ambientes permitiendo esto una mayor difusión de los mismos.

***Modularidad***

La modularidad es la habilidad de dividir un problema en pequeños módulos o subproblemas que resultan más fáciles de resolver en comparación del todo. Algunos lenguajes presentan ventajas en este punto ya que sus módulos pueden ser compilados de manera individual permitiendo esto detectar los errores que se pudieran generar en cada

uno de los módulos de manera más rápida, sin tener que verificar el programa completo. Siempre y cuando el lenguaje permita compilarlos en forma individual.

Es importante mencionar que el desarrollo de un sistema en módulos aparte de hacer legible un programa y además detectar los errores de una manera rápida nos permite dar la posibilidad de extender el sistema a futuro, es decir, de agregar más módulos haciendo modificaciones mínimas al programa principal. Esto incluso nos lleva a pensar en la utilización de módulos de otros programas que pudieran servir a nuestro sistema sin la necesidad de volver a programarlos, es decir, hacer reutilización de código.

Existen algunos lenguajes que fueron diseñados con la filosofía de trabajar de manera estructurada o modular por lo tanto se considera que la modularidad sí es un factor importante para el desarrollo de un sistema, sobre todo si se contempla el posible crecimiento del sistema a futuro.

*Factores humanos*

Otros de los factores que quizá no sean tan importantes, dado que son factores un poco subjetivos, son los factores humanos los cuales incluyen la calidad de la documentación y la facilidad de utilización del lenguaje. Si la documentación que proporciona el lenguaje (tanto en manuales de usuario como en la ayuda en línea) es muy densa, es decir, no es fácil de entender y no está organizada de una manera que se pueda localizar rápidamente lo que se busca, entonces dicha documentación resultará inusitada.

La facilidad de utilización del lenguaje se determina cuando un lenguaje es capaz de proporcionarnos un medio ambiente de programación integrado de compilación y edición.

*Factores técnicos*

Los factores técnicos son uno de los criterios más importantes para la evaluación de un lenguaje. Dentro del desarrollo de un sistema encontramos una etapa de análisis en

el cual se determinan los tipos de datos de entrada y salida del sistema, así como las estructuras y archivos que serán utilizados, es por eso que en la selección del lenguaje de implantación se debe elegir aquel o aquellos que nos de las máximas facilidades con respecto a los tipos de datos, las estructuras y archivos que serán utilizados.

Existen algunos lenguajes que proporcionan más facilidades para el manejo de estructuras como son: listas, árboles, pilas, etc. así como el manejo de registros, archivos, conjuntos, etc. lo que nos permite desarrollar aplicaciones más complejas.

### *Rapidez*

Finalmente llegamos al punto quizá menos importante para la evaluación de un lenguaje que es la rapidez. Este es un punto muy difícil de evaluar ya que no sólo depende del lenguaje que utilicemos, sino que también depende de muchos otros factores, uno de los cuales es el mismo programador ya que si el programador hace manejos innecesarios de algunos recursos como son el acceso a disco, a la memoria o al sistema por ejemplo, esto hará que el programa se vuelva un poco lento, por éstas razones en general la rapidez de un lenguaje no puede ser evaluada tan fácilmente ya que involucra de alguna manera cierto subjetivismo.

El criterio técnico para la evaluación de lenguaje es muy importante sin embargo existe otro criterio para la evaluación que es el criterio de costo.

Se pueden distinguir tres tipos de costos dentro de un proyecto de programación. El primero es el costo del desarrollo del sistema. El segundo es el costo de operación del sistema y el tercero es el mantenimiento durante el ciclo de vida del sistema.

La idea de costos puede resultar obvia cuando se habla del desarrollo de un sistema comercial, sin embargo, cuando hablamos de un sistema que no es comercial no podemos imaginarnos cómo determinar el costo.

Todos los sistemas tienen asociado un costo. En el caso de los sistemas comerciales los costos son: el tiempo de computadora, tiempo de programación, y muchos

más. Por otra parte en los que no son de tipo comercial, como por ejemplo los que son desarrollados por los estudiantes en los cursos académicos, los costos no son tan obvios, sin embargo existen. Cuando desarrollamos un sistema no comercial los costos son por ejemplo, el tiempo que se invierta en la realización del mismo ya que mientras menos tiempo utilicemos en el desarrollo tendremos más tiempo para hacer otros programas o actividades. Quizá en este caso los costos no son de tipo monetario pero eso no implica que los costos de tiempo no deban ser considerados.

Como vemos para seleccionar el lenguaje de implantación debemos tener en cuenta los puntos que fueron descritos anteriormente pero algo que es quizá más importante antes de establecer los criterios para seleccionar el lenguaje es el analizar y valorar los datos de entrada y sobre todo determinar que es lo que se espera como resultado.

Una vez que se establecieron los criterios de selección lo único que resta es verificar cuál es el lenguaje o lenguajes que cumplen con los mismos y utilizar el que mejor se acople a nuestras necesidades.

En el caso de la implantación computacional del algoritmo para la clasificación y análisis de datos tipo tipicidad y contraste, que fue presentado en el capítulo 3 se seleccionó como lenguaje de programación el lenguaje C. Las razones por las cuales fue seleccionado dicho lenguaje son las siguientes:

1) Las facilidades que proporciona para el manejo de estructuras como son listas, pilas, colas, etc., así como para el manejo de archivos. En particular en este trabajo se hace uso de listas circulares y de un descriptor de archivos.

2) Las ventajas que proporciona, ya que es un lenguaje estructurado, lo que permite planificar de una manera modular el sistema permitiendo dividirlo en pequeños problemas y a su vez considerando extensiones a futuro de nuevos modulos sin tener que hacer grandes modificaciones.

3) Gracias a las ventajas que se mencionaron en el punto 2) se pudo reutilizar algunas funciones que fueron programadas para otro proyecto sin tener que hacer grandes modificaciones, es decir, que se hizo reutilización de código.

4) El punto más importante en la elección del lenguaje de programación es que este programa será agregado como un módulo más a un sistema ya establecido que está escrito en C. Sin embargo, si las características del lenguaje C no se hubieran acoplado a las necesidades del problema se hubiera tenido que escoger otro lenguaje que se acoplara a dichas necesidades.

Otro de los factores que quizá no tenga mucha importancia pero que también fue considerado para la elección del lenguaje es la cantidad de bibliografía con la que se cuenta así como la ayuda en línea que proporciona el lenguaje.

Finalmente podemos decir que el sistema fue realizado para trabajar en una microcomputadora bajo sistema operativo MS-DOS utilizando el compilador de Turbo C++ 2.0 de Borland. Sin embargo, otras de las ventajas que proporciona el trabajar en lenguaje C es la portabilidad, por lo tanto este sistema pudiese llevarse a otros ambientes como es el ambiente UNIX sin muchas modificaciones, lo que permitiría su mayor utilización.

## 4.2 Estructuras y funciones

Fundamentalmente podemos decir que el sistema en su totalidad está conformado por dos partes importantes. La primera es un modulo de lectura y validación de datos y la segunda es el modulo que ejecuta el algoritmo de tipicidad y contraste. Cada uno de estos modulos a su vez está constituido por una serie de procedimientos.

En los siguientes parrafos daremos una explicasión general de cada modulo y de las funciones más importantes de ellos.

Analicemos el modulo de lectura. En primer lugar diremos que los datos que se requieren para ejecutar el algoritmo de tipicidad y contraste son en esencia los que corresponden a la matriz de aprendizaje MA, los cuales se enumeran a continuación.

1) Número de clases

2) Número de objetos

3) Número de atributos

4) Tipo de cada una de las variables

5) Valencia de las varaiables (Cuando son k-valentes)

6) Pesos informacionales de las variables

7) Criterios de comparación de las variables

8) Descripción estandar de cada uno de los objetos.

9) Función de semejanza

Toda esta información es leida de un descriptor de archivos y de un archivo de datos. Es importante mencionar que dentro del sistema no se crean ni el descriptor de archivos ni el archivo de datos. Como ya se explico anteriormente este programa formará parte de un sistema ya establecido por lo tanto se considera que dichos archivos fueron creados con anterioridad.

Los archivos de datos contienen las descripciones de cada uno de los objetos de la matriz de aprendizaje. Cada renglón del archivo corresponde a un objéto de la matriz de aprendizaje. El primer dato por renglón representa la clase a la que pertenece el objeto y

los datos restantes corresponden a los valores que toma el objeto en cada una de las variables.

Por otra parte el archivo de descriptores contiene las características más importantes de cada uno de los archivos de datos. Cada uno de los renglones (entendiendose que un renglón termina hasta donde se encuentra un retorno de carro) del descriptor de archivos contiene la siguiente información. En primer lugar aparece el nombre del archivo de datos, después el número de clases, los objetos que contiene la matriz de aprendizaje que se encuentra en dicho archivo, el codigo de la función de semejanza, en seguida el número de variables y después aparece el nombre de la primer variable seguido del tipo de variable, en el caso de ser k-valentes aparece la valencia, el criterio de comparación, en caso de ser modulo la diferencia aparece el épsilon y por último el peso informacional, después aparece el nombre de la segunda variable con sus respectivos datos y así hasta la variable n.

Una vez que hemos establecido el contenido del descriptor de archivos como el del archivo de datos especificaremos como funciona el modulo de lectura.

En primer lugar se verifica que el archivo de descriptores exista y que no este vacío. Este proceso se lleva a cabo por medio de la función LEE_ARCH_DESC. Una vez que verifico que exite el arhivo y que no esta vacío se empieza a leer dicho archivo verificando que los datos del mismo sean correctos es decir que no existan errores de sintaxis. Este proceso se lleva a cabo por medio de la función OBTEN_TOKEN la cual va verificando caracter a caracter el contenido del archivo, el cual debe tener la sintaxis que se describio en parrafos anteriores. Una vez que se verifico que los datos que se leyeron correspondientes al número de objetos y número de atributos estan correctos, empieza a leer la información correspondiente a las variables creando en ese momento una lista circular doblemente ligada donde en cada nodo se guardara la información de dichas variables como se muestra en la figura 1. La función que se encarga de llenar la información de cada uno de los nodos de la lista es LEE_RASGO_ARCH. Esta información corresponde a los siguientes puntos:

1) Nombre de la variable

2) Número

3) Tipo

4) Valencia

5) Criterio de comparación

6) Epsilon

7) Peso informacional

Una vez que se leyó la última variable, el último nodo de la lista se conecta con el primer nodo de la misma.

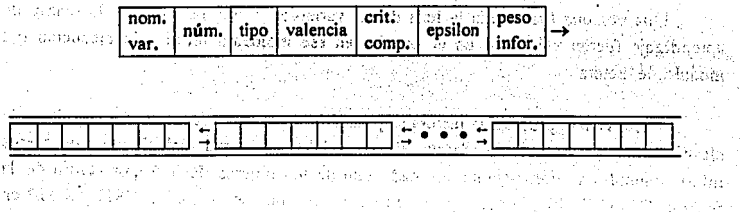



fig 1 Lista circular doblemente ligada correspondiente a las varibles.

Después de crear la lista de atributos se verifica que el archivo de datos no este vacío ya que se empezarán a leer los datos del mismo. Este proceso lo realiza la función LECT. Si el archivo no está vacío el proceso de lectura de datos comienza por medio de la función LEE_MA_ARCH. Esta función se encarga de ir leyendo los datos del archivo.

y ponerlos en un arreglo como se muestra en la figura 2. Los datos son metidos al arreglo de la manera en que estan organizados en el archivo, es decir, el primer elemento es el número de la clase a la que pertenece el primer objeto, seguido de esto se colocarán los valores que toman cada una de las variables del objeto, verificando al mismo tiempo que dichos valores correspondan al tipo de la variable, después se coloca el número de la clase del segundo objeto y sus correspondientes valores de las variables y así sucesivamente se hace este proceso con cada uno de los objetos que se encuentran en el archivo de datos.


fig 2 Arreglo de almacenamiento de los valores de MA.

Una vez que fue creada la lista de las variables y que los datos de la matriz de aprendizaje fueron colocados en el arreglo, en ese momento termina la ejecución del modulo de lectura.

Terminado el proceso de lectura se pasa al modulo de la ejecución del algoritmo de clasificación y análisis de datos. Este modulo comienza cálculando los pesos informacionales y diferenciantes de cada uno de los objetos de MA por medio de la función CALC_PI_PD, la cual contiene una función que se denomina CRIT_COMP en donde se verifica si los valores de un par de objetos correspondientes a una misma variable son semejantes, con respecto a un cierto criterio de comparación. Asimismo se verifica que el criterio de comparación corresponda al tipo de variable.

Los pesos informacionales y diferenciantes son guardados dentro de un archivo de datos. Pero al mismo tiempo son guardados también en un arreglo como el que se muestra a continuación en la figura 3.

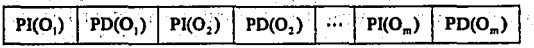



fig 3 Arreglo de pesos informacionales y diferenciantes de los objetos de MA.

Después de calcular tanto los pesos informacionales como los diferenciantes, se lleva a cabo el cálculo de la tipicidad de cada uno de los objetos para cada una de las clases por medio de la función TIPICIDAD, dentro de la cual encontramos otra función que se llama FUN_SEM, que verifica si las descripciones de un par de objetos son semejantes con respecto a una cierta función de semejanza.

Los resultados obtenidos al ejecutar la función TIPICIDAD son guardados en un archivo y al mismo tiempo en un arreglo como el que se muestra en la figura 4. Al mismo tiempo se obtiene la tipicidad máxima del objeto y se guarda al final de las tipicidades por clase para cada objeto.

El siguiente paso dentro del algoritmo es calcular la medida de contraste de cada uno de los objetos. Este proceso se lleva a cabo por medio de la función CONTRASTE que tiene como parametro auxiliar la tipicidad máxima de cada objeto. Los resultados del contraste de cada uno de los objetos son guardados en el arreglo correpondiente a las tipicidades que se genero con anterioridad y que se muestra en la siguiente figura.

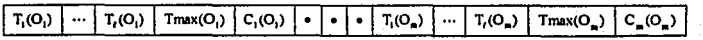



fig 4 Arreglo de tipicidades y contrastes para cada uno de los objetos.

Ya que fue calculada la tipicidad y el contraste de cada uno de los objetos se pasa al proceso de clasificación o de análisis de datos dependiendo de lo que se quiera hacer. Si el objetivo es analizar únicamente los objetos de MA entonces se aplica la función ANALI_OBJ la cual se encarga de determinar de que tipo son cada uno de los objetos y que tiene como función auxiliar REG_SOL que se encarga de calcular los parametros de comparación $T_{0,s}$, $C_{0,s}$ que forman parte de la regla de solución para definr los tipos de objetos. Sin embargo si se desea clasificar objetos que no se encuentren dentro de MA entonces se hace llamado a la función CLASIFICA la cual se encarga de verificar en que clase o clases se deben clasificar los objetos, los cuales son leidos de un archivo de datos. Los resultados tanto del análisis de datos como de la clasificación son guardados en archivos de datos respectivamente.

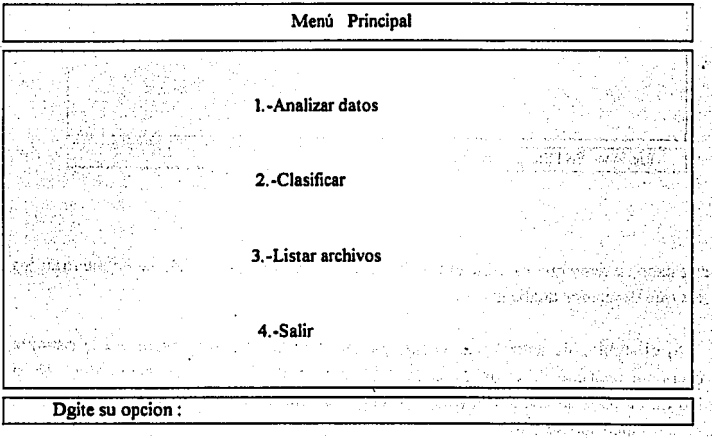
Es importante mencionar que las funciones definidas en esta sección no son todas, sin embargo, son las más importantes.

## 4.3 Manejo del sistema

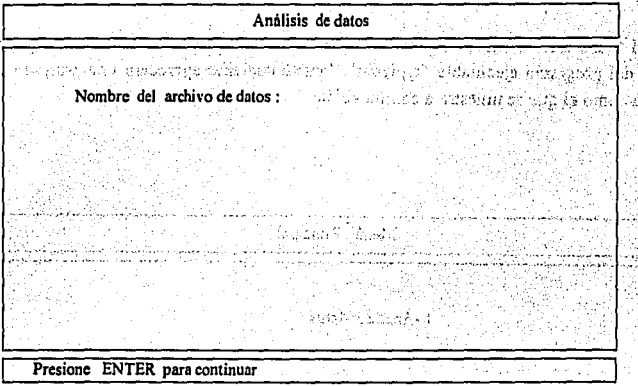
Para ejecutar el programa de clasificación y análisis de datos en primer lugar se debe de verificar que existan los siguientes archivos en la misma unidad de disco.

- tipycont.exe
- defins.h
- lectura.c
- algoritm.c
- interfac.c
- descrip.dat
- archivo(s) de datos

Si todos estos archivos se encuentran en el directorio entonces se debe digitar el nombre del programa ejecutable "typicont". Inmediatamente aparecera una pantalla con un menú como el que se muestra a continuación.




Si la opción que se selecciona es la número 1 entonces aparecera una nueva pantalla como la que se muestra a continuación.




En este caso se tiene que escribir el nombre del archivo de datos donde se encuentran los objetos que deseamos analizar.

Si el archivo de datos no se encuentra, entonces en la parte inferior de la pantalla aparecera un mensaje de error diciendo " El archivo de datos no se encuentra". Si el archivo de datos si se encuentra pero los datos que contiene no son correctos aparecera entonces un mensaje de error.

Los archivos de datos deben de tener la sintaxis siguiente:

Clase del objeto $O_1$ Valor de $x_1$ para $O_1$ . . . . . Valor de $x_n$ para $O_1$

Clase del objeto $O_2$ Valor de $x_1$ para $O_2$ . . . . Valor de $x_n$ para $O_2$

.

.

Clase del objeto $O_m$ Valor de $x_1$ para $O_m$ . . . . . Valor de $x_n$ para $O_m$

Como podemos observar cada uno de los renglones del archivo corresponde a la descripción estandar de un objeto donde el primer elemento del renglón representa la clase a la que pertenece el objeto. Es importante mencionar que las descripciones estandar de los objetos se finalizan con un ENTER.

Si no existen errores de sintaxis el proceso de análisis de datos inicia. Durante el proceso se pueden generar una serie de errores como son:

- El criterio de comparación no corresponde al tipo de variable

- El epsilon del criterio de comparación no corresponde al mismo

- Los pesos informacionales no son numericos

- Los valores de una variable k-valente no corresponden a la valencia.

Si algun error de los descritos anteriormente fuera detectado aprecera un mensaje de error en la parte inferior de la pantalla.

Una vez que se realizan todas las operaciones necesarias aperecera en la pantalla los resultados del análisis, dandonos el tipo de cada uno de los objetos. Si los objetos exceden al tamaño de la pantalla se iran mostrando por bloques presionando la tecla ENTER entre cada bloque hasta terminar de listar todos los resultados.

Si elegimos la opción número 2 del menú principal aparecera enseguida una pantalla como la que se muestra en la figura 5. En la cual se pide el nombre del archivo de datos donde se localizan las descripciones de los objetos correspondientes a la matriz de aprendizaje. Asimismo se pide el archivo de los objetos nuevos a ser clasificados.

Si el archivo de datos de la matriz de aprendizaje ya fue alguna vez clasificado y se cuenta con los pesos informacionales y diferenciantes de dicho archivo entonces el proceso de clasificación se inicia a partir del cálculo de la tipiciada y el contraste tanto para los objetos de la matriz como para los objetos a ser clasificados. Finalmente apareceran los resultados de la clasificación en la pantalla. Al igual que en el análisis de datos si los resultados de los objetos no caben dentro de la pantalla estos serán listados por bloques.

En el caso de que la matriz de aprendizaje no haya sido analizada con anterioridad el proceso de clasificación inicia desde el análisis de sintaxis del los archivos de datos y del archivo de descriptores. Si existiera algun error dentro de los archivos entonces aparecera un mensaje de error en la parte inferior de la pantalla de lo contrario serán desplegados los resultados

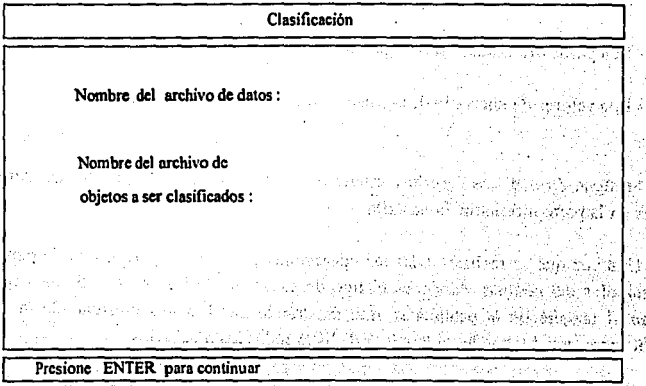



fig. 5 Menú de clasificación

Si elegimos la opción número 3 del menú principal obtendremos el listado de los archivos que conforman al descriptor de archivos, es decir, podremos ver todos los archivos de datos existentes. La información que se desplegará por cada uno de los archivos es la siguiente:

- Nombre del archivo de datos
- Número de clases
- Número de objetos que contiene
- Función de semejanza
- Número de variables que contiene
- Nombre de cada una de las variables
- Tipo de cada una de las variables
- Valencia en caso de ser variable k-valente
- Criterio de comparación de cada variable
- Valor del epsilon del criterio de comparación
- Peso informacional de cada variable

Es importante mencionar que en las tres primeras opciones del menú principal una vez que han sido ejecutadas se regresa nuevamente al menú principal.

Por último diremos que si se elige la opción número 4 del menú principal se dara por terminado la ejecución del sistema y regresaremos al sistema operativo.

# CONCLUSIONES

Tomando en consideración que el objetivo y motivación de este trabajo, planteado al inicio del mismo, era el desarrollar un algoritmo de clasificación y análisis de datos que se acoplara de una manera más adecuada a las características de los problemas de las diferentes áreas que están relacionadas con el Reconocimiento de Patrones podemos decir que el objetivo propuesto se ha cumplido de manera satisfactoria en los siguientes puntos:

1.- Las variables que describen a los objetos pueden ser de cualquier tipo, como por ejemplo, numéricas, bivalentes e incluso difusas.

2.- Se considera la posibilidad de trabajar con ausencia de parte de la información acerca de los objetos estudiados.

3.- El criterio de comparación para cada una de las variables pueden ser diferentes, si es necesario. Todos los criterios de comparación deben ser de tipo booleano.

4.- Las funciones de semejanza que se utilizan para hacer las comparaciones entre objetos no necesariamente tiene que ser la igualdad estricta. Se pueden utilizar diferentes funciones de semejanza siempre y cuando éstas sean de tipo booleano.

5.- Se considera que no todas las variables proporcionan la misma información, por lo tanto se hace una ponderación apriori de cada una de ellas dependiendo de qué tan importantes sean para la clasificación.

6.- Se pueden clasificar objetos nuevos, entendiéndose como nuevos aquellos que no se encuentran dentro de la matriz de aprendizaje.

7.- Las clases en que está dividida la matriz de aprendizaje no necesariamente tienen que ser disjuntas.

8.- Los tipos de objetos pueden ser desde 2 hasta la cantidad que se desee

Además de los resultados teóricos obtenidos se realizó la implantació computacional del algoritmo.

Como vemos el algoritmo desarrollado toma en cuenta algunas de la características que presentan los problemas reales, sin embargo, quedan otra consideraciones o características que no son contempladas en este algoritmo, pero qu son motivo del desarrollo de trabajos posteriores.

## BIBLIOGRAFIA


1.- Barkakati, N.
"Turbo C Bible"
SAMS, 1991.

2.- Chapa, S.
" Herramientas para Consultas y Capturas basadas en el descriptor de archivos"
Informe Técnico; Serie Amarilla. CINVESTAV, I.P.N; 1985.

3.- Devore, J
"Probability and Statistics for Engineering and the sciences"
Brooks/Cole Publishing Co. 1991

4.- Diordenko, L.; Vaskovsvskii B.
"Algoritmo de clasificación para la determinación de anomalías AGE"
Problemy Kibernetiki Rev.,23, pp.131-145 (En Ruso)

5.- Duda, R.; Hart, P
"Pattern Classification and scene analysis"
Wiley - Interscience Publication. 1973

6.- Fukunaga, K.
"Introduction to Statical Pattern Recognition"
Academic Press, Inc. Second Edition 1990

7.- Harry, C.
"Introduction to Mathematical Techniques in Pattern Recognition"
Wiley - Interscience Publication. 1972

8.- Kernighan, B; Ritchie, D.

"El lenguaje de Programación C"
Printece-Hall Hispanoamericana, S.A., 1985.

9.- Lazo, M.
"Modelos basados en la teoría de testores para la selección de rasgos y la clasificación supervisada con descripciones no clásicas de los objetos"
Tesis doctoral de la Universidad de las Villas, Cuba. 1994

10.- Rosenblatt, F.
"The perceptron: A Probabilistic Model for Information Storage and Organization in the Brain"
Psychological Rev., 65, No. 6, pp. 386-408; 1958.

11.- Ruiz, J.
"Modelo de Algoritmos de Reconocimiento con Aprendizaje Parcial"
Memorias del 3er. Congreso Iberoamericano de Inteligencia Artificial
Grupo Noriega Editores, 1992.

12.- Ruiz, J.; Fuentes, A.
"Un modelo cibernético de la delincuencia juvenil"
Ciencias Matemáticas Rev.,II, pp.101-153, 1981.

13.- Ruiz, J; Lazo, M
"Modelos Matemáticos para el Reconocimiento de Patrones"
Aportaciones Matemática. Serie textos IMATE - UNAM (en prensa)

14.- Shalkoff, R.
"Pattern Recognition: Statical, Structural and Neural Approaches"
Wiley & Sons, Inc. 1977

15.- Sklansky, Jack
"Pattern Recognition: Introductions and Fundations"
Dowden, Hutchinson & Ross, Inc.; 1973.

16.- Spiegel, M
"Estadística"
McGraw-Hill. 1970.

17.- Tou, J.; Gonzalez, R.
"Pattern Recognition Principles"
Addison Wesley, Mass. 1974.

18.- Ward, T.
"Applied Programming Techniques in C"
Scott, Foresman and Company, 1982.

19.- Watanabe, S.
"Pattern Recognition: Human and Mechanical"
Wiley & Sons, Inc. 1985

## APENDICE

En esta sección se presentan los listados de la implantación computacional del algoritmo de clasificación y análisis de datos, que como ya se menciono fue realizada en lenguaje C.

Los listados están divididos en cuatro partes, que son:

A1.- Programa principal y definiciones

A2.- Rutinas de lectura de datos

A3.- Rutinas del Algoritmo

A4.- Rutinas de Interface

A5.- Ejemplos de Archivos de Datos.

Dentro de la sección A1 se encuentra el programa principal llamado **tipycont.c**, así como, las definiciones de la variables y estructuras que se utilizan en el mismo y que se encuentran en el archivo **defins.h.**

En la sección A2 se localizan todas las rutinas para la lectura de los datos de entrada del programa, las cuales se encuentran en el archivo **lectura.c.**

La sección A3 contiene todas las rutinas con las cuales se llevan a cabo los pasos del algoritmo de análisis de datos y clasificación. Estas se localizan en el archivo **algorit.c.**

La sección A4 tiene las rutinas que forman parte de la interface con el usuario y se encuentran en el archivo **interfac.c.**

Finalmente la sección A5 tiene un ejemplo de la estructura del descriptor de arcivos y de un archivo de datos.

## A1.- Programa principal y definiciones

```
/*********************  defins.h  ******************/
/* Este archivo contiene todas las definiciones de las va-    */
/* riables que seran utilizadas en el rpograma principal,     */
/* asi como la estructura en la cual se guardara la infor-    */
/* macion correspondiente a las variables. Dicha estructu-    */
/* ra representa una lista doblemente ligada.                 */
/************************************************************/
struct rasgos{
            int num_rasgo;
            char nom_rasgo[25];
            char tip;
            char k;
            char crit_comp;
            double epsilon;
            double peso_inf_rasgo;
            struct rasgos *ant;
            struct rasgos *sig;
            };

typedef char CADENA[10];
typedef char CADENA2[12];
typedef struct rasgos NODO_RASGO;
NODO_RASGO *pri_nom_rasg;
CADENA *MA;
double *mat;
double *mat_tip_cont;
char nom_arch[25];
char nom_des[30];
char nom_archivo1[12];
char nom_archivo2[12];
char nom_archivo3[12];
char token[25];
char buffer[10];
int token_num,indice;
int tot_obj;
int rasgonum;
int num_clase;
FILE *entrada;
int longi;

/*********************  PROGRAMA PRINCIPAL  *********************/
/* Por medio del programa principal se pueden ejecutar las opciones   */
/* de analisis de datos, clasificacion y listado de archivos.         */
/* este programa incluye los codigos de interface.c, lectura.c        */
/* algorit.c.                                                         */
/*                                                                    */
/*      Entrada: Ninguna                                              */
/*   Salida: Ninguna                                                  */
/*   Funciones Auxiliares:MENU_PRIN,LECTURA_DATOS,CAL_PI_PD,          */
/*                CONTRASTE,GUARDA_ARCH,LEE_DATOS_CLA               */
/*                LEE_ARCH_TO_MAT,CLASIFICA,LISTA_DESC              */
/*   Programas Auxiliares:interface.c, lectura.c, algorit.c           */
/*   Archivos Auxiliares: defins.h                                    */
```

```
/*****************************************************************/
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <stdlib.h>
#include <alloc.h>
#include <dos.h>
#include "defins.h"
#include "INTERFAC.C"
#include "LECTURA"
#include "ALGORIT.C"

void main(){
char  res,lec;
int resp;
char nom_arch3[12];
char ext1[4]=".TYC";
char nom_arch4[12];

  fclose(entrada);
  resp = 0;
  do
  {
    MENU_PRIN();
    flushall();
    scanf("%d",&resp);
    switch(resp)
    {
          case 1: LECTURA_DATOS();
                  CAL_PI_PD(MA,pri_nom_rasg);
                  TIPICIDAD(MA,mat,MA,tot_obj,pri_nom_rasg);
                  CONTRASTE(mat_tip_cont,tot_obj);
                  strcpy(nom_arch3, BUSCA_NOM_ARCH(nom_arch,ext1));
                  GUARDA_ARCH(mat_tip_cont,tot_obj,num_clase,nom_arch3);
                  free(MA);
                  free(mat);
                  free(mat_tip_cont);
                  resp=0;
                  break;

          case 2: LECTURA_DATOS();
                  if(LEE_DATOS_CLA())
                  {
                    LEE_ARCH_TO_MAT(nom_arch4);
                    TIPICIDAD(MA, mat,mat_obj_cla,nom_obj_clas);
                  }
                  else
                  {
                    CALC_PI_PD(MA,prinom_rasg);
                    TIPICIDAD((MA, mat,mat_obj_cla,nom_obj_clas);
                  }
                  CLASIFICA(mat_tip_con);
                  free(MA);
                  free(mat);
                  free(mat_tip_cont);
```

```
                free(mat_obj_cla);
                resp=0;
                break;
        case 3: LISTA_DESC();
                flushall();
                fclose(entrada);
                resp=0;
                break;

        case 4: break;

        default: resp = 0;
                 break;
    }
  }while(resp < 4);
}
```

## A2.- Rutinas de lectura de datos

```
/*********************  Función OBTEN_TOKEN()  **********/
/* Esta función se engarga de analizar de que tipo son los caracteres     */
/* que va obteniendo de un archivo, clasificaodolos en 5 tipos a saber:   */
/* 1) si el caracter(es) es un digito , 2) si el caracter inicial es      */
/* una letra 3)si el caracter es un punto 4) si el caracter es un  y      */
/* 5) si el caracter es un retorno de carro \n.                           */
/*                                                                        */
/*        Entrada: Ninguna.                                               */
/*    Salida: El número correspondiente al caracter leido.                */
/*        Funciones auxiliares: Ninguna.                                  */
/**************************************************************/
int OBTEN_TOKEN()
{
char carac;
int aux, indice =1;

  while( (carac = getc(entrada) ) == ' ' || carac == '\t')
                ;

  if(carac >= '0' && carac <= '9')
  {
    aux = 1;
    token[0] = carac;
    while( (carac = getc(entrada)) >= '0' && carac <= '9' || carac == '.')
          token[indice++] = carac;
    token[indice] = '\0';
    ungetc(carac, entrada);
  }
  else
    if( (carac >= 'A' && carac <= 'Z') || (carac >= 'a' && carac <= 'z') )
    {
          aux = 2;
          token[0] = carac;
          while( (carac = getc(entrada) ) != ' ' && carac != '\t' && carac != '\n'&& !feof(entrada))
            token[indice++] = carac;
```

```
            token[indice] = '\0';
            ungetc(carac, entrada);
        }
        else
            if( carac == '.')
              aux = 3;
            else
              if(carac == '=')
              {
                aux = 4;
                token[0] = carac;
                token[1] = '\0';
              }
              else
                if(carac == '\n')
                      aux = 5;
                else
                      aux = 0;
    return(aux);

}

/*********************   Función LECT()   ************/
/* Esta función se encarga de verificar si el archivo de datos     */
/* esta  vacío.                                                    */
/*          Entrada: Ninguna.                                      */
/*     Salida: 1 si el archivo de datos no esta vacío, 0 en        */
/*             caso contrario                                      */
/*          Funciones auxiliares: Ninguna.                         */
/*****************************************************************/
int LECT(void)
{
char micar;

  if( (entrada = fopen(nom_arch, "r") ) == 0)
  {
    ERROR("No se puede abrir  el archivo: ", nom_arch, 0);
    return(0);
  }
  else
  {
    micar = getc(entrada);
    if(!feof(entrada) )
    {
          ungetc(micar, entrada);
          return(1);
    }
    else
    {
          ERROR("El archivo est  vacío", "  ", 0);
          return(0);
    }
  }
}

/*********************   Función LEE_RASGO_ARCH()   *********/
```

```
/* Esta función se encarga de llenar un nodo de la lista de variables        */
/* con la información correspondiente,  la cual lee del archivo de datos.    */
/* Si alguno de los datos esta mal en el archivo, se manda un mensaje de     */
/* error.                                                                    */
/*                                                                           */
/*        Entrada: El apuntador al nodo que se va a llenar                   */
/*    Salida: 1 si el los datos del archivo fueron correctos, 0 en caso      */
/*            contrario                                                      */
/*        Funciones auxiliares: OBTEN_TOKEN,ERROR,REAL                       */
/*****************************************************************************/
int LEE_RASGO_ARCH(NODO_RASGO *auxrasgo, int i)
{

  strcpy(auxrasgo->nom_rasgo, token);
  auxrasgo->num_rasgo = i+1;
  if(OBTEN_TOKEN() == 2)
  {
    auxrasgo->tip = token[0];
    if(token[0] == 'k')
          if(OBTEN_TOKEN() == 1)
            auxrasgo->k = token[0];
          else
          {
            ERROR("El valor de k no es un valor nÚmerico en el rasgo:"," ",i+1);
            return(0);
          }
    else
          auxrasgo->k = '1';
    if( OBTEN_TOKEN() == 1)
          if(token[0] == '1')
          {
            auxrasgo->crit_comp=token[0];
            if( (OBTEN_TOKEN() == 1) && (auxrasgo->tip != 'b') && (REAL(token)) )
              auxrasgo->epsilon = atof(token);
            else
            {
              ERROR("Hay un error en el epsilon del criterio de comparación del rasgo"," ",i+1);
              return(0);
            }
          }
          else
            if(token[0] == '2')
            {
              auxrasgo->crit_comp=token[0];
              auxrasgo->epsilon=0.0;
            }
            else
            {
              ERROR("Error en criterio de comparación del rasgo"," ",i+1);
              return(0);
            }
    else
    {
          ERROR("Error en el criterio de comparación del rasgo"," ",i+1);
          return(0);
    }
```

```
    if( (OBTEN_TOKEN() == 1) && (REAL(token)) )
          auxrasgo->peso_inf_rasgo = atof(token);
    else
    {
          ERROR("Hay un error en el peso informacional del rasgo:"," ",i+1);
          return(0);
    }
    return(1);
  }
  else
  {
    ERROR("Hay un error en el tipo del rasgo:"," ",i+1);
    return(0);
  }
}

/*********************  Función LEE_ARCH_DESC()  ******************/
/* Esta función se encarga de llenar la lista completa de variables,          */
/* verificando antes si el archivo de descriptores existe, ya que de este     */
/* obtendra la información para llenar los nodos de la lista                  */
/*                                                                            */
/*        Entrada: Ninguna                                                    */
/*    Salida: El apuntador al primer nodo de la lista                         */
/*        Funciones auxiliares: OBTEN_TOKEN,ERROR,LEE_RASGO_ARCH              */
/******************************************************************************/
NODO_RASGO* LEE_ARCH_DESC()
{

int i;
char mira;
NODO_RASGO *rasgo, *auxrasgo, *otro;

  gotoxy(6,23); printf("Presione ENTER para continuar");
  gotoxy(6,7);
  printf("Nombre del archivo:");
  scanf("%s",nom_arch);
  strcpy(nom_des, "a:descrip.dat");
  if( (entrada = fopen(nom_des, "r+") ) == NULL)
  {
    ERROR("No se puede abrir el descriptor de archivos"," ",0);
    exit(1);
  }
  while(!feof(entrada) && OBTEN_TOKEN() == 2)
  {
    if(strcmp(token, nom_arch) == 0)
    {
          if(OBTEN_TOKEN() == 1)
            tot_obj = atoi(token);
          else
          {
            ERROR("Hay un error en el num de objetos del archivo en el descriptor"," ",0);
            return(NULL);
          }
          if(OBTEN_TOKEN() == 1)
            rasgonum = atoi(token);
          else
```

```
        {
          ERROR("Hay un error en el num de rasgos del archivo  en el descriptor", " ",0);
          return(NULL);
        }
        rasgo = (NODO_RASGO *) malloc (sizeof(NODO_RASGO));
        auxrasgo = rasgo;
        for(i = 0; i< (rasgonum-1); i++)
        {
          if(OBTEN_TOKEN() == 2)
          {
            if(!LEE_RASGO_ARCH(auxrasgo, i)) return(NULL);
            otro = (NODO_RASGO *) malloc (sizeof(NODO_RASGO));
            auxrasgo->sig = otro;
            otro->ant = auxrasgo;
            auxrasgo = otro;
          }
          else
          {
            ERROR("Existe un error en el nombre del rasgo:"," ",i+1);
            return(NULL);
          }
        }
        if(OBTEN_TOKEN() == 2)
        {
          if(!LEE_RASGO_ARCH(auxrasgo, i)) return(NULL);
          auxrasgo->sig = rasgo;
          rasgo->ant = auxrasgo;
          return(rasgo);
        }
        else
        {
          ERROR("El nombre del £ltimo rasgo no es valido"," ",0);
          return(NULL);
        }
    }
    else
        do
        {
          mira =getc(entrada);
          if(feof(entrada)) break;
          ungetc(mira, entrada);
        }while(OBTEN_TOKEN() != 5);
  }
  return(NULL);
}

/*********************    Funci¢n REAL()    ***************/
/* Esta funci¢n se encarga de verificar si una cadena correspon-   */
/* de a un n£mero real, o no. Checando si todos los caracteres     */
/* de la misma son dígitos y si consta de un punto unicamente.     */
/*                                                                 */
/*        Entrada: Una cadena                                      */
/*    Salida: 1 si la cadena corresponde a un n£mero real y 0      */
/*          en caso contrario                                      */
/*        Funciones auxiliares: ERROR                              */
/*******************************************************/
```

```
int REAL(char caden[])
{
char *c;
int punto;

  c = caden;
  punto = 0;
  while(*c)
  {
    if(*c >= '0' && *c <= '9')
          c++;
    else
          if(*c == '.')
            if(!punto)
            {
              punto = 1;
              c++;
            }
            else
            {
              ERROR("Hay mas de un punto en el nÚmero real!! "," ",0);
              return(0);
            }
          else
          {
            ERROR("El nÚmero tiene caracteres no nÚmericos", " ",0);
            return(0);
          }
  }
  return(1);
}

/*********************    Funci¢n PRUEBA()    *************/
/* Esta funci¢n se encarga de verificar si el valor de una varia-    */
/* ble determinada corresponde con el tipo de la misma. Los tipos    */
/* son: booleano, rea y k-valente                                    */
/*                                                                   */
/*         Entrada: Tipo de la variable, su valor y la valencia      */
/*     Salida: 1 si el valor corresponde al tipo, 0 en caso          */
/*             contrario                                             */
/*         Funciones auxiliares: ERROR,REAL                          */
/*********************************************************/
int PRUEBA(char tipo, char *variable, char k)
{

  switch(tipo)
  {
    case 'b':while(1)
             {
                    if(*variable == '1' || *variable == '0' || *variable == '*')
                      return(1);
                    else
                    {
                     ERROR("El rasgo es de tipo booleano asi que s¢lo puede tomar valor 0 ¢ 1 "," ",0);
                     return(0);
                    }
```

```
            }

    case 'r': while(1)
                {
                 if(REAL(variable) || *variable == '@')
                   return(1);
                 else
                 {
                  ERROR("El rasgo no es de tipo real "," ",0);
                  return(0);
                 }
                }

    case 'k': while(1)
                {
                 if((*variable >= '0' && *variable <= k) || *variable == '@')
                   return(1);
                 else
                 {
                   ERROR("El valor de la variable est  fuera de rango "," ",0);
                   return(0);
                 }
                }
  }
  return(0);
}

/*********************  Función VALOR_MATRIZ()  ************/
/* Esta función se encarga de obtener el dato que se encuentra en la po-  */
/* sición (ren, col) de una matriz de caracteres. Dicha matriz es de      */
/* acceso dinamico.                                                       */
/*                                                                        */
/*       Entrada:EL nombre de la matriz, la posición por renglon y colum- */
/*         na del elemento a obtener y la cantidad de columnas de         */
/*         la matriz.                                                     */
/*    Salida: El elemento que se encuentra en la posición (ren, col)      */
/*       Funciones auxiliares: NINGUNA                                    */
/**************************************************************/
CADENA *VALOR_MATRIZ(CADENA *matriz, int ren, int col, int REN)
{
  return(*(matriz + (ren * (REN+1)) + col));
}

/*********************  Función PON_EN_MATRIZ()  ************/
/* Esta función se encarga de poner en una matriz el dato que se encuen-  */
/* tra en la posición (ren, col) de la misma. Dicha matriz es de acceso   */
/* dinamico.                                                              */
/*                                                                        */
/*       Entrada:EL nombre de la matriz, la posición por renglon y colum- */
/*         na del elemento a obtener, la cantidad de columnas de          */
/*         la matriz y el valor a colocar en la matriz.                   */
/*    Salida: NINGUNA                                                     */
/*       Funciones auxiliares: NINGUNA                                    */
/**************************************************************/
void PON_EN_MATRIZ(CADENA *matriz, short ren, short col, char valor[], int REN)
{
```

```
  strcpy(*(matriz + (ren * (REN+1)) + col), valor);
}

/*********************    Función LEE_MA_ARCH()    ****************************/
/* Esta función se encarga de obtener los datos de un archivo y poner-          */
/* los dentro de una matriz de acceso dinamico.                                 */
/*                                                                              */
/*        Entrada:EL nombre de la matriz y el apuntador al primer nodo de       */
/*          la lista de variables.                                              */
/*    Salida: NINGUNA                                                           */
/*        Funciones auxiliares:PANTALLA,TITULO,OBTEN_TOKEN, PON_EN_MATRIZ       */
/*                  PRUEBA, ERROR.                                              */
/*******************************************************************************/
void LEE_MA_ARCH(CADENA *MA, NODO_RASGO *list_rasg)
{
int ren, col;
char mira;


   PANTALLA();
   TITULO("      MATRIZ DE APRENDIZAJE");
   for(ren = 0; ren < tot_obj; ren++)
   {
        if( (mira = getc(entrada)) != '\n')
          ungetc(mira,entrada);
        if(OBTEN_TOKEN() == 1)
        {
          PON_EN_MATRIZ(MA, ren, 0, token, rasgonum);
          for(col = 1; col <= rasgonum; col++)
          {
            OBTEN_TOKEN();
            while(list_rasg->num_rasgo != col)
                  list_rasg = list_rasg->sig;
            if(PRUEBA(list_rasg->tip, token,list_rasg->k))
                  PON_EN_MATRIZ(MA, ren, col, token, rasgonum);
            else
                  break;
          }
        }
        else
          ERROR("Hay un error en el número de clase del objeto siguiente"," ",0);
   }
}

/*********************    LECTURA_DATOS()    ******************/
/* Esta función se encarga de llenar la lista de varibles y de                  */
/* llenar la matriz de datos por medio de algunas funciones                     */
/* auxiliares                                                                   */
/*                                                                              */
/*        Entrada:Ninguna                                                       */
/*    Salida: Ninguna                                                           */
/*        Funciones auxiliares: PANTALLA, TITULO,LEE_ARCH_DESC,                 */
/*                  LECT,LEE_MA_ARCH,ERROR                                      */
/*******************************************************************************/
void LECTURA_DATOS()
{
```

```
  PANTALLA();
  TITULO(    "An lisis de datos");
  if((pri_nom_rasg = LEE_ARCH_DESC()) != NULL)
  {
    fclose(entrada);
    if(LECT())
    {
        MA = (CADENA *) malloc (tot_obj * (rasgonum + 1) * (sizeof (CADENA)));
        LEE_MA_ARCH(MA, pri_nom_rasg);
    }
    else exit(1);
  }
  else
  {
    ERROR("El nombre del archivo no existe en el descriptor"," ",0);
        flushall();
  }
}
/*********************   Funci¢n LISTA_DESC()   **********************/
/* Esta funci¢n se encarga de listar el archivo de descriptores                */
/*                                                                             */
/*       Entrada:Ninguna                                                       */
/*    Salida: Ninguna                                                          */
/*       Funciones auxiliares: PANTALLA,TITULO,ERROR,OBTEN_TOKEN */
/*****************************************************************/
void LISTA_DESC()
{
int i, num;
char mira;

  PANTALLA();
  TITULO ( "     DESCRIPTOR DE ARCHIVOS:");
  strcpy(nom_des, "a:descrip.dat");
  rewind(entrada);
  if( (entrada = fopen(nom_des, "r+") ) == NULL){
    ERROR("No se puede abrir el descriptor de archivos"," ",0);
    return;
  }
  gotoxy(MinX+10,MinY); printf("Nombre del\t  No. de\t  No. de ");
  gotoxy(MinX+10,MinY + 1);printf(" archivo  \t  objetos \t rasgos ");
  gotoxy(MinX+10,MinY + 2);printf("=============================================");
  gotoxy(MinX+10,MinY + 4);printf("=============================================");
  while(!feof(entrada))
  {
    gotoxy(MinX+10,MinY + 3);
    if( (mira = getc(entrada)) != '\n') ungetc(mira, entrada);
    for(i = 0; i < 3; i++)
    {
        OBTEN_TOKEN();
        printf("%12.12s",token);
    }
    num = atoi(token);
    flushall();
    for(i = 0; i < num; i++)
    {
        OBTEN_TOKEN();
```

```
        gotoxy(MinX+5,MinY + 5);printf("Nombre del rasgo %d: %s \t",i+1,token);
        OBTEN_TOKEN();
        gotoxy(MinX+5,MinY + 7);printf("Tipo : %s\t\t",token);
        if(tolower(token[0]) == 'k')
        {
          OBTEN_TOKEN();
          gotoxy(MinX+15,MinY + 7);  printf("con k = %s", token);
        }
        OBTEN_TOKEN();
        gotoxy(MinX+5,MinY+9); printf("Criterio de comparación : %s\t\t\t", token);
        if(token[0] == '1')
        {
          OBTEN_TOKEN();
          gotoxy(MinX+35, MinY+9); printf("con epsilon = %s", token);
        }
        OBTEN_TOKEN();
        gotoxy(MinX+5, MinY+11); printf("Peso informacional : %s\t",token);
        ERROR("Para continuar con otro rasgo"," ",0);
    }
    ERROR("Para continuar con otro archivo"," ",0);
  }
  PANTALLA();
}
```

## A3.- Rutinas del algoritmo

```
/*********************  Función GUARDA_ARCH()  *********************/
/* Esta función se encarga de guardar en un archivo los valores de una  matriz  */
/*                                                                              */
/*       Entrada:EL nombre de la matriz, la cantidad de columnas y renglones y el  */
/*         glones y el nombre del archivo donde seran guardados                 */
/*         los datos                                                            */
/*    Salida: Ninguna                                                           */
/*       Funciones auxiliares: ERROR                                            */
/********************************************************************************/
void GUARDA_ARCH(double *matriz, int ren, int col, char *nom_archivo)
{
FILE *archivo;
int i,j;

  if((archivo = fopen(nom_archivo, "w")) == NULL)
  {
    ERROR("No se puede abrir el archivo: ",nom_archivo,0);
    exit(1);
  }
  for(i=0;i<ren;i++)
  {
    for(j=0;j<col;j++)
          fprintf(archivo,"%lf\t",matriz[(i*j)+j]);
    fprintf(archivo,"\n");
  }
  fclose(archivo);
}
```

```
/*********************  Funci¢n BUSCA_NOM_ARCH()  *********************/
/* Esta funci¢n se encarga de cambiar la extensi¢n de un archivo       */
/*                                                                     */
/*        Entrada:EL nombre del archivo y la nueva extensi¢n del archivo */
/*    Salida: una cadena que contiene el nombre del archivo con la nv  */
/*          ext.                                                       */
/*        Funciones auxiliares: Ninguna                                */
/*********************************************************************/
CADENA2 *BUSCA_NOM_ARCH (char *cadena, char *ext)
{
int i,indice;
char *cadena2;

  cadena2=(char *) malloc (12*(sizeof(char)));
  indice=strcspn(cadena,".");
  for (i=0; i<indice; i++)
    cadena2[i]=cadena[i];
  cadena2[i]='\0';
  strcat(cadena2,ext);
  free(cadena2);
  return (cadena2);
}

/*********************  Funci¢n REG_SOL()  *********************/
/* Esta funci¢n se encarga de verificar de que tipo es un objeto */
/* dado. Teniendo 4 tipos de objetos: Propios, Generales, At¡picos */
/* y No propios.                                                  */
/*                                                                */
/*        Entrada:Los parametros de soluci¢n y la tipiciadad y el */
/*          contraste del objeto.                                 */
/*          la matriz.                                            */
/*    Salida: Cadena que contiene el tipo de objeto               */
/*        Funciones auxiliares: NINGUNA                           */
/*****************************************************************/
CADENA *REG_SOL(double To, double Ko,·int x, double Ti, double Ki)
{

  if(Ti>To)
    if(Ki>Ko)
          if(x)
            return("Propio");
          else
            return("No propio");
    else
          return("General");
  else
    return("At¡pico");
}

/*********************  Funci¢n CALC_PARAM()  *********************/
/* Esta funci¢n se encarga de obtener un parametro que sive para la */
/* determinar de que tipo es un objeto.                             */
/*                                                                  */
/*        Entrada:Una matriz de pesos inofrmacionales y el n£mero de co- */
/*          lumnas y rasgos.                                        */
```

```
/*     Salida: valor del parametro                                          */
/*          Funciones auxiliares: NINGUNA                                   */
/*************************************************************************/
duoble CALC_PARAM(* double matriz, int num_obj, int columna)
{
double Xmax, Xmin, Xo;
int i;

  Xmax=matriz[columna];
  Xmin=matriz[columna];
  for(i=0;i<num_obj;i++)
    if(matriz[i*(num_clase+2)+columna] > Xmax)
          Xmax=matriz[i*(num_clase+2)+columna];
    else
          if(matriz[i*(num_clase+2)+columna] < Xmin)
            Xmin=matriz[i*(num_clase+2)+columna];

  Xo=(Xmax-Xmin);
  return(Xo);
}

/*******************    Funci¢n ANALI_OBJ()    ***********************/
/* Esta funci¢n se encarga de analizar los datos de toda una matriz,      */
/* determina  de que tipo son y los guarda en un archivo de datos.        */
/*                                                                        */
/*          Entrada:La matriz de los objetos  a ser clasificados, matriz  */
/*            de tipicidades y contrastes de dichos objetos y la          */
/*            cantidad de objetos a ser clasificados                      */
/*     Salida: Ninguna.                                                   */
/*          Funciones auxiliares: CALC_PARAM,BUSCA_NOM_ARCH,ERROR         */
/*                    VALOR_MATRIZ,REG_SOL,DESPLIEGA_RESULT               */
/*************************************************************************/
void ANALI_OBJ(double *matriz2, CADENA *matriz1, int num_obj)
{
char *obj;
char ext1[4]=".ADD";
int i,clase;
double Ko,To,c_obj,contraste;
char *sol;
FILE *archivo;

  To=CALC_PARAM(matriz2,tot_obj,0);
  Ko=CALC_PARAM(matriz2,tot_obj,1);
  strcpy(nom_archivo2, BUSCA_NOM_ARCH(nom_arch,ext1));
  if((archivo = fopen(nom_archivo, "w")) == NULL)
  {
    ERROR("No se puede abrir el archivo: ",nom_archivo,0);
    exit(1);
  }
  for(i=0;i<num_obj;i++)
  {
    c_obj=strtod(VALOR_MATRIZ(matriz1,i,0,rasgonum) &obj);
    clase=(int) c_obj;
    Tipmax=matriz2[i*(num_clase+2)+num_clase];
    valor2=matriz[i*(num_clase+2)+clase-1];
    contraste=matriz[i*(num_clase+2)+num_clase+1]
```

```
    if(valor1 == valor2)
        x=1;
    else
        x=0;
    sol=REG_SOL(To,Ko,x,Tipmax,contraste)
    fprintf(archivo,"%lf\n",sol)
  }
  fclose(archivo);
  DESPLIEGA_RESULT(nom_archivo2,tot_obj,1);
}

/*********************    Función CRIT_COMP()    *********************/
/* Esta función se encarga de verificar si dos valores son semejantes   */
/* o no dependiendo del criterio de comparacion. En este caso solo      */
/* existen 2 criterios de comparacion; la igualdad estricta, y la se-   */
/* mejanza por diferencia.                                              */
/*                                                                      */
/*        Entrada:Las matrices a las cuales pertenecen los elementos a  */
/*          comparar.                                                   */
/*    Salida: 1 si los elementos son semejantes 0 en caso contrario     */
/*        Funciones auxiliares: VALOR_MATRIZ                            */
/************************************************************************/
int CRIT_COMP(CADENA *matriz1, CADENA *matriz2, double epsilon, int ob1, int ob2, int rasg)
{
double valor1,valor2,z;
char *val1;
char *val2;
  if(epsilon == 0.0)
    if( (strcmp(VALOR_MATRIZ(matriz1,ob1,rasg,rasgonum),VALOR_MATRIZ(matriz2,ob2,rasg,rasgonum))) == 0)
        return(1);
    else
        return(0);
  else
  {
    valor1=strtod(VALOR_MATRIZ(matriz1,ob1,rasg,rasgonum), &val1);
    valor2=strtod(VALOR_MATRIZ(matriz2,ob2,rasg,rasgonum), &val2);
    z=fabs(valor1-valor2);
    if(z <= epsilon)
        return(1);
    else
        return(0);
  }
}

/*********************    Función FUN_SEM()    *********************/
/* Esta función se encarga de verificar si dos objetos son semejan-     */
/* tes, dependiendo de una cierta funcion de semejanza.                 */
/*                                                                      */
/*        Entrada: Nombre de las matrices a las que pertenecen los      */
/*          objetos, indice de los objetos,lista de rasgos              */
/*    Salida: 1 si los objetos son semejantes 0 en caso contrario       */
/*        Funciones auxiliares: CRIT_COMP                               */
/************************************************************************/


int FUN_SEM(CADENA *matriz1, CADENA *matriz2, int ob1, int ob2, int epsilon_fun, NODO_RASGO *l_rasgo)
```

```
{

int cuenta_rasgo, i;

  cuenta_rasgo=0;
  for(i=0; i<rasgonum; i++)
  {
    while(l_rasgo->num_rasgo != i)
        l_rasgo=l_rasgo->sig;
    if( CRIT_COMP(matriz1, matriz2, l_rasgo->epsilon, ob1, ob2, i) )
        cuenta_rasgo=cuenta_rasgo+1;
  }
  if(cuenta_rasgo >= epsilon_fun)
    return(1);
  else
    return(0);
}

/*********************  Función CALC_PI_PD()  ************************/
/* Esta función obtiene el Peso informacional y diferenciante de los  */
/* objetos de una matriz determinada y los guarda en una matriz de    */
/* acceso dinamico, asi como en un archivo.                           */
/*                                                                    */
/*        Entrada:El nombre de la matriz, y la lista de variables que */
/*          conforman dicha matriz                                    */
/*    Salida: Ninguna                                                 */
/*        Funciones auxiliares:VALOR_MATRIZ,CRIT_COMP,GUARDA_ARCH     */
/*                  BUSCA_NOM_ARCH                                    */
/**********************************************************************/

void CAL_PI_PD(CADENA *matriz,NODO_RASGO *l_rasgo)
{

int cont1;
int cont2;
float PI, PD;
int i,j,l;
CADENA *c_obj1;
CADENA *c_obj2;
char ext1[4]=".PID";
char nom_arch2[12];

  mat=(double *) malloc (2*tot_obj * (sizeof(double)));
  for(i=0; i<tot_obj; i++)
  {
    PI=0.0;
    PD=0.0;
    strcpy(c_obj1,VALOR_MATRIZ(matriz,i,0,rasgonum));
    for(j=1;j<=rasgonum;j++)
    {
        cont1=0;
        cont2=0;
        while(l_rasgo->num_rasgo != j)
          l_rasgo=l_rasgo->sig;
        for(l=0;l<tot_obj;l++)
        {
```

```
            strcpy(c_obj2,VALOR_MATRIZ(matriz,l,0,rasgonum));
            if( (CRIT_COMP(matriz,matriz,l_rasgo->epsilon,i,l,j)) )
              if( (strcmp(c_obj1,c_obj2)) == 0 )
                    cont1=cont1+1;
              else
                    cont1=cont1;
            else
              if( (strcmp(c_obj1,c_obj2)) !=0)
                    cont2=cont2+1;
              else
                    cont2=cont2;
          }
          PI=PI+(cont1 * l_rasgo->peso_inf_rasgo);
          PD=PD+(cont2 * l_rasgo->peso_inf_rasgo);
      }
    mat[i*2]=PI;
    mat[(i*2)+1]=PD;
  }
  strcpy(nom_arch2, BUSCA_NOM_ARCH(nom_arch,ext1));
  GUARDA_ARCH(mat,tot_obj,2,nom_arch2);
}

/*********************  Función TIPICIDAD()  *********************/
/* Esta función se encarga de obtener la tipicidad de los objetos de      */
/* una matriz dada. Colocando los resultados en otra matriz.              */
/*                                                                        */
/*      Entrada:EL nombre de la matriz, la matriz de pesos informa        */
/*        cionales y diferenciantes de dichos objetos, el nume-           */
/*        de objetos y la lista de variables de la matriz.                */
/*   Salida: Ninguna                                                      */
/*      Funciones auxiliares: VALOR_MATRIZ, FUN_SEM                       */
/*************************************************************************/

void TIPICIDAD(CADENA *matriz1, double *mat_PI_PD,CADENA *matriz2,int num_obj, NODO_RASGO *l_rasgo)
{

int i,j, l, clase;
double c_obj,Tmax,tipic;
double T_j_1,T_j_0,T_i_1,T_i_0;
char *c_o;

  mat_tip_cont=(double *) malloc ((clase+2)*num_obj *(sizeof(double)));
  for(i=0; i<num_obj; i++)
  {
    Tmax=0.0;
    for(j=0;j<num_clase; j++)
    {
          T_j_1=0.0;
          T_j_0=0.0;
          T_i_1=0.0;
          T_i_0=0.0;
          for(l=0; l< tot_obj;l++)
          {
            c_obj=strtod(VALOR_MATRIZ(matriz1,l,0,rasgonum), &c_o);
            clase= (int) c_obj;
            if(FUN_SEM(matriz1,matriz2,l,i,umbral,l_rasgo))
```

```
            if( clase == (j+1) )
                T_j_1=T_j_1+mat_PI_PD[2*i];
            else
                T_i_1=T_i_1+mat_PI_PD[2*i];
          else
            if(clase == (j+1) )
                T_i_0=T_i_0+mat_PI_PD[(2*i)+1];
            else
                T_i_0=T_i_0+mat_PI_PD[(2*i)+1];
          }
          tipic=((T_j_1+T_i_0)/(T_j_0+T_i_1));
          mat_tip_cont[(i*(num_clase+2))+j]=tipic;
          if(tipic > Tmax)
            Tmax=tipic;
      }
      mat_tip_cont[(i*(num_clase+2))+num_clase]=Tmax;
    }
}

/*********************  Función CONTRASTE()  ***************/
/* Esta función se encarga de calcular el contraste de los objetos de      */
/* una matriz dada, guardando los resultados en la matriz de tipici-       */
/* dades.                                                                  */
/*                                                                         */
/*        Entrada:EL nombre de la matriz que contiene los objetos y el     */
/*          numero de objetos.                                             */
/*    Salida: Ninguna                                                      */
/*        Funciones auxiliares: NINGUNA                                    */
/***************************************************************/

void CONTRASTE(double *matriz, int num_obj)
{

int i,j;
double dividendo, Tmax;
double resul_par=0.0, contraste=0.0;

  for(i=0; i<num_obj; i++)
  {
    dividendo=0.0;
    Tmax = matriz[(i*(num_clase+2))+num_clase];
    for(j=0; j < num_clase; j++)
          dividendo=dividendo+(Tmax - matriz[i*(num_clase+2)+j]);
    resul_par=(dividendo)/(num_clase-1);
    contraste=sqrt(resul_par);
    matriz[i*(num_clase+2)+num_clase+1]=contraste;
  }
}
```

## A4.- Rutinas de interface

```
int columna = 0, renglon = 0;
int MaxX=79, MaxY=18;
```

```
int MinX=2, MinY=6;
char blanco[75]="                                                                    ";

/*********************  Función VENTANA  ***************/
/* Esta función crea una ventana en las coordenadas determinadas  */
/* los parámetros de entrada de la misma.                         */
/*                                                                */
/*        Entrada: Coordenadas (X1, Y1), (X2, Y2)                 */
/*        Salida: Ninguna                                         */
/*        Funciones auxiliares: Ninguna.                          */
/******************************************************************/

 void VENTANA(short X1, short Y1, short X2, short Y2)
 {

 short ren, col;

  for(col = X1; col <= X2; col++)
  {
    gotoxy(col,Y1);printf("ß");
    gotoxy(col,Y2);printf("Ü");
  }
  for(ren = Y1; ren <= Y2; ren++)
  {
        gotoxy(X1,ren); printf("Û");
        gotoxy(X2,ren); printf("Û");
  }
 }


/*********************  Función PANTALLA  ******************/
/* Esta función crea la pantalla principal que consta de 3 ventanas. La  */
/* primera que corresponde a la parte de los títulos, la segunda al área de */
/* trabajo y la tercera a los mensajes de error.                          */
/*                                                                        */
/*        Entrada: Ninguna.                                               */
/*        Salida:  Ninguna                                                */
/*        Funciones Auxiliares: VENTANA                                   */
/**************************************************************************/

void PANTALLA()
{

 clrscr();
 VENTANA(MinX-1, 1, MaxX+1,3);
 VENTANA(MinX-1,MinY-1, MaxX+1, MaxY+1);
 VENTANA(MinX-1, 21, MaxX+1, 24);
}


/*********************  Funcón TITULO  ***********/
/* Esta función se encarga de poner una cadena de caracteres*/
/* en la ventana correspondiente al título.                  */
/*                                                           */
/*        Entrada: Una cadena de caracteres                  */
/*        Salida:  Ninguna                                   */
```

```
/*        Funciones Auxiliares: Ninguna                              */
/*************************************************************/

void TITULO(char *caden)
{
  gotoxy(20,2); printf("%s",caden);
}


/*********************    Función ERROR    **********/
/* Esta función se encarga de poner dos cadenas de carac- */
/* teres en la ventana correspondiente a los mensajes de    */
/* error.                                                   */
/*                                                          */
/*        Entrada: Dos cadenas de caracteres                */
/*        Salida: Ninguna                                   */
/*        Funciones Auxiliares:Ninguna                      */
/*************************************************************/

void ERROR(char *cad1, char *cad2, int j)
{

  gotoxy(6,22); printf("%s", blanco);
  gotoxy(4,23);printf("%s", blanco);
  gotoxy(4,22); printf("%s", cad1);
  if(j > 0)
    printf("%d", j);
  else
    printf("%s", cad2);
  gotoxy(65,23);printf("Presione ENTER");
  getch();
}


/*********************    Función MENU_PRIN    *****/
/* Esta función se encarga de presentar el menú principal   */
/* del sistema.                                             */
/*                                                          */
/*        Entrada:Ninguna                                   */
/*        Salida: Ninguna                                   */
/*      Funciones Auxiliares: PANTALLA, TITULO              */
/*************************************************************/

void MENU_PRIN()
{

 clrscr();
 PANTALLA();
 TITULO("          Menú Principal          ");
 gotoxy(30,7); printf("1.- Analizar datos");
 gotoxy(30,10); printf("2.- Clasificar");
 gotoxy(30,13); printf("3.- Listar archivos");
 gotoxy(30,16); printf("4.- Salir");
 gotoxy(65,23); printf("Presione ENTER");
 gotoxy(6,22); printf("Digite su opción: ");
}
```

## A5.- Ejemplos de archivos de datos

/********** Prototipo del descriptor de archivos **********/

| | | | | | | |
|---|---|---|---|---|---|---|
| arch1.ext | número de clases | función de semejanza[1] | epsilon de semejanza[2] | número de objetos | número de atributos | datos de cada una de las variables * |
| arch2.ext | número de clases | función de[1] semejanza | epsilon de semejanza[2] | número de objetos | número de atributos | datos de cada una de las variables * |
| | • | | | | | |
| | • | | | | | |
| | • | | | | | |
| arch2.ext | número de clases | función de[1] semejanza | epsilon de semejanza[2] | número de objetos | número de atributos | datos de cada una de las variables * |

1. Si la función de semejanza es por igualdad estricta el codigo es 2 y se omitira el parametro epsilon de semejanza.
2. Si la función de semejanza es por el modulo de la diferencia menor que un epsilon el codigo será 1 y el epsilon será de acuerdo al criterio del usuario.

* Los datos de cada una de las variables se representan de la siguiente manera:

| nombre variable | tipo de | valencia[1] | criterio de comparación | epsilon del crit.comp [2] | peso informacional |
|---|---|---|---|---|---|

1. En caso de que el tipo de variable sea k-valente se pondra la valencia de lo contrario el dato se omite.
2. En caso de que el criterio de comparación sea modulo la deferencia se pondra el epsilon de error en caso contrario se omite.

El codigo del criterio de comparación será 1 si es por modulo de la diferencia y 2 si es por igualdad estricta.

/********** Archivo de datos arch1.ext **********/

| | | | | |
|---|---|---|---|---|
| clase del objeto O1 | X1(O1) | X2(O1) | . . . | Xn(O1) |
| clase del objeto O2 | X1(O2) | X2(O2) | . . . | Xn(O2) |
| . | | | | |
| . | | | | |
| . | | | | |
| clase del objeto Om | X1(Om) | X2(Om) | . . . | Xn(Om) |

Dode Xi(O) es el valor admisible del objeto O para la variable Xi.