CENTRO DE INVESTIGACION Y ESTUDIOS AVANZADOS DEL IPN

DEPARTAMENTO DE INGENIERIA ELECTRICA
SECCION COMPUTACION

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

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

PRESENTA

LETICIA VEGA ALVARADO

DIRECTOR DE TESIS: DR. JOSE RUIZ SHULCLOPER

MEXICO, D.F.

ENERO, 1996

XM

Indice

Introducción

El reconocimiento es un atributo del ser humano, así como de otros organismos vivos. El proceso de reconocimiento está presente en cada instante de sus vidas. Pueden reconocer los objetos que están a su alrededor, de una manera rápida y sencilla, y moverse y actuar sobre la base de ellos. En este sentido, 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 de manera similar a como lo hace el ser humano. Así, a principios de los años 50´s, con el surgimiento y auge de las computadoras digitales se dieron los primeros pasos en el reconocimiento automatizado de patrones. En la década de los 60´s se incrementaron rápidamente las investigaciones en el área 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.

En los últimos años ha crecido enormemente el interés en el reconocimiento de patrones en estudios interdisciplinarios e investigaciones de diferentes campos como son: Medicina, Psicología, las Geociencias, Sociología y otras más en las cuales con frecuencia surgen problemas que tienen que ver con la clasificación, el diagnóstico, pronóstico y el análisis de datos. Por ejemplo, el pronóstico de complicaciones postoperatorias de un paciente, la determinación de zonas perspectivas para algún mineral, el diagnóstico del estado mental de un paciente, etc.. Esto ha creado una necesidad inmensa de desarrollar métodos y técnicas para ser utilizadas en el diseño de sistemas de información computarizado para el reconocimiento de patrones.

No obstante que el desarrollo y creación de los métodos y técnicas de reconocimiento de patrones surgieron por la necesidad de resolver los problemas planteados anteriormente, la mayoría de dichos métodos no cumplen con las características más importantes que presentan los problemas, ya que imponen una serie de restricciones que provocan que el problema se tenga que ajustar a la técnica y no de manera contraria, es decir, que la técnica se ajuste a las propiedades del problema.

Uno de los factores más importantes que se presentan es que, ergeneral, los 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, como es el caso del sexo, 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, que están en dependencia de la persona que las describe. Sin embargo, muchos métodos consideran 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 éstos números y se hallan normas y distancias; 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.

Otra de las características que también aparece con frecuencia, es que no siempre se tiene la información completa, es decir, que hay propiedades que no han sido determinadas ya sea porque es imposible hacerlo o porque resulte muy costoso; como por ejemplo, en el caso de la geología, que para 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. Sin embargo, muchos métodos no consideran la posibilidad de ausencia de información y cuando esta se presenta en los problemas reales se hace uso de la estadística para estimarla, lo cual nos podría llevar a falsear la información, ya que para estimar la información se parte de una serie de presupuestos que en muchas ocasiones no se cumplen en los problemas reales, por ejemplo, considerar que los datos faltantes siguen una distribución normal.

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, 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 toda certeza afirmar si un objeto pertenece a una cierta clase, sino que valora de alguna manera cierto grado de certidumbre para su afirmación. Pero muchos de los métodos desarrollados hasta ahora consideran únicamente la posibilidad de clases no disjuntas.

Considerando las limitantes que presentan la mayoría de los modelos que se utilizan en la solución de los problemas reales, nos damos cuenta que aun cuando en los últimos años el Reconocimiento de Patrones ha tenido un gran auge en el desarrollo de métodos y técnicas, sigue existiendo la necesidad de crear, buscar o extender modelos que se ajusten de una manera más adecuada a las características del problema y no de manera contraria, como se ha venido realizando históricamente.

Por tanto, uno de los objetivos y motivaciones de este trabajo es fundamentalmente la necesidad de resolver de una manera más apropiada los problemas que hemos mencionado a lo largo de este epígrafe, elaborando un algoritmo de análisis de datos y clasificación 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 pueden ser de cualquier tipo: numéricas, bivalentes e incluso difusas; que pueda trabajar con ausencia de información en las descripciones de los objetos y que tome en cuenta que los objetos pueden pertenecer a más de una clase e incluso, que puedan pertenecer a todas las clases con cierto grado de certidumbre, i.e, que las clases en que se divide nuestro problema no necesariamente tengan que ser disjuntas. La implantación computacional del algoritmo constituye el otro objetivo del trabajo, y para el desarrollo de éste 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 Vaskovskii B.[1].

El presente trabajo está constituido por tres capítulos. En el capítulo 1 haremos una breve descripción de lo que es el análisis de datos y de su importancia en algunas áreas de la investigación. Describiremos el algoritmo de análisis de datos propuesto por Diordenko L. y Vaskosvskii B. [1], así como los primeros resultados obtenidos, los cuales se presentan en el trabajo de Vega L. [2], mencionando las limitantes que presentan los mismos. El capítulo 2 estará dedicado a la exposición de un nuevo algoritmo de clasificación y análisis de datos multivariado, basado en las ideas principales de los algoritmos presentados en el capítulo 1, mostrando las ventajas de dicho algoritmo. Por último en el capítulo 3 se describe la implantación computacional del algoritmo presentado en el capítulo 2.

Los resultados obtenidos en este trabajo han sido expuestos en los siguientes eventos:

- Segundo Taller Iberoamericano de "Informática y Geociencias", realizado en La Habana, Cuba del 1ro al 4 de Agosto de 1994.

- Conferencia Internacional Ciencia y Tecnología para el desarrollo "CIMAF 95", 1er Taller Iberoamericano de Reconocimiento de Patrones, efectuado en la ciudad de La Habana , Cuba del 23 al 27 de enero de 1995.

y publicados en las Memorias del "1er Taller Iberoamericano de Reconocimiento de Patrones, efectuado en la ciudad de La Habana , Cuba del 23 de al 27 de enero de 1995.

Capítulo 1

Conceptos y resultados previos

1.1 Análisis de datos y la determinación de anomalías

A través de la historia, el hombre ha hecho grandes esfuerzos para entender y conocer su medio ambiente. En la antigüedad el hombre se enfrentaba a una serie de fenómenos para los cuales no tenía ninguna explicación lógica, pero poco a poco por medio de la observación repetitiva y el análisis de algunos de ellos, pudo explicarse el origen de los mismos.

Con el paso del tiempo se crean instrumentos y mecanismos más precisos para la observación de ciertos fenómenos, lo que incrementa la cantidad de información adquirida a través de dichas observaciones; tal es el caso de los dispositivos electrónicos para sensar las ondas sísmicas, de los electrocardiógrafos,de los métodos para determinar la aparición de eclipses, etc. Por tanto, las investigaciones científicas comienzan a ser un área donde se generan una gran cantidad de datos.

Es evidente entonces que la necesidad de métodos matemáticos para el análisis de estos datos resulta crucial. Las primeras técnicas creadas para el análisis de datos son de tipo estadístico.

El análisis de datos puede llevarse a cabo con diferentes propósitos, por ejemplo, se pueden determinar algunas medidas estadísticas como son: la varianza, la media, la moda, la desviación estándar, la frecuencia, etc. Por ejemplo, se puede establecer cuáles son las estaturas que aparecen con más frecuencia en un grupo de jóvenes adolescentes, qué datos tienen frecuencia máxima, es decir, cuál es la moda de las estaturas, qué tan semejantes son todas ellas con respecto a la(s) más frecuente(s), cómo se distribuyen las mismas, etc. Asimismo, existen otras medidas que nos proporcionan la relación entre las variables obtenidas a través de las observaciones del fenómeno en estudio, como es la

medida de correlación. En el caso de las estaturas se puede definir qué relación existe entre la estatura y el sexo, por ejemplo.

Dentro del área del Reconocimiento de Patrones existe una gama muy grande de problemas que están relacionados con el análisis de datos; tal es el caso de la determinación de patrones existentes en una muestra, esto es, dado un conjunto de datos establecer cómo se agrupan. En este caso, el proceso de agrupamiento se puede llevar a cabo verificando cuáles son las características que cumplen los datos y agrupando aquellos que se asemejan más con respecto a dichas características, ejemplos de esto son los algoritmos Holotipo y Class[3].

Otro ejemplo de análisis de datos en problemas de reconocimiento de patrones, (específicamente en problemas de clasificación supervisada, que son aquellos donde se tienen muestras de cada una de las clases en que se divide el problema), es el de determinar la importancia informacional de un objeto dentro de una clase dada, i.e, qué tanta información aporta o qué tan característico es dicho objeto para la clase. Este proceso se puede realizar de diferentes maneras , una de ellas es la verificar la frecuencia de aparición de los datos que describen a dicho objeto dentro de la clase y a partir de ahí definir su importancia; u obtenerse 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 muestras representativas.

En general, en los problemas de clasificación sería ideal contar con una muestra en la cual todos los 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 muestra representativa, 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, o haciendo una ponderación de los objetos, de tal manera que sean tratados de una manera diferenciada. Existen muchas maneras para determinar los datos que no son representativos de una muestra, los cuales son llamados "anomalías"; una de ellas podría ser el considerar anomalías a todos aquellos datos que sean menores en una cantidad ε que la media de la muestra; otra manera estaría en función de la semejanza de los datos de la muestra, considerando anómalos a todos aquellos que no sean suficientemente semejantes a la mayoría.

En muchos casos la determinación de anomalías resulta de gran utilidad, ya que estas pueden por sí solas identificar áreas de gran interés. Tal es el caso de los problemas que se presentan en la Geología, Geofísica, Geoquímica y otras donde los valores anómalos pudieran reflejar la presencia de fenómenos raros que dieran indicios, por ejemplo, de la existencia de yacimientos de un determinado mineral. Sin embargo, no siempre la presencia de valores anómalos representa la existencia de fenómenos de interés, algunas veces la aparición de anomalías puede ser indicio de una muestra pobre, es decir, no representativa o de falta de conocimiento de la región que se esté trabajando.

Hasta el momento hemos hablado únicamente de los problemas de análisis de datos considerándolos de tipo univariado; por ejemplo, cuando mencionamos que podíamos obtener la frecuencia de aparición de las estaturas de un grupo de adolescentes o cuando hablamos de la semejanza de las mismas. Sin embargo, existen problemas donde es necesario considerar la aparición de variables de manera simultánea, por ejemplo, no sólo considerar la estatura del grupo de adolescentes sino también el sexo y su edad, y así determinar cuál es la frecuencia de aparición de adolescentes que tienen x-edad, r-estatura y z-sexo. A este tipo de análisis se le conoce como "análisis multivariado"

En la práctica, en muchos de los problemas es más importante considerar las variables de manera conjuntual que de manera independiente. Por ejemplo, en el caso de la medicina la aparición de fiebre por sí sola no es tan importante dentro de alguna enfermedad, como la aparición de fiebre y vómito simultáneamente o la aparición de fiebre y diarrea. Es por eso que el análisis multivariado es de gran importancia en muchos de los problemas relacionados con el reconocimiento de Patrones.

Por último, podemos decir que en general 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, pero es importante mencionar que por sí solo forma un problema.

1.2 Algoritmo para el análisis de datos(Diordenko L. y Vaskovskii)

El algoritmo que se describirá en ésta sección se encuentra enmarcado dentro del problema de clasificación supervisada, el cual será definido más adelante. Este algoritmo es para el análisis de datos, que como ya mencionamos, constituye una parte muy importante para el Reconocimiento de Patrones (RP). El algoritmo fue elaborado y ha sido utilizado de manera particular, 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 hará el planteamiento formal del problema de clasificación supervisada, se darán los conceptos fundamentales del algoritmo de análisis de datos y finalmente se hará el planteamiento formal del problema de análisis de datos.

Un problema de clasificación supervisada desde el enfoque lógico-combinatorio [4] de R. P. se define de la siguiente manera:

Se tienen m descripciones de objetos $I(O_1),\ldots,I(O_m)$ divididos en ℓ clases, $K_1,\cdots, K_\ell$. Sea $R=\{x_1,\cdots, x_n\}$ el conjunto de rasgos en términos de los cuales se describen los objetos donde 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, puede ser de tipo booleano, tomar valores dentro de un conjunto $\{ a_1, \ldots, a_k \}$ 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. 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 de un objeto dado no se conoce. Toda esta información puede ser representada en una matriz donde las descripciones de los objetos conforman las filas de la misma (Fig. 1). A dicha matriz la denominaremos *matriz de aprendizaje* (MA) en lo sucesivo.

		x_1		x_n
	O_1	$x_1(O_1)$		$x_n(O_1)$
K_1	⋮	⋮	. .	⋮
	O_s	$x_1(O_s)$		$x_n(O_s)$
	⋮	⋮		⋮
	⋮	⋮		⋮
	⋮	⋮	. . .	⋮
	O_t	$x_1(O_t)$		$x_n(O_t)$
K_ℓ	⋮	⋮		⋮
	O_m	$x_1(O_m)$		$x_n(O_m)$

$$¿O = (x_1(O) \cdots x_n(O)) \in K_i?$$

fig 1 Representación simbólica del problema de clasificación con aprendizaje.

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 $K_1, \cdots, K_\ell$ pertenece.

Una vez que hemos definido el problema de clasificación supervisada, haremos una breve descripción de los conceptos fundamentales del algoritmo de análisis de datos que se presenta en esta sección.

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, se dice 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, este modelo únicamente se basa 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 en los cuatro tipos de objetos que serán descritos posteriormente. Dados los conceptos fundamentales del algoritmo, haremos la formalización del problema de análisis de datos.

Se tiene un problema como el que se planteó al principio de esta sección, que está representado simbólicamente en la matriz de aprendizaje (MA Fig. 1). En este caso los valores admisibles de los rasgos son de tipo real. Cada una de las clases de MA puede estar constituida por objetos de 4 tipos dependiendo de qué tan típico y contrastante sea cada uno de ellos 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 el problema que se está estudiando y por tanto presentan interés particular.

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

Descripción del algoritmo

El algoritmo de clasificación que a continuación se describe fue propuesto por Diordenko L. y por Vaskosvskii B. [1]

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 anteriormente. El algoritmo consta de dos etapas.

PRIMERA ETAPA

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

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 (fig 2), tomando en consideración y respetando las reglas generales para formar las distribuciones de frecuencia, y por consiguiente, los histogramas de frecuencia.

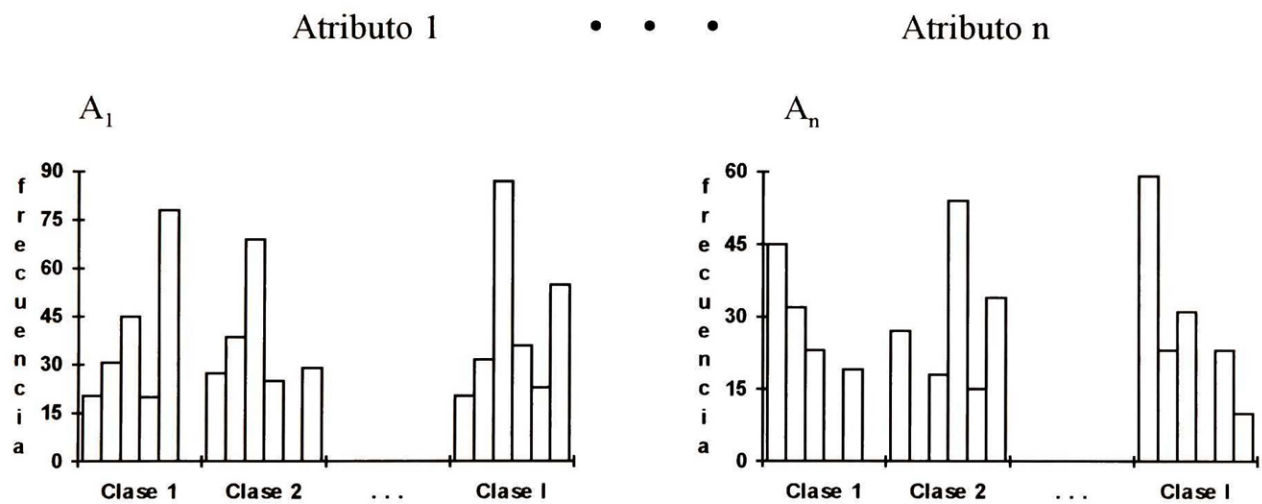


fig. 2 Histogramas de frecuencia de los atributos para cada una de las clases

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,. . . , ℓ

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 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 (fig. 2) 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 "se ubica" en el intervalo i del histograma construido para el mismo. El índice del número del intervalo (i) cuando se habla de objetos puede omitirse, y por tanto la tipicidad de cada objeto para uno de los atributos en cada una de las clases en lo subsecuente se designará como $t_j(s)$ donde: j es el índice correspondiente al atributo y s es el 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(que a continuación se muestra).

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

se convierte en una expresión matricial del tipo:

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

donde:

n: Cantidad de atributos

ℓ : Cantidad de clases

Lo que significa que cada objeto viene expresado en términos de un arreglo bidimensional, como se muestra en la matriz de la fig 3. Con este paso concluye la primera etapa del algoritmo.

	x_1	x_2	$\cdots$	x_n
Clase K_1'	$t_1(1)$	$t_2(1)$	$\cdots$	$t_n(1)$
Clase K_2'	$t_1(2)$	$t_2(2)$	$\cdots$	$t_n(2)$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
Clase K_ℓ'	$t_1(\ell)$	$t_2(\ell)$	$\cdots$	$t_n(\ell)$

fig. 3 Matriz de tipicidades por atributo para cada objeto.

SEGUNDA ETAPA

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, mediante la siguiente expresión.

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

donde s=1, ..., ℓ

Para cada objeto se calculan ℓ 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 clases, por lo que se tiene la tipicidad de cada objeto en cada una de las clases.

El objeto será más típico para aquella clase donde T(s) sea máximo, por tanto, el nivel de tipicidad (T) del objeto (O_i) será igual al máximo de las tipicidades por clase, tal como se muestra en la siguiente expresión.

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

Finalmente, para cada objeto O_i se tendrá un valor de T que denominaremos (T_i), correspondiente a la tipicidad máxima en las clases.

En este sentido resulta también importante determinar la clase asociada a T_i , o sea, la clase donde se obtuvo el valor máximo de T(s). Esta clase la denominaremos Clase Máxima del objeto O_i (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). Es importante mencionar que la expresión para el cálculo del contraste es exactamente igual a la expresión de la desviación estándar, la cual nos proporciona una medida de cuánto se dispersan las tipicidades de cada una de las clases con respecto a la tipicidad máxima.

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 delimitación alto-bajo

(POR; tal que, $1 \leq POR \leq 100$). Los valores de umbral para T_i y C_i se calculan por medio de las siguientes expresiones:

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

donde:

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.

Limitantes del algoritmo

Como ya hemos mencionado, 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 se hace uso de modelos y métodos que no se ajustan de la mejor forma a las características de dichos problemas, sino por el contrario, tratan de que los problemas se ajusten a las condiciones que impone el modelo. En este sentido, utilizar modelos sin violentar los principios de la modelación matemática, no es posible. Tal es el caso del algoritmo aquí presentado, el cual tiene un conjunto de condiciones o limitantes que hacen deficiente su aplicación a la mayoría de los problemas de clasificación citados, aunque cabe mencionar que la esencia del algoritmo es buena, ya que para hacer el análisis de los datos toma en consideración qué tan representativo o típico es un objeto con respecto a cada una de las clases y, a su vez, qué tan contrastante es con respecto al resto de las clases, lo cual nos da una idea de cómo se comporta dicho objeto; por tanto, éste algoritmo pudiera ser de mucha utilidad si se eliminaran las limitantes que a continuación se enlistan, dando las razones por las cuales se considera que éstas constituyen un problema.

1) Unicamente trabaja con variables de tipo real.

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 ; k-valentes como el tipo de suelo en la Geología (arenoso, pantanoso, rocoso); hasta variables de tipo lingüístico, como es el caso del dolor el cual puede estar gradado en poco, regular, mucho, etc. Así como este problema, existen muchos otros en los cuales se utilizan diferentes tipos de variables; por lo tanto, no siempre se puede pedir que los problemas sean representables en el espacio de los números reales, a menos que se suponga erróneamente, que las variables cualitativas se puedan modelar con números sobre los cuales sean admisibles operaciones aritméticas.

2) No considera la posibilidad de ausencia de información.

En muchas ocasiones en los problemas no contamos con la información completa para todos los objetos, es decir, 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 Medicina, existen características de algunos pacientes que se desconocen, como pudieran ser los antecedentes familiares 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, suponiendo leyes de distribución de valores de la misma.

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

Las variables que describen a los objetos no siempre nos proporcionan la misma información, es decir, existen algunas variables que son más importantes que otras y por eso mismo, resultan ser más discriminantes cuando se lleva a cabo el proceso de clasificación. Evidentemente las variables que son muy importantes dentro del proceso de clasificación tienen un peso informacional muy grande y aquellas que son menos importantes y quizá no imprescindibles tienen un peso informacional no muy grande. Un ejemplo de esto sería 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) No permite utilizar diferentes criterios de comparación entre valores de una variable.

El criterio de comparación que se maneja en el algoritmo, es un criterio por intervalos y éste es el mismo para todas las variables, lo que 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 ellas deben 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 en que se consideren por el especialista sólo variables reales, digamos, no siempre se encuentran fundamentos en la disciplina en particular (Geociencias, Medicina, etc.) para justificar el tratamiento uniforme de las variables; por ejemplo, en el pronóstico de la actividad solar, es imprescindible tener en cuenta el año en que ocurre, la variable "año" no puede analizarse mediante el criterio de intervalos, ya que la actividad solar sigue una función sinusoidal de período 11 años, luego dos años consecutivos nunca podrían ser considerados análogos y si los que tengan una diferencia de 11 años. Además, al construir los intervalos el algoritmo demanda el cumplimiento de una serie de reglas que no tienen sentido en los problemas reales, como en el número de intervalos, exigiendo que sea mínimo de 5 y máximo 20 y que la cantidad de valores que exista en cada intervalo sea mínimo de 5.

5) Utiliza como función de semejanza la igualdad estricta.

De la misma manera que los criterios de comparación, las funciones de semejanza no tienen porqué limitarse a un sólo tipo. En este caso el algoritmo maneja como función de semejanza la igualdad estricta, es decir, 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 completas de los objetos sean iguales; sino sólo algunos de los valores de sus descripciones 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) No clasifica objetos nuevos, es decir, objetos que no están en la matriz de aprendizaje.

El algoritmo sólo clasifica los objetos que se encuentran en la matriz de aprendizaje, en el sentido, de estructurarlos en ciertas categorías(típico atípico, etc.). No obstante, en muchos problemas de la realidad lo que se quiere es clasificar objetos que no se encuentren dentro de 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 raros que se cumplan en la práctica.

1.3 Algoritmo para clasificación y análisis de datos "TC" (L. Vega)

Como vimos el algoritmo que se presentó en la sección anterior tiene una serie de limitantes que hacen que no se pueda aplicar a muchos de los problemas reales, es por eso que en esta sección se proponen algunas modificaciones hechas por L.Vega[2]que elimiman dichas limitantes 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, pués estará basada en los pesos informacionales de los rasgos y objetos, el peso diferenciante de los objetos, así como de su semejanza. Todos estos conceptos serán definidos más adelante.

En este algoritmo como ya se mencionó, 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, 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, lo cual nos dará en forma graduada el contraste de un objeto dependiendo de qué tan típico sea dicho objeto para cada una de las clases.

El plantemiento del problema de clasificación se hará de la misma manera que en la sección anterior, donde toda la información se representa simbólicamente por medio de la matriz de aprendizaje (MA) (fig. 1). Aunque una diferencia es que aquí las variables pueden ser de cualquier tipo y se considera la posibilidad de ausencia de información.

Al principio de esta sección se dijo que el concepto de tipicidad está relacionado con el peso informacional y diferenciante de los rasgos y objetos, sin embargo, para poder definirlos es necesario introducir previamente las definiciones de criterio de comparación y función de semejanza.

Denominaremos *criterio de comparación* δ_i a la función con la cual podemos comparar los valores que un mismo rasgo x_i. toma en un par de objetos. 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. En todos los casos δ_i

estará definido sobre M_i x M_i y tomará valores en dependencia de qué tipo de criterio de comparación sea. Formalmente, se define el criterio de comparación de la siguiente manera:

$$_i: M_i \times M_i \rightarrow V_i$$

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 δ_i es una métrica o si la respuesta de la comparación de dos valores es por ejemplo de naturaleza difusa.

Los criterios de comparación forman la base sobre la que definiremos a las *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 cada una de las variables que intervienen en la descripción de los objetos. Igualmente 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.

Una vez definidos los criterios de comparación y la función de semejanza podemos definir lo que es el peso informacional de rasgos y objetos, así como el peso diferenciante de los objetos.

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 pudiera no ser importante para determinar el tipo de enfermedad que puedan tener, sin embargo, la presencia de un determinado síntoma quizá sería más importante en el proceso de clasificación; así, la variable síntoma x será más importante que la variable color de piel. De esta forma 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 mencionados, 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, ésta medida puede ser de utilidad en algunos problemas particulares 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 más importantes en el proceso de clasificación.

De la misma manera que el peso informacional de los rasgos, en muchos problemas de Reconocimiento de Patrones se hace necesario establecer una medida que permita hacer una diferenciación entre la información proveniente de uno u otro 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 $\delta_i(O_t, O_v) \in D_i$ significa que O_t 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_j a la magnitud

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

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

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, ya que se van sumando los pesos informacionales de los rasgos en los cuales el objeto O es semejante con los objetos de la clase K_j. Es importante mencionar que si al comparar los valores existe ausencia de información el peso informacional de la variable no se suma. Ya que si se tomara en cuenta, se podría pensar que en medida que exista más ausencia de información el peso informacional de los objetos aumenta, hecho que no parece razonable.

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 las que el mismo no fue asociado, es decir, a las clases que son diferentes de la clase 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

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

siendo $\overset{i}{j}(O) = \left|\left\{O_t \in (MA - K_j^{'}) \ / \ \delta_i(O, O_t) \notin D_i\right\}\right|$

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. Al igual que en el peso informacional, aquí se van sumando los pesos informacionales de los rasgos en los cuales el objeto O no se asemeja a los objetos de las clases diferentes a K_j.

Estas dos magnitudes pretenden caracterizar de alguna manera cuánto se parece O 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 pueden calcularse para cada objeto O del universo y pudieran utilizarse directamente para decidir acerca de la pertenencia de un objeto a clasificar en alguna clase, en este sentido, el clasificador que se propone considera únicamente los pesos de los objetos de MA.

De modo que a $O \in K_j^{'}$ 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 a continuación, aunque en realidad falta mencionar los tipos de objetos que forman cada una de las clases. Si recordamos, en la sección anterior se dijo que los objetos que constituían a cada una de las clases podían estar divididos en cuatro tipos.

I) Objetos propios. II) Objetos generales.

III) Objetos atípicos IV) Objetos no propios

Esta división se hace dependiendo de las medidas de tipicidad y contraste que presentan cada uno de los objetos. Pero recordemos también que esta división se hizo cuando el algoritmo se aplicó en la clasificación de anomalías dentro de la Geología. Sin

embargo, ahora el algoritmo se presenta de una manera más general, ya que los tipos de objetos que tendremos dependerán del problema que se quiera resolver. En el algoritmo "TC" 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 $T_{0,s}$, $C_{0,s}$. En principio diremos que los objetos están definidos dentro de los cuatro tipos mencionados anteriormente y que las reglas $T_{0,s}$, $C_{0,s}$ se obtienen de la misma manera que en el algoritmo de Diordenko L. y Vaskovskii B.

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

Descripción del algoritmo "TC"

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 δi y los pesos informacionales $p(x_i)$ para cada una de las variables, así como la función de semejanza β para la comparación entre objetos, es imortante mencionar que toda esta información se obtiene en el proceso de modelación del problema que se quiera resolver.

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 $T_{0,s}$, $C_{0,s}$. 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 de la sección anterior.

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 matriz que se muestra en la fig.4.

		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'_ℓ	⋮	⋮	⋮
	O_m	$PI_\ell(O_m)$	$PD_\ell(O_m)$

fig. 4 Matriz de pesos informacionales y diferenciantes de los objetos de MA

Paso 4) 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, mediante la siguiente expresión, que está basada en las medidas de semejanza y diferencia por objeto.

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

donde :

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

$T_{j,1}(O_h)$ 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

$$T_{j,0}(O_h) = \frac{1}{\left|K_j^{'}\right|} \sum_{\substack{Oi \in K_j^{'} \\ \beta(O_h, Oi) \notin D}} PD_j(O_i) \quad \text{para } h=1, \ldots, m$$

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.

En este sentido, la tipicidad nos refleja en qué medida el objeto O_h es representativo(tipico) para la clase K_j, tomando en consideración la semejanza con respecto a los objetos de su clase a través de la expresión $T_{j,1}(O_h)$, la no semejanza con

respecto a los objetos de las clases restantes utilizando la expresión $T_{i,0}(O_h)$, la magnitud en que se diferencia de los objetos de su clase $T_{j,0}(O_h)$ y la semejanza con respecto a los objetos de las clases restantes con la expresión $T_{i,1}(O_h)$. Por tanto a medida que el objeto O_h se paresca más a los objetos de la clase K_j y se diferencie de los objetos de las clases restantes, será más tipico, de manera contraria la tipicidad será menor.

Para cada objeto se calculan ℓ valores de la tipicidad T en correspondencia con la cantidad de clases como se muestra en la fig. 5.

	K_1'	$\cdots$	K_ℓ'
O_1	$T_1(O_1)$	$\cdots$	$T_\ell(O_1)$
O_2	$T_1(O_2)$	$\cdots$	$T_\ell(O_2)$
$\vdots$	$\cdots$	$\cdots$	$\cdots$
O_m	$T_1(O_m)$	$\cdots$	$T_\ell(O_m)$

fig. 5 Tipicidades de los objeto para cada una de las clase

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 $r \leq \ell$, ya que el máximo se puede alcanzar en más de una clase. Por lo tanto denotaremos K_r^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:

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

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. En este sentido, a medida que las tipicidades de las clases donde el valor de $T_j(O_h)$ no es máximo, se dispersan con respecto a la tipicidad máxima el contraste del objeto será mayor, por el contrario si estas medidas no se alejan mucho de la tipicidad máxima el contrate será menor.

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 MA sean más típicos para una clase distinta a la que 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.

Una regla de solución es clasificar a los objetos en aquellas clases donde se haya obtenido la tipicidad máxima. Nótese que la tipicidad máxima no necesariamente es única, lo que permite la multiclasificación de los objetos.

Resultados previos

Hasta este punto se han dado los primeros pasos en el desarrollo del algoritmo de clasificación y análisis de datos "TC", obteniéndose algunos resultados importantes como son:

- Utilizar cualquier tipo de variables para describir a los objetos admisibles.
- Tener diferentes criterios de comparación de tipo booleanos para cada una de las variables.
- 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 en el algoritmo de Diordenko.
- Considerar diferentes tipos de funciones de semejanza booleanas. La semejanza no está ligada estrictamente a la igualdad.
- Considerar la importancia informacional de los rasgos y objetos, generando ventajas dentro de la clasificación.
- 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.
- Tener s diferentes tipos de objetos con $s \geq 2$, permitiendo que la cantidad de objetos sea determinada en función de las necesidades del problema.

No obstante que en este primer acercamiento al algoritmo se han obtenido resultados que resuelven algunas de las limitantes, del algoritmo de Diordenko L. y Vaskosvskii B. [1], planteadas al inicio de éste trabajo, subsisten otras que deben ser consideradas, por ejemplo:

- que la tipicidad y el contraste puedan ser calculados haciendo un análisis multivariado de los datos y no univariado como se desarrolló hasta el momento.
- tomar en cuenta clases difusas, es decir, cuando los objetos pertenezcan a cada una de las clases en cierto grado ya que hasta ahora sólo se ha considerado que los objetos pertenenzcan o no a una clase, mediante un tratamiento booleano que permite la multiclasificación.

- que las funciones de semejanza y los criterios de comparación no sólo sean de tipo booleano.

- obtener una medida de contraste para cada objeto con respecto a cada una de las clases.

- proponer algunos mecanismos por medio de los cuales se pueda establecer cómo están estructuradas cada una de las clases, sin tener que establecer a priori los tipos de objetos en los cuales serán clasificados los objetos que conforman MA.

En el siguiente capítulo daremos un algoritmo que toma los primeros resultados obtenidos hasta el momento y resuelve estos últimos puntos.

Capítulo 2

Algoritmo para clasificación y análisis de datos TC+

2.1 Conceptos básicos.

En el capítulo anterior se dió un primer acercamiento al algoritmo de clasificación y análisis de datos que resuelve algunos de los objetivos planteados al principio de este trabajo, sin embargo, quedan algunos puntos que no han sido considerados. En este capítulo se propone un algoritmo que se basa en los resultados obtenidos previamente agregando una serie de conceptos nuevos que nos permiten incrementar dichos resultados eliminando todas las limitantes expuestas, cubriéndose así los objetivos planteados inicialmente.

Una de las limitaciones era no considerar que los objetos de MA pudieran pertenecer a cada una de las clases en cierto grado y también se mencionó que en los problemas reales el especialista no siempre puede, con toda certeza, afirmar si un objeto *pertenece* a una cierta clase o no, sino que valora de alguna manera cierto *"grado de pertenencia"*, por tanto las clases se vuelven imprecisas. Resulta entonces necesario definir el significado de grado de pertenencia. Para lo cual daremos la definición haciendo la comparación con la teoría de conjuntos clásicos y difusos.

En el caso de los conjuntos podemos decir que si X es un conjunto clásico, denominado universo, cuyos elementos están denotados por x; la pertenencia de x a un conjunto A, donde $A \subseteq X$, es vista como una función característica μ_A de X tal que:

$$\mu_A(x) = \begin{cases} 1 & \text{si y sólo si} \quad x \in \\ \\ 0 & \text{si y sólo si} \quad x \notin A \end{cases}$$

{0,1} es denominado *conjunto evaluativo*

Si el conjunto evaluativo es definido como el intervalo [0,1], entonces A es denominado un *conjunto difuso* [5] y $\mu_A(x$ el *grado de pertenencia* de *x* a A

Una vez que definimos el grado de pertenencia de un elemento en un conjunto difuso, haremos el planteamiento formal del problema de clasificación y análisis de datos.

Sea M un universo de objetos admisibles $O_1,\cdots, O_m$ y $\tilde{K}_1,\ldots,\tilde{K}_\ell$ una familia de subconjuntos difusos de M a los que llamaremos clases (en particular pudiera ser una ℓ-partición difusa de M [6]). A cada objeto O_i de M puede asociarse un ℓ-uplo de pertenencia $(O_i) = (\mu_{i,1},\ldots, \mu_{i,\ell})$, de modo que $\mu_{i,j} = \mu_j(O_i)$ denota el grado de pertenencia del objeto O a la clase $\tilde{K}_j$, con j=1, ... ,ℓ. 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. 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. La información de las descripciones de los objetos seguirán conforman las filas de *matriz de aprendizaje* (MA).

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 cuál es el grado de pertenencia de O para cada una de las $K_1,\cdots, K_\ell$.

El algoritmo seguirá basado en los conceptos de tipicidad y contraste, así como en los conceptos de peso informacional y diferenciante de objetos. Sin embargo la manera de calcular estas medidas se hará diferente ya que en primer lugar se considerará que los objetos pertenecen a cada una de las clases en cierto grado y en segundo lugar, la comparación entre objetos se hará por medio de semejanzas parciales y no por una función de semejanza total, esto es, que para decir que dos objetos son semejantes se tiene que verificar si son semejantes en ciertos conjuntos de variables por medio de las funciones de semejanza que se definan para los mismos; lo que nos permite hacer un análisis multivariado ya que se consideran subconjuntos de variables para obtener la semejanza entre objetos. Cabe mencionar que en los problemas reales muchas veces es

más importante considerar un conjunto de características y no cada una de ellas por separado, por ejemplo, en el caso de la Medicina, para un médico sea más importante el saber que un paciente tiene la combinación de diarrea, fiebre y dolores de estómago que saber que tiene fiebre o diarrea. A partir de las ideas de conjuntos de apoyo y clases difusas daremos los conceptos que las formalizan.

Sea $I(O)=(x_1(O),\cdots, x_n(O))$ la descripción del objeto O. Llamaremos $\overline{\omega}$-*parte* de la descripción del objeto O y la denotaremos $\overline{\omega}I(O)$ al vector r-dimensional ($r \leq n$) $(x_{i_1}(O),\cdots, x_{i_r}(O))$, donde $\overline{\omega} = (\omega_1,\ldots,\omega_n)$, $\omega_j = 1$ para $j = i_1\cdots i_r$ y $\omega_p = 0$ para $p \neq j$. Por *sistema de conjuntos de apoyo* Ω_A entenderemos un conjunto de $\overline{\omega}$-partes.

Sean dados MA, β, Ω_A

Definición 1.- Llamaremos *peso informacional multivariado de un objeto i* en la clase $\widetilde{K}_j$ a la magnitud

$$PI_j^{MV}(O_i) = \frac{1}{|\Omega_A| \sum_{\Omega\in\Omega_A} P(\Omega)} \sum_{\Omega\in\Omega_A} PI_j^{MV}(\overline{\omega}\, O_i) \qquad (2.1)$$

siendo $|\Omega_A|$ la cantidad de conjuntos de apoyo; $P(\Omega)$ es un parámetro de ponderación asociado a cada $\overline{\omega}$-parte y $PI_j^{MV}(\overline{\omega}\, O_i)$ es el peso informacional de cada $\overline{\omega}$-parte del objeto, la cual se define de la siguiente manera:

$$PI_j^{MV}(\overline{\omega}\, O_i) = \frac{P(\Omega)}{\left|\widetilde{K}_j'\right|} \sum_{p=1}^{m} \beta(\overline{\omega} \,|\, \mu_{i,J}, \overline{\omega}\, O_p \,|\, \mu_{pJ})$$

donde $\widetilde{K}_j' = \left\{O_i \,\middle|\, \mu_{i,J}, i = 1,\ldots,m\right\}$,

$\beta(\overline{\omega}O_i \,|\, \mu_{i,j}, \overline{\omega}O_p \,|\, \mu_{p,j}) = f(\mu_{i,J}, \mu_{p,J})\beta(\overline{\omega}\, O_i, \overline{\omega}\, O_p)$, tal que la función f pudiera ser por ejemplo, el promedio de las pertenencias de los objetos a $\widetilde{K}_j$.

$$f(\mu_{i,J}, \mu_{p,J}) = \frac{\mu_{i,J} + \mu_{p,j}}{2}$$

El peso informacional de un objeto nos proporciona la medida de semejanza de dicho objeto con respecto a una clase dada, utilizando la suma de las semejanzas parciales de las combinaciones de rasgos que conforman al conjunto de apoyo.

Definición 2.- Denominaremos *peso diferenciante multivariado de un objeto* a la magnitud

$$PD_j^{MV}(O_i) = \frac{1}{|\Omega_A| \sum_{\Omega \in \Omega_A} P(\Omega)} \sum_{\Omega \in \Omega_A} PD_j^{MV}(\overline{\omega}\, O_i) \qquad (2.2)$$

donde $PD_j^{MV}(\overline{\omega}\, O_i)$ representa el peso diferenciante de cada $\overline{\omega}$-parte del objeto, el cual se define de la siguiente manera:

$$PD_j^{MV}(\overline{\omega}\, O_i) = \frac{P(\Omega)}{|\widetilde{CK}_j'|} \sum_{\substack{s=1 \\ s \neq j}}^{\ell} \sum_{p=1}^{m} (1 - \beta(\varpi O_i | \mu_{i,s},\ \varpi O_p | \mu_{p,s}))$$

tal que $\widetilde{CK}_j' = \{O_i | \mu_{i,t}, i = 1,\ldots,m;\ t \neq j,\ t = 1,\ldots,m\}$

El peso diferenciante de un objeto con respecto a una clase dada representa la medida de diferenciación de dicho objeto con las clases restantes.

Hasta este punto hemos dado los conceptos básicos para el algoritmo que será descrito en la siguiente sección, sin embargo, falta mencionar los tipos de objetos que forman cada una de las clases. En la siguiente sección se proponen 4 maneras de determinar los tipos de objetos.

2.2 Descripción del algoritmo TC+

Paso 1) Determinación de parámetros de entrada
Determinación de los criterios de comparación δ_i, correspondiente a cada una de las variables; del sistema de conjuntos de apoyo, Ω_A; pesos informacionales de los conjuntos de apoyo $P(\Omega)$ y de la función de semejanza β para cada uno de los conjuntos de apoyo, en este sentido, es importante mencionar que la función puede ser la misma para cada conjunto (que es lo que hasta ahora se ha encontrado en la práctica).

Paso 2) Obtención de pesos informacionales y diferenciantes.

En este paso se calcula el peso informacional $PI_j^{MV}(O)$ y diferenciante $PD_j^{MV}(O)$ de todos los objetos de MA, utilizando las fórmulas (2.1) y (2.2) respectivamente, definidas en la sección anterior.

Paso 3) Cálculo de la tipicidad

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

$$T_j^{MV}(O_h) = \frac{T_{j,1}^{MV}(O_h) + \sum_{\substack{i=1 \\ i \neq j}}^{\ell} T_{i,0}^{MV}(O_h)}{T_{j,0}^{MV}(O_h) + \sum_{\substack{i=1 \\ i \neq j}}^{\ell} T_{j,1}^{MV}(O_h)} \qquad (2.3)$$

Donde las expresiones $T_{j,1}^{MV}$ y $T_{j,0}^{MV}$ se determinan de la siguiente manera:

$$T_{j,1}^{MV}(O_i) = \begin{cases} \dfrac{1}{\left|K_j'\right| \sum_{p=1}^{m} PI_j^{MV}(O_p)} \sum_{\Omega \in \Omega_A} \sum_{p=1}^{m} \beta(\varpi O_i \left| \mu_{is}, \varpi O_p \right| \mu_{ps}) PI_j^{MV}(O_p) & \text{si } \mu_{i,j} = \\ PI_j^{MV}(O_i), & \text{si } \mu_{i,j} \neq 0 \end{cases}$$

$$T_{j,0}^{MV}(O_i) = \begin{cases} \dfrac{1}{\left|\widetilde{K}_j'\right| \sum_{p=1}^{m} PI_j^{MV}(O_p)} \sum_{\Omega \in \Omega_A} \sum_{p=1}^{m} (1 - \beta(\varpi O_i \left| \mu_{i,j}, \varpi O_p \right| \mu_{p,j})) PI_j^{MV}(\varpi\, O_p), & \text{si } \mu_{i,j} = 0 \\ PI_j^{MV}(O_i), & \text{si } \mu_{i,j} \neq 0 \end{cases}$$

$T_{j,1}^{MV}$ es el promedio de los pesos informacionales de los objetos que, estando en $\widetilde{K}_j'$, son semejantes a O_h y como $PI_j(O_i)$ es una medida de cuánto se asemeja O_i 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 los objetos de la clase K_j.

$T_{j,0}(O)$ nos da el promedio de los pesos informacionales de los objetos que, estando en K_j' no son semejantes a O; y como $PI_j(O_i)$ es una medida de cuánto se asemeja O_i 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'

$$K_{J,0}^{MV}(O_i) = \begin{cases} \dfrac{1}{\left|K_J^{'}\right| \sum\limits_{O_p \in K_J^{'}} PD_J^{MV}(O_p)} \sum\limits_{\Omega \in \Omega_A} \sum\limits_{p=1}^{m} (1 - \beta(\overline{\omega} O_i \left| \mu_{i,j}, \overline{\omega} O_p \right| \mu_{p,j})) PD_J^{MV}(\overline{\omega}\, O_p), & \text{si } \mu_{i,j} = 0 \\ PD_J^{MV}(O_i), & \text{si } \mu_{i,j} \neq 0 \end{cases}$$

Al igual que la tipicidad se obtendrán ℓ valores del contraste para cada objeto en correspondencia con la cantidad de clases.

Paso 5) Análisis de los objetos de MA.

Aquí se hará la clasificación de los objetos de MA en los diferentes tipos definidos. Como ya mencionamos se cuenta con 4 maneras diferentes de analizar los tipos de objetos. Las cuales serán definidas después del algoritmo. Sin embargo, podemos considerar este paso de igual manera que en el paso 6 del algoritmo TC, descrito en la sección 1.3. Con esto se finaliza el algoritmo.

Hasta el paso 5 se hace únicamente el análisis de la estructura interna de MA, que por sí sola conforma una de las aplicaciones del algoritmo, sin embargo, si se desea clasificar objetos que no se encuentran dentro de MA se lleva a cabo del paso 1 al 3 y se salta al paso 6.

Paso 6) Clasificación de objetos que no están en MA
En este paso se clasifican los objetos que no se encuentran en MA, para lo cual se obtiene la tipicidad por medio de la fórmula (2.3), con respecto a cada una de las clases que conforman a MA para cada uno de los objetos a clasificar. Los objetos serán clasificados en todas las clases con un cierto grado, dependiendo de la tipicidad que se obtenga para cada una de las clases. Es importante mencionar que el grado de pertenencia puede ser cero. Este es el último paso del algoritmo. [FIN]

Es importante mencionar que todas las definiciones que fueron dadas anteriormente, consideran que los objetos pertenecen a cada una de las clases en cierto

grado y que las semejanzas entre los objetos se determinan de una manera multivariada, es decir, por medio de la suma de las semejanzas parciales. Sin embargo, existen dos casos que podemos considerar particulares del caso difuso y multivariado:

1. Los objetos pertenecen a las clases con grado de pertenencia 1 ó 0 lo que implica que nos encontramos ante el caso booleano (duro).

2. Si consideramos que el sistema de conjuntos de apoyo es $\{\{x_1,\cdots, x_n\}\}$ entonces se trata de un análisis univariado.

Determinación de los tipos de objetos

Cuando se hizo la descripción del algoritmo se mencionó que existen 4 maneras de definir los tipos de objetos, sin embargo, en realidad lo que se hace es ordenar los objetos con respecto a cada una de las clases sobre la base de la tipicidad y el contraste, hecho que nos permite conocer la estructura de cada una de las clases. Es importante recordar que se tienen ℓ-uplos de tipicidades $(T_1^{MV}(O),\ldots,T_\ell^{MV}(O))$ y contrastes $(K_1^{MV}(O),\ldots,K_\ell^{MV}(O))$. A continuación se explican cuáles son los cuatro tipos de ordenamientos.

I. En función de las tipicidades de los objetos de manera decreciente.

II. En función del contraste de los objetos en forma decreciente.

Como se puede observar estas dos primeras variantes, únicamente consideran para el ordenamiento una de las magnitudes, ya sea la tipicidad o el contraste. Sin embargo estos ordenamientos pudieran no ser representativos, ya que existe la posibilidad de que un objeto (O_1) pudiera anteceder a un objeto (O_2) y que la diferencia entre las tipicidades de los dos objetos fueran mínimas, pero que la diferencia en su contraste fuera alta y que el contraste mayor lo tuviera el objeto O_2.

Por ejemplo:

Sea $T_j^{MV}(O_1) = 0.7$, $K_j^{MV}(O_1) = 0.2$ la tipicidad y el contraste del objeto O_1 con respecto a la clase $\tilde{K}_j'$ y $T_j^{MV}(O_2) = 0.6$, $K_j^{MV}(O_2) = 0.8$ la tipiciadad y el contraste del objeto O_2 con respecto a la clase $\tilde{K}_j'$

Es evidente que si ordenáramos a los objetos con respecto a la tipicidad el objeto O_1 estaría antes que el O_2 sin importar que O_2 tuviera un mayor contraste y que la diferencia de tipicidades entre ambos fuera pequeña.

Con este ejemplo podemos ver que en los dos ordenamientos pasados existen algunos problemas que nos pudieran dar una falsa idea de la estructuración de cada una de las clases, es por eso, que se propone como tercera opción considerar ambas magnitudes.

III.Considerar la tipicidad y el contraste

Sean O_1 y O_2 dos objetos de MA. Consideremos entonces todas las posibles comparaciones de las tipìcidades y contrastes de ambos objetos, en términos de mayor que, menor que e igual.

1) El caso más sencillo es cuando la tipicidad de ambos objetos es igual y el contraste de ambos también es igual. Lo que implica que O_1 se coloca en el mismo lugar que O_2

2) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) = K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) < T_j^{MV}(O_2)$
entonces O_1 sucede a O_2

3) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) = T_j^{MV}(O_2)$
entonces O_1 antecede a O_2

4) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_1)$ donde $T_j^{MV}(O_2) > T_j^{MV}(O_1)$
entonces O_1 sucede a O_2

5) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) > T_j^{MV}(O_2)$
entonces O_1 antecede a O_2

6) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) < T_j^{MV}(O_2)$

6.1) Si $K_j^{MV}(O_1) = K_j^{MV}(O_2)$ entonces O_1 sucede a O_2

*6.2) Si $K_j^{MV}(O_1) > K_j^{MV}(O_2)$?

6.3) Si $K_j^{MV}(O_1) < K_j^{MV}(O_2)$ entonces O_1 sucede a O_2

7) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) < K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) = T_j^{MV}(O_2)$
entonces O_1 sucede a O_2

8) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) < K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) > T_j^{MV}(O_2)$
8.1) Si $K_j^{MV}(O_1) = K_j^{MV}(O_2)$ entonces O_1 antecede a O_2
8.2) Si $K_j^{MV}(O_1) > K_j^{MV}(O_2)$ entonces O_1 antecede a O_2
* 8.3) Si $K_j^{MV}(O_1) < K_j^{MV}(O_2)$?

9) $T_j^{MV}(O_1) = K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) < K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) < T_j^{MV}(O_2)$
entonces O_1 antecede a O_2

0) $T_j^{MV}(O_1) > K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) = T_j^{MV}(O_2)$
10.1) Si $K_j^{MV}(O_1) = K_j^{MV}(O_2)$ entonces O_1 es ordenado en el mismo lugar de O_2
10.2) Si $K_j^{MV}(O_1) > K_j^{MV}(O_2)$ entonces O_1 antecede a O_2
10.3) Si $K_j^{MV}(O_1) < K_j^{MV}(O_2)$ entonces O_1 sucede a O_2

1) $T_j^{MV}(O_1) > K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) > T_j^{MV}(O_2)$
11.1) Si $K_j^{MV}(O_1) = K_j^{MV}(O_2)$ entonces O_1 antecede a O_2
11.2) Si $K_j^{MV}(O_1) > K_j^{MV}(O_2)$ entonces O_1 antecede a O_2
* 11.3) Si $K_j^{MV}(O_1) < K_j^{MV}(O_2)$?

2) $T_j^{MV}(O_1) > K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) > K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) < T_j^{MV}(O_2)$
12.1) Si $K_j^{MV}(O_1) = K_j^{MV}(O_2)$ entonces O_1 sucede a O_2
*12.2) Si $K_j^{MV}(O_1) > K_j^{MV}(O_2)$?
12.3) Si $K_j^{MV}(O_1) < K_j^{MV}(O_2)$ entonces O_1 sucede a O_2

3) $T_j^{MV}(O_1) > K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) < K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) = T_j^{MV}(O_2)$
entonces O_1 sucede a O_2

4) $T_j^{MV}(O_1) > K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) < K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) > T_j^{MV}(O_2)$

14.1) Si $K_j^{MV}(O_1) = K_j^{MV}(O_2)$ entonces O_1 antecede a O_2

14.2) Si $K_j^{MV}(O_1) > K_j^{MV}(O_2)$ entonces O_1 antecede a O_2

* 14.3) Si $K_j^{MV}(O_1) < K_j^{MV}(O_2)$?

5) $T_j^{MV}(O_1) > K_j^{MV}(O_1)$ & $T_j^{MV}(O_2) < K_j^{MV}(O_2)$ donde $T_j^{MV}(O_1) < T_j^{MV}(O_2)$
entonces O_1 sucede a O_2

es importante notar, que en los casos en los que aparece un * no se puede determinar cuál objeto antecede a otro, es por eso, que este ordenamiento no es total sino parcial. En este sentido se podría pensar en la variante de establecer algún parámetro de comparación dado por el experto, el cual sería determinante en el ordenamiento de los objetos con *.

IV. Verficar cual es la relación entre la ecuación de la tipicidad y el contraste.

En ésta última propuesta se hace un análisis de la relación que existe entre la tipicidad y el contraste de manera analítica, haciendo una serie de despejes con respecto a las ecuaciones de la tipicidad y el contrate, se puede poner a la tipicidad en función del contraste, obteniéndose la siguiente ecuación:

$$T_j^{MV}(O) = K_j^{MV}(O)\left[\frac{K_{j,0}^{MV}(O) + \sum_{1}^{\ell} T_{i,1}^{MV}(O)}{T_{j,0}^{MV}(O) + \sum_{1}^{\ell} T_{i,1}^{MV}(O)}\right] + \frac{T_{j,1}^{MV}(O) - K_{j,1}^{MV}(O)}{T_{j,0}^{MV}(O) + \sum_{1}^{\ell} T_{i,1}^{MV}(O)}$$

Esta ecuación corresponde a la ecuación de una recta, por tanto, si ponemos a la tipcidad en el eje de las ordenadas y al contraste en el eje de las abscisas, podemos representar a los objetos gráficamente como rectas en el plano y generar una regla, para encontrar un ordenamiento total de los objetos, sobre la base de su representación geométrica en el plano. Es importante mencionar que como la tipicidad y el contraste solo toman valores positivos, se toma en cuenta únicamente el primer cuadrante.

Sean las rectas $L(O_1)$, $L(O_2)$ correspondientes a los objetos O_1 y O_2 respectivamente, diremos que O_1 antecede a O_2 si:

a) las rectas no se intersectan y $L(O_1)$ esta por encima de $L(O_2)$. Como se muestra a continuación en la fig 6.

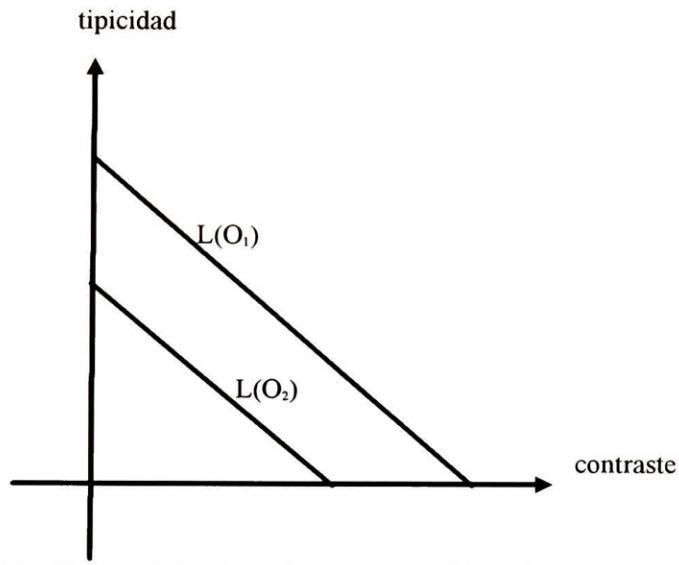


fig 6 Ordenamiento cuando las rectas no se intersectan.

b) las rectas $L(O_1)$, $L(O_2)$ se intersectan y el área (positiva) del triángulo que se forma por encima del punto de intersección es mayor que el área (negativa) del triángulo que se forma por debajo de la intersección. Como se muestra en la figura 7.

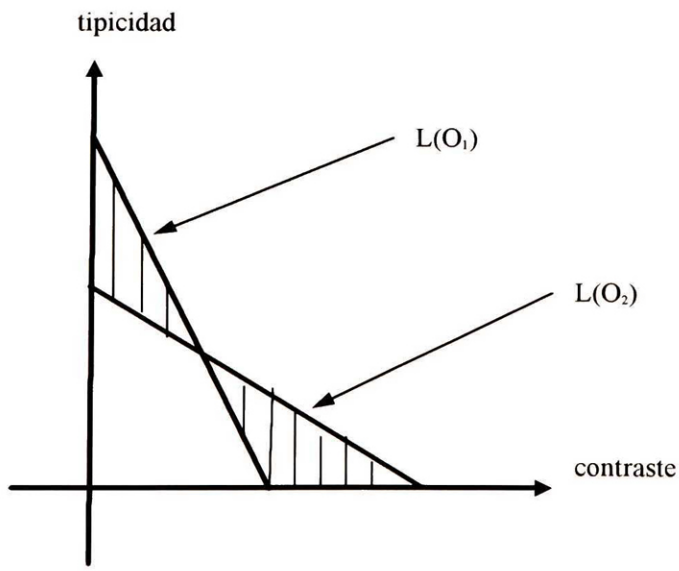


fig 7 Ordenamiento cuando las rectas se intersectan.

Si ninguna de estas opciones sucede entonces O_2 antecede a O_1 a excepción de que la recta de O_1 sea igual a la recta de O_2, o las áreas sean iguales, en cuyo caso son ordenados en el mismo lugar.

Como se puede ver la última opción nos proporciona un ordenamiento total y además es sobre la base de ambas medidas (tipicidad y contraste).

La selección de una de estas formas de estructuración de los objetos de MA, está en función de las características específicas del problema que se está resolviendo.

Resultados

Los resultados obtenidos en este trabajo conforman un algoritmo de clasificación y análisis de datos, que permite:

* que todas las variables puedan ser de cualquier tipo

* que las variables no proporcionan la misma información

* ausencia de información

* definir diferentes criterios de comparación para cada una de las variables

* trabajar en medios difusos, es decir, cuando los objetos pertenecen a cada una de las clases en cierto grado.

* que la semejanza entre objetos pueda ser determinada por medio de semejanzas parciales, es decir, poder hacer un análisis multivariado de los datos

* ordenar los objetos en una clase sobre la base de su tipiciadad y contraste.

* clasificar objetos que no se encuentran en MA

Capítulo 3
Implantación computacional.

3.1 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[6] que son: (1) establecer un criterio de selección (2) determinar cuál o cuáles lenguajes se acoplan de mejor manera a las características del problema que será resuelto computacionalmente, sobre la base del criterio establecido.

Una preselección del lenguaje de implantación es ubicar de manera rápida, aquellos que pudieran acoplarse mejor a las características de nuestro problema y eliminar todos los que no están relacionados con él. Por ejemplo, si hablamos del desarrollo computacional de una aplicación de inteligencia artificial, lo más factible sería utilizar lenguajes funcionales como PROLOG o LISP, pero si nuestro problema está relacionado con base de datos tendremos que seleccionar lenguajes como DBASE o FOXPLUS, y evidentemente descartaremos a PROLOG y LISP. El descartar los lenguajes que no tienen relación con las características del problema es de gran ayuda, ya que de alguna manera se reducen las opciones 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, que pueden ser determinados considerando los siguientes puntos:

a) Portabilidad

b) Modularidad

c) Factores humanos

d) Factores técnicos

e) Rapidez

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 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 sentido, ya que sus módulos pueden ser compilados de manera individual, permitiendo detectar los errores que se pudieran generar en cada uno de los módulos de manera más rápida, sin tener que verificar el programa completo.

Es importante mencionar que el desarrollo de un sistema en módulos, además de hacer legible un programa y detectar los errores de una manera rápida nos permite la posibilidad de extender el sistema a futuro, es decir, de agregar más módulos, haciendo modificaciones mínimas al programa principal; incluso nos da la posibilidad de usar módulos de otros programas que pudieran servir a nuestro sistema, sin la necesidad de volver a programarlos, es decir, reutilizar el 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 es un factor importante para el desarrollo de un sistema, sobre todo si se contempla el crecimiento del sistema a futuro.

Factores humanos

Otro factor es el humano y es el que se vincula a 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) no es clara y no está organizada de una manera que se pueda localizar rápidamente lo que se busca, entonces dicha documentación resultará inusitada.

La facilidad en el uso 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 puntos 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.

Rapidez

Llegamos al punto quizá menos importante para la evaluación, que es la rapidez. Este es un punto muy difícil de evaluar, ya que no sólo depende del lenguaje que utilicemos, sino también de muchos otros factores, como por ejemplo el programador, ya que si él hace manejos innecesarios de algunos recursos, como son el acceso a disco, a la memoria o al sistema, esto hará que el programa se vuelva lento. La rapidez de un lenguaje no puede ser evaluada tan fácilmente ya que involucra al factor humano.

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 (TC+), se seleccionó como lenguaje de programación el lenguaje C, las razones fueron 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. Por ser un lenguaje estructurado, permite planificar el sistema de manera modular, dando la posibilidad de dividirlo en pequeños problemas y a su vez considerando extensiones a futuro de nuevos módulos sin tener que hacer grandes modificaciones. En este sentido, es importante mencionar que se reutilizaron algunas funciones que fueron programadas para otro proyecto, sin tener que hacer grandes modificaciones (reutilización de código).

3. El punto más importante en la elección del lenguaje de programación fue 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.

4. Otro de los factores que también fue considerado para la elección del lenguaje fue 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.

3.2 Estructuras y funciones

Fundamentalmente podemos decir que el sistema está conformado por dos partes. 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 módulos a su vez está constituido por una serie de procedimientos.

En los siguientes párrafos daremos una explicación general de cada modulo y de las estructuras y funciones más importantes de ellos.

Analicemos el modulo de lectura. 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 conjuntos de apoyo y la pertenencia de cada uno de los objetos a cada una de las clases. Toda esta información se encuentra dentro de 4 archivos que son:

1) Descriptor de archivos.- En este archivo se guarda la información más importante correspondiente a las los atributos que describen a los objetos de la matriz de aprendizaje, como son; número de atributos, nombre de cada atributo, tipo, valencia(en caso de ser k-valente), peso informacional, criterio de comparación, etc. Además se tiene la información de la cantidad de clases de MA, el número de subconjuntos de apoyo y el nombre del archivo donde está guardada las descripciones estándar de cada uno de los objetos.

2) Archivo de datos de MA.- Este archivo contiene la información correspondiente a la descripción estándar de cada uno de los objetos.

3) Archivo de datos de pertenencias.- Aquí se tiene la pertenencia de cada uno de los objetos a cada una de las clases.

4) Archivo de datos de conjuntos de apoyo. Por último en este archivo se guarda la información de las variables que conforman a cada uno de los conjuntos de apoyo, así como, los pesos informacionales de los mismos.

Es importante mencionar que dentro del sistema no se crean ninguno de estos archivos 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 son leídos cuando se ejecuta el modulo de lectura y la información es colocada en 4 estructuras diferentes que son:

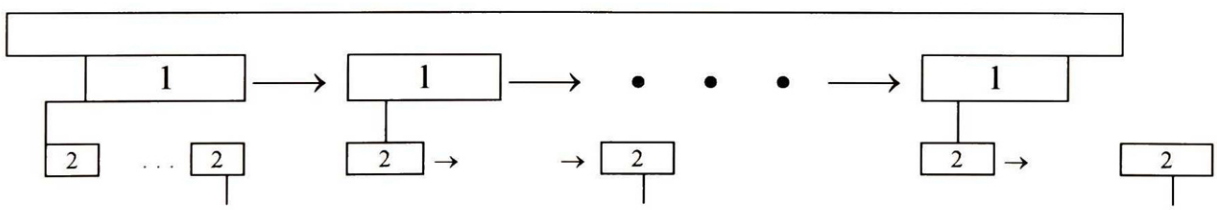
1) Arreglo de acceso dinámico el cual guarda la siguiente información:

nombre atribut. 1	tipo	valencia	criterio comparac.	ε de compar.	peso infor.	.	nombre atribut. n	tipo	valencia	criterio comparac	ε de compar.	peso infor.

2) Arreglo de acceso dinámico para la MA.

$x_1(O_1)$	$x_2(O_1)$		$x_n(O_1)$	$x_1(O_2)$	$x_2(O_2)$	. . .	$x_n(O_2)$	•	•	•	$x_2(O_2)$	. . .	$x_n(O_2)$

3) Lista circular para pertenencia de los objetos a las clases



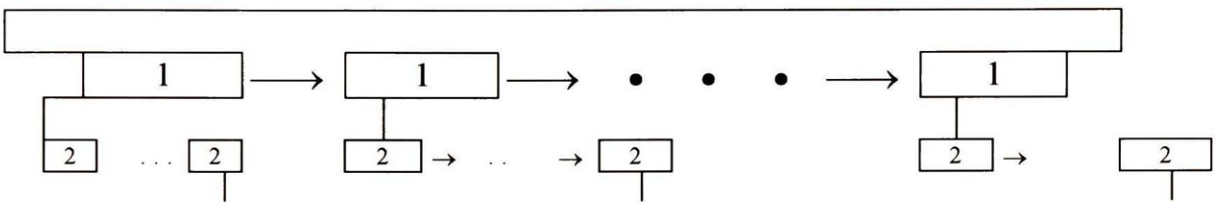
donde los nodos marcados con el número 1 contienen la siguiente información:

- Número de la clase
- apuntador a la lista de objetos
- apuntador a la clase siguiente

los nodos marcados con el número 2 contienen la información:

- número de objeto
- pertenencia a la clase

4. Lista circular para conjuntos de apoyo



donde los nodos marcados con el número 1 contienen la siguiente información:

- número del conjunto de apoyo
- cardinalidad del conjunto
- peso informacional
- función de semejanza
- épsilon de semejanza
- apuntador al siguiente conjunto
- apuntador a los índices de las variables que conforman al conjunto

los nodos marcados con el número 2 contienen los índices de las variables del conjunto de apoyo.

Una vez que se crearon y llenaron las estructuras, el proceso de lectura se termina y se pasa al módulo de la ejecución del algoritmo de clasificación y análisis de datos(TC+). Este modulo comienza calculando los pesos informacionales y diferenciantes de cada uno de los objetos de MA por medio de la función PI_PD, la cual contiene una función que se denomina SEMEJANZA en donde se verifica si los valores de un par de objetos correspondientes a un conjunto de apoyo son semejantes, con respecto a una cierta función de semejanza.

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

$PI(O_1)$	$PD(O_1)$	$PI(O_2)$	$PD(O_2)$	$\cdots$	$PI(O_m)$	$PD(O_m)$

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.

Los resultados obtenidos al ejecutar la función TIPICIDAD son guardados en un archivo y al mismo tiempo en un arreglo de acceso dinámico como el de los pesos informacionales y diferenciantes. De la misma manera es guardada la información del contraste de 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 parámetros de comparación $T_{0,s}$, $C_{0,s}$ que forman parte de la regla de solución para definir 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 leídos 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.

- analisis.exe

- defins.h

- lectura.c

- algoritmo.c

- descrip.dat

Si todos estos archivos se encuentran en el directorio entonces se debe digitar el nombre del programa ejecutable "análisis". Inmediatamente aparecerá una pantalla con un menú con dos opciones, que son 1) análisis, 2)clasificación. En cualquiera de los dos casos el programa se encargara de solicitar los nombres de los archivos donde se guarda la información de la matriz de aprendizaje, pertenencias de los objetos a cada clase y conjuntos de apoyo.

Si toda la información anterior esta bien entonces los resultados de aplicar el algoritmo serán puestos en un archivo con extensión .ANA si se solicito análisis de datos y .CLA si se solicito clasificar objetos, estos archivos llevarán el mismo nombre que el archivo donde se guarda la información de MA.

En caso de existir algún error en la información el programa desplegará un mensaje de error que determine donde está el mismo.

En los apéndices se da un ejemplo de la estructura de cada uno de los archivos de datos, así como los listados del sistema.

Conclusiones

Tomando en consideración que el objetivo y motivación de este trabajo, era 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 los objetivos propuestos se cumplieron de manera satisfactoria ya que el algoritmo permite:

- utilizar variables cualitativas y cuantitativas simultáneamente

- trabajar con diferentes criterios de comparación

- considerar ausencia de información en algunas variables de ciertos objetos de MA

- considerar que no todas las variables proporcionan la misma información

- utilizar de funciones de semejanza más flexible y por lo tanto más cercanas a la realidad

- que los objetos puedan pertenecer a cada una de las clases de MA en diferentes grados, es decir, abordar problemas de tipo difuso.

- que los objetos pueden ser ordenados para cada una de las clases con 4 criterios diferentes.

Resulta importante mencionar que para el desarrollo de este este trabajo se propusieron algunos conceptos nuevos como son la tipicidad y el contraste multivariado, el peso informacional y diferenciante multivariado; el concepto de ordenamiento de los objetos, tomando en cuenta que estos se pueden representar como rectas en el plano. Además se incorporaron otros conceptos que forman parte de la teoría de Reconocimiento de Patrones en el enfoque Lógico-combinatorio como son: el concepto de analogía entre objetos, el peso informacional de los rasgos,

el peso informacional univariado de los objetos y el criterio de analogía entre los rasgos. Por tanto se considera que sin el desarrollo y conjunción de los conceptos antes mencionados, no se hubieran alcanzado los objetivos de manera satisfactoria.

Finalmente se concluye el trabajo con la presentación del sistema computacional del algoritmo desarrollado (TC+).

Apéndice

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

Los listados están divididos en tres partes, que son:

A1.- Programa principal y definiciones.

A2.- Rutinas de lectura de datos.

A3.- Rutinas del algoritmo.

Dentro de la sección A1 se ncuentra el programa principal llamado analisis.exe, así ncomo las definiciones de las 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.

Finalmente la sección A3 contiene todas las rutinas con las cuales se lleva a cabo los pasos del algoritmo "TC++", las cuales se localizan en el archivo algorit.c.

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 clases{
                int clase;
                float pertenencia;
                struct clases *sig;
               };

struct objetos_pert{
                        int num_obj;
```

```
                        struct clases *apunta_clase;
                        struct objetos_pert *sig;
                       };

struct conjuntos_apoyo{
                                int numero;
                                int cardinalidad;
                                float peso_info;
                                int func_sem;
                                float epsilon;
                                int *indices;
                                struct conjuntos_apoyo *sig;
                       };

typedef struct clases NODO_CLA;
typedef struct objetos_pert NODO_OBJ;
typedef struct conjuntos_apoyo NODO_CONJ;
NODO_OBJ *pri_nom_obj;
NODO_CLA *pri_nom_cla;
NODO_CONJ *pri_nom_conj;
typedef char CADENA[10];
typedef char CADENA2[12];
CADENA *RASGOS_INFO;
CADENA *MA;
int *CARDINAL_CLASE;
/*CADENA *mat_obj_cla;
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 arch_pert_cla[12];
char token[25];
char buffer[10];
int token_num,indice;
int tot_obj;
int rasgonum;
int num_clase;
int tot_conj_apoyo;
int tot_obj_cla;
int f_semejanza;
int num_carac_info=6;
double umbral;
FILE *entrada;

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <stdlib.h>
#include <alloc.h>
#include <dos.h>
#include <math.h>
#include "defins.h"
#include "LECTURA.C"
#include "ALGORIT.C"


/*********************    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: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                             */
/*****************************************************************/


void main()
{
int resp;
char nom_arch3[12];
char ext2[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);
                  flushall();
                  TIPICIDAD(MA,mat,MA,tot_obj,pri_nom_rasg);
                  CONTRASTE(mat_tip_cont,tot_obj);
                  strcpy(nom_arch3, *BUSCA_NOM_ARCH(nom_arch,ext2));
                  GUARDA_ARCH(mat_tip_cont,tot_obj,num_clase+2,nom_arch3);
                  ANALI_OBJ(mat_tip_cont,MA,tot_obj);
                  free(MA);
                  free(mat);
                  free(mat_tip_cont);
                  resp=0;
                  break;

          case 2: LECTURA_DATOS();
                  CAL_PI_PD(MA,pri_nom_rasg);
                  flushall();
                  gotoxy(6,9);printf("Nombre del archivo de objetos a clasificar:");
                  scanf("%s",nom_archivo3);
                  gotoxy(6,11);printf("Numero de objetos que contiene el archivo: ");
                  flushall();
                  scanf("%d",&tot_obj_cla);
                  mat_obj_cla = (CADENA *) malloc (tot_obj * (rasgonum+1) * (sizeof (CADENA)));
                  LEE_ARCH_MA(nom_archivo3,mat_obj_cla,pri_nom_rasg,tot_obj_cla);
                  flushall();
                  TIPICIDAD(MA,mat,mat_obj_cla,tot_obj_cla,pri_nom_rasg);
                  CLASIFICA(tot_obj_cla,mat_tip_cont);
                  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.

```
/*********************  Funci¢n OBTEN_TOKEN()  *********************/
/* Esta funci¢n 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 n£mero correspondiente al caracter leido.            */
/*            Funciones auxiliares: Ninguna.                           */
/*********************************************************************/

int OBTEN_TOKEN(void)
{
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);

}

/*********************  Funci¢n VALOR_MATRIZ()  *********************/
/* Esta funci¢n se encarga de obtener el dato que se encuentra en la po-*/
/* sici¢n (ren, col) de una matriz de caracteres. Dicha matriz es de   */
/* acceso dinamico.                                                    */
/*                                                                     */
/*            Entrada:EL nombre de la matriz, la posici¢n 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 posici¢n (ren, col)  */
/*            Funciones auxiliares: NINGUNA                            */
```

```
/*************************************************************************/

CADENA *VALOR_MATRIZ(CADENA *matriz, int ren, int col, int REN)
{
  return (matriz[(ren * (REN+1)) + col]);
}


/*********************    Funci¢n PON_EN_MATRIZ()    *********************/
/* Esta funci¢n se encarga de poner en una matriz el dato que se encuen- */
/* tra en la posici¢n (ren, col) de la misma. Dicha matriz es de acceso  */
/* dinamico.                                                             */
/*                                                                       */
/*            Entrada:EL nombre de la matriz, la posici¢n 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);
}


/*********************    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 digitos 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
                {
                  printf("Hay mas de un punto en el n£mero real!! ");
                  return(0);
                }
              else
              {
                printf("El n£mero tiene caracteres no n£mericos");
                return(0);
              }
  }
  return(1);
}


/*********************    Funci¢n LEE_RASGO_ARCH()    *********************/
```

```
/* Esta funci¢n se encarga de llenar un nodo de la lista de variables    */
/* con la informaci¢n 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(int i)
{

  PON_EN_MATRIZ(RASGOS_INFO, i, 0, token, num_carac_info-1);
  if(OBTEN_TOKEN() == 2)
  {
    PON_EN_MATRIZ(RASGOS_INFO, i, 1,token, num_carac_info-1);
    if(token == "k")
          if(OBTEN_TOKEN() == 1)
            PON_EN_MATRIZ(RASGOS_INFO, i, 2, token, num_carac_info-1);
          else
          {
            printf("El valor de k no es un valor n£merico en el rasgo:%d",i+1);
            return(0);
          }
    else
          PON_EN_MATRIZ(RASGOS_INFO, i, 2,"1", num_carac_info-1);
    if( OBTEN_TOKEN() == 1)
          switch(token[0])
          {
            case '1':PON_EN_MATRIZ(RASGOS_INFO, i, 3, token, num_carac_info-1);
                     if( (OBTEN_TOKEN() == 1) &&  (strcmp((VALOR_MATRIZ(RASGOS_INFO, i, 1,num_carac_info-1)), "b")
==0 )
                        && (REAL(token)) )
                     PON_EN_MATRIZ(RASGOS_INFO, i, 4,token,num_carac_info-1);
                     else
                     {
                        printf("Hay un error en el epsilon del criterio de comparaci¢n del rasgo %d",i+1);
                        return(0);
                     }
                     break;

            case '2':PON_EN_MATRIZ(RASGOS_INFO, i, 3, token, num_carac_info-1);
                     PON_EN_MATRIZ(RASGOS_INFO, i, 4, "0.0", num_carac_info-1);
                     break;

            default: break;
          }
    else
    {
          printf("Hay un error en el epsilon del criterio de comparaci¢n del rasgo %d",i+1);
          return(0);
    }
    if( (OBTEN_TOKEN() == 1) && (REAL(token)) )
          PON_EN_MATRIZ(RASGOS_INFO, i, 5, token, num_carac_info-1);
    else
    {
          printf("Hay un error en el peso informacional del rasgo: %d",i+1);
          return(0);
    }
    return(1);
  }
  else
  {
    printf("Hay un error en el tipo del rasgo: %d",i+1);
    return(0);
  }
}

/*********************    Funci¢n LEE_ARCH_DESC()    *********************
```

```
/* Esta funci¢n se encarga de llenar la lista completa de variables,    */
/* verificando antes si el archivo de descriptores existe, ya que de este*/
/* obtendra la informaci¢n 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    */
/**********************************************************************/

int LEE_ARCH_DESC(void)
{

int i;
char mira;
char *umb;

  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)
  {
    printf("No se puede abrir el descriptor de archivos");
    exit(1);
  }
  while(!feof(entrada) && OBTEN_TOKEN() == 2)
  {
    if(strcmp(token, nom_arch) == 0)
    {
            if(OBTEN_TOKEN() == 1)
              num_clase = atoi(token);
            else
            {
              printf("Hay un error en el num de clases del archivo en el descriptor");
              return(0);
            }
            if( OBTEN_TOKEN() == 1)
              tot_conj_apoyo = atoi(token);
            else
            {
              printf("Hay un error en el n£mero de conjuntos de apoyo");
              return(0);
            }
            if(OBTEN_TOKEN() == 1)
              tot_obj = atoi(token);
            else
            {
              printf("Hay un error en el num de objetos del archivo en el descriptor");
              return(0);
            }
            if(OBTEN_TOKEN() == 1)
              rasgonum = atoi(token);
            else
            {
              printf("Hay un error en el num de rasgos del archivo  en el descriptor");
              return(0);
            }
            if((RASGOS_INFO=(CADENA *) malloc(rasgonum*num_carac_info*(sizeof(CADENA)))) != NULL)
            {
              for(i = 0; i< rasgonum; i++)
                if(OBTEN_TOKEN() == 2)
                {
                        if(!LEE_RASGO_ARCH(i))
                          return(0);
                }
                else
                {
                        printf("Existe un error en el nombre del rasgo:%d",i+1);
                        return(0);
                }
```

```
                return(1);
              }
    }
    else
              do
              {
                mira =getc(entrada);
                if(feof(entrada)) break;
                ungetc(mira, entrada);
               }while(OBTEN_TOKEN() != 5);
  }
  return(0);
}

/*********************    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':if(*variable == '1' || *variable == '0' || *variable == '*')
                        return(1);
                else
                {
                        printf("El rasgo es de tipo booleano as¡ que s¢lo puede tomar valor 0 ¢ 1 ");
                        return(0);
                }

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

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

/*********************    Funci¢n LEE_MA_ARCH()    *********************/
/* Esta funci¢n 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 *matri)
{
 int i, j;
```

```
 char mira;

   for(i = 0; i < tot_obj; i++)
   {
            if( (mira = getc(entrada)) != '\n')
              ungetc(mira,entrada);
            for(j = 0; j < rasgonum; j++)
            {
              OBTEN_TOKEN();
              if(PRUEBA(VALOR_MATRIZ(RASGOS_INFO,j,1,num_carac_info-1),token,
                        VALOR_MATRIZ(RASGOS_INFO,j,2,num_carac_info-1)))
                PON_EN_MATRIZ(matri, i, j, token, rasgonum-1);
              else
                break;
            }
   }
}


/*********************    Funci¢n LECT()    *********************/
/* Esta funci¢n se encarga de verificar si el archivo de datos  */
/* est  vac¡o.                                                  */
/*            Entrada: Ninguna.                                 */
/*     Salida: 1 si el archivo de datos no esta vac¡o, 0 en    */
/*            caso contrario                                    */
/*            Funciones auxiliares: Ninguna.                    */
/*****************************************************************/

int LECT(char *archivo)
{
char micar;

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


/*********************    DATOS_MA()    *********************/
/* Esta funci¢n se encarga de verificar si el archivo de datos  */
/* esta lleno y llenar la matriz de datos con la informaci¢n    */
/* del archivo, utilizando algunas funciones auxiliares.        */
/*                                                              */
/*            Entrada:Ninguna                                   */
/*     Salida: Ninguna                                          */
/*            Funciones auxiliares: PANTALLA, TITULO,           */
/*                    LECT,LEE_MA_ARCH,ERROR                    */
/*****************************************************************/

void DATOS_MA(void)
{

  if(LECT(nom_arch))
  {
    if( (MA=(CADENA *) malloc (tot_obj*rasgonum*(sizeof(CADENA)))) !=NULL)
```

```
    {
            LEE_MA_ARCH(MA);
            fclose(entrada);
    }
    else
    {
            printf("No hay memoria suficiente");
            exit(1);
    }
  }
  else
    exit(1);
}
/****************** LISTA_DESC() ****************/

void LISTA_DESC(CADENA *matriz, int ren, int col)
{
 int i,j;

  for(i=0;i<ren;i++)
  {
    for(j=0; j<col ;j++)
            printf("[%d][%d]=%s ",i,j,VALOR_MATRIZ(matriz,i,j,col-1));
    printf("\n");
  }
}

/***************** LISTA_DESC2() *****************/
void LISTA_DESC2(int *matriz, int ren, int col)
{
 int i,j;

  for(i=0;i<ren;i++)
  {
    for(j=0; j<col ;j++)
            printf("[%d][%d]=%d ",i,j, matriz[(i*col)+j]);
    printf("\n");
  }
}

/******************** LEE_OBJ_ARCH ***************/

int LEE_OBJ_ARCH(NODO_OBJ *auxobj, int i)
{

  int j;
  char mira;
  NODO_CLA *clases, *auxclases, *otraclase,*aux;

  if( (mira=getc(entrada)) != '\n')
    ungetc(mira,entrada);
  auxobj->num_obj=i+1;
  clases=(NODO_CLA *) malloc (sizeof(NODO_CLA));
  auxclases=clases;
  for(j=0; j< num_clase; j++)
  {
    if(OBTEN_TOKEN() == 1)
    {
            if(atof(token) != 0.0)
            {
              auxclases->clase=j+1;
              auxclases->pertenencia=atof(token);
              aux=auxclases;
              if(j != (num_clase-1))
              {
                otraclase=(NODO_CLA *) malloc(sizeof(NODO_CLA) );
                auxclases->sig=otraclase;
                auxclases=otraclase;
              }
            }
    }
```

```
    else
    {
            printf("Hay un error en el archivo de pertenencias de los objetos");
            return(0);
    }
  }
  if(atof(token) == 0.0)
  {
    free(auxclases);
    aux->sig=NULL;
  }
  else
    auxclases->sig=NULL;
  auxobj->apunta_clase=clases;
  return(1);
}

/***************  LEE_ARCH_CLA  *************/

NODO_OBJ *LEE_ARCH_CLA(void)
{
  int i;
  NODO_OBJ *objeto, *auxobjeto, *otrobjeto;

  objeto=(NODO_OBJ*) malloc (sizeof(NODO_OBJ));
  auxobjeto=objeto;

  for(i=0; i<(tot_obj-1); i++)
  {
    if(!(LEE_OBJ_ARCH(auxobjeto,i))) return(NULL);
    otrobjeto=(NODO_OBJ*) malloc (sizeof(NODO_OBJ));
    auxobjeto->sig=otrobjeto;
    auxobjeto=otrobjeto;
  }
  if(!LEE_OBJ_ARCH(auxobjeto,i)) return(NULL);
  auxobjeto->sig=objeto;
  return(objeto);
}

/***************  LEE_CARD_CLA()  ***************/
void LEE_CARD_CLA(void)
{
 int i;
 char mira;

  if( (mira = getc(entrada)) != '\n')
    ungetc(mira,entrada);
  for(i=0;i<num_clase;i++)
  {
    if(OBTEN_TOKEN()==1)
            CARDINAL_CLASE[i]=atoi(token);
    else
    {
            printf("Hay un error en la cardinalidad de las clases");
            exit(1);
    }
  }
}
/********************* DATOS_PERTENENCIA() **************/
void DATOS_PERTENENCIA(void)
{

  gotoxy(6,18);
  printf("Nombre del archivo:");
  scanf("%s",arch_pert_cla);
  if(LECT(arch_pert_cla))
  {
    if( (pri_nom_obj=LEE_ARCH_CLA()) != NULL)
            if( (CARDINAL_CLASE=(int *) malloc (num_clase *(sizeof (int)))) !=NULL )
              LEE_CARD_CLA();
            else
```

```
            {
              printf("No hay memoria suficiente");
              exit(1);
            }
    else
    {
            printf("No se puede leer el archivo de pertenencias");
            exit(1);
    }
  }
  else exit(1);
}

/*****************  LEE_CONJ  ***************/
int LEE_CONJUNTOS(NODO_CONJ *auxconj, int i)
{
 int j;
 char mira;

  auxconj->numero=i+1;
  if( (mira = getc(entrada)) != '\n')
    ungetc(mira,entrada);
  if(OBTEN_TOKEN () == 1)
    auxconj->cardinalidad=atoi(token);
  else
  {
    printf("Hay un error en la cardinalidad del conjunto");
    return(0);
  }
  if(OBTEN_TOKEN () == 1)
    auxconj->peso_info=atof(token);
  else
  {
    printf("Hay un error en el peso informacional del conjunto");
    return(0);
  }
  if(OBTEN_TOKEN () == 1)
    auxconj->func_sem=atoi(token);
  else
  {
    printf("Hay un error en el codigo de la funcion de semejanza");
    return(0);
  }
  if(OBTEN_TOKEN () == 1)
    auxconj->epsilon=atof(token);
  else
  {
    printf("Hay un error en el epsilon de la funcion de semejanza");
    return(0);
  }
  auxconj->indices=(int *) malloc (auxconj->cardinalidad * (sizeof(int)));
  for(j=0;j<auxconj->cardinalidad; j++)
  {
    if(OBTEN_TOKEN() == 1)
            auxconj->indices[j]=atoi(token);
    else
    {
            printf("Los indices de los rasgos del conj de apoyo no son numericos");
            return(0);
    }
  }
  return(1);
}

/******************  LEE_ARCH_CAPOYO  *********/
NODO_CONJ *LEE_ARCH_CAPOYO(void)
{
 int i;

  NODO_CONJ *conjunto, *auxconjunto, *otroconjunto;
```

```
  conjunto=(NODO_CONJ *) malloc (sizeof(NODO_CONJ));
  auxconjunto=conjunto;
  for(i=0; i<(tot_conj_apoyo - 1);i++)
  {
    if(!LEE_CONJUNTOS(auxconjunto,i)) return(NULL);
    otroconjunto=(NODO_CONJ *) malloc (sizeof(NODO_CONJ));
    auxconjunto=otroconjunto;
  }
  if(!LEE_CONJUNTOS (auxconjunto, i)) return(NULL);
  auxconjunto->sig=conjunto;
  return(conjunto);
}

/*****************  DATOS_CONJ_APOYO  **************/
void DATOS_CONJ_APOYO(void)
{

  gotoxy(6,20);
  printf("Nombre del archivo de conj de apoyo:");
  scanf("%s",arch_conj_apoyo);

  if(LECT(arch_conj_apoyo))
  {
    if((pri_nom_conj=LEE_ARCH_CAPOYO()) == NULL)
    {
            printf("Hay un error en el archivo de conjs de apoyo");
            exit(1);
    }
  }
  else
  {
    printf("No se puede leer el archivo de conjs de apoyo");
    exit(1);
  }
}


/*********************  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                      */
/****************************************************************/
 LECTURA_DATOS(void)
{
  clrscr();
  if(LEE_ARCH_DESC())
  {
    fclose(entrada);
    LISTA_DESC(RASGOS_INFO,rasgonum,num_carac_info);
    DATOS_MA();
    fclose(entrada);
    LISTA_DESC(MA,tot_obj,rasgonum);
    DATOS_PERTENENCIA();
    fclose(entrada);
    LISTA_DESC2(CARDINAL_CLASE,1,num_clase);
    DATOS_CONJ_APOYO();
    free(RASGOS_INFO);
    free(MA);
    free(CARDINAL_CLASE);
  }
  else
  {
    printf("El nombre del archivo no existe en el descriptor");
    flushall();
    exit(0);
  }
```

```
}
```

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 ren- */
/*           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*col)+j]);
    fprintf(archivo,"\n");
  }
  fclose(archivo);
}


/*********************    Función BUSCA_NOM_ARCH()    *********************/
/* Esta función se encarga de cambiar la extensión de un archivo        */
/*                                                                      */
/*            Entrada:EL nombre del archivo y la nueva extensión 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);
  cadena2[indice+4]='\0';
  free(cadena2);
  return (cadena2);
}

void LEE_ARCH_MA(char *archi,CADENA *matriz, NODO_RASGO *list_rasg,int num_obj)
{
int ren, col;
char mira;

    flushall();
    rewind(entrada);
    if( (entrada=fopen(archi,"r+")) == NULL)
```

```
    {
            ERROR("No se puede abrir el archivo de resultados"," ",0);
            exit(1);
    }
    for(ren = 0; ren < num_obj; ren++)
    {
            if( (mira = getc(entrada)) != '\n')
              ungetc(mira,entrada);
            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(matriz, ren, col, token, rasgonum);
                else
                        break;
            }
    }
    fclose(entrada);
}

void DESPLIEGA_RESULT(char *archi,int renglones,int columnas,char *titulo)
{
int i,j,x;
char mira;

  rewind(entrada);
  if( (entrada=fopen(archi,"r+")) == NULL)
  {
    ERROR("No se puede abrir el archivo de resultados"," ",0);
    exit(1);
  }
  gotoxy(MinX+10,MinY);printf(titulo);
  gotoxy(MinX+10,MinY+1);printf("==================");
  while(! feof(entrada))
  {
    if( (mira=getc(entrada)) !='\n') ungetc(mira,entrada);
    x=0;
    for(i=0;i<renglones;i++)
    {
            flushall();
            for(j=0;j<columnas;j++)
            {
              OBTEN_TOKEN();
              gotoxy(MinX+10+(10*j), MinY+2+x);printf("%s",token);
            }
            if(x>14)
            {
              ERROR("Para continuar              "," ",0);
              PANTALLA();
              TITULO("      Analisis de Datos");
              gotoxy(MinX+10,MinY);printf("Objeto\tTipo");
              gotoxy(MinX+10,MinY+2);printf("====================");
              x=0;
            }
            x=x+1;
            if( (mira=getc(entrada)) !='\n') ungetc(mira,entrada);
    }
    mira=getc(entrada);
    if(feof(entrada)) break;
  }
  fclose(entrada);
  ERROR(" "," ",0);
}

void CLASIFICA(int num_obj, double *matriz)
{
int i,j;
double tmax;
char nom_arch2[12];
```

```
char ext1[4]=".CLA";
FILE *archivo;


  strcpy(nom_arch2, *BUSCA_NOM_ARCH(nom_archivo3,ext1));
  if((archivo = fopen(nom_arch2, "w")) == NULL)
  {
    ERROR("No se puede abrir el archivo: ",nom_archivo1,0);
    exit(1);
  }
  for(i=0;i<num_obj; i++)
  {
    fprintf(archivo,"O%d",i+1);
    tmax=matriz[i*(num_clase+2)+num_clase];
    for(j=0; j<num_clase; j++)
              if(matriz[i*(num_clase+2)+j] == tmax)
                fprintf(archivo,"\t%d",1);
              else
                fprintf(archivo,"\t%d",0);
    fprintf(archivo,"\n");
  }
  fclose(archivo);
  PANTALLA();
  TITULO("   Resultados de la clasificacion");
  DESPLIEGA_RESULT(nom_arch2,num_obj,num_clase+1,"Objeto\tClases");
}

/*********************    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 solucion 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                           */
/**********************************************************************/

double CALC_PARAM( double *matriz, int num_obj, int columna)
{
double Xmax, Xmin, Xo;
int i;

  Xmax=matriz[num_clase+columna];
```

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

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


/*********************    Función ANALI_OBJ()    *********************/
/* Esta función 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 nom_arch2[12];
char ext1[4]=".ADD";
int i,clase,x;
double Ko,To,Tipmax,c_obj,contraste;
double valor2;
CADENA sol;
FILE *archivo;

  To=CALC_PARAM(matriz2,tot_obj,0);
  Ko=CALC_PARAM(matriz2,tot_obj,1);
  strcpy(nom_arch2, *BUSCA_NOM_ARCH(nom_arch,ext1));
  if((archivo = fopen(nom_arch2, "w")) == NULL)
  {
    ERROR("No se puede abrir el archivo: ",nom_archivo1,0);
    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=matriz2[i*(num_clase+2)+clase-1];
    contraste=matriz2[i*(num_clase+2)+num_clase+1];
    if(Tipmax == valor2)
            x=1;
    else
            x=0;
    strcpy(sol,*REG_SOL(To,Ko,x,Tipmax,contraste));
    fprintf(archivo,"O%d\t%s\n",i+1,sol);
  }
  fclose(archivo);
  PANTALLA();
  TITULO("      Analisis de Datos");
  DESPLIEGA_RESULT(nom_arch2,tot_obj,2,"Objeto\tTipo");
}


/*********************    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)
{

char *val1, *val2;
double valor1,valor2,z;
/*char *val1, *val2;*/

  val1=(char *) malloc (10*(sizeof(char)));
  val2=(char *) malloc (10*(sizeof(char)));
  flushall();
  if(epsilon == 0.0)
  {
    strcpy(val2,*VALOR_MATRIZ(matriz2,ob2,rasg,rasgonum));
    strcpy(val1,*VALOR_MATRIZ(matriz1,ob1,rasg,rasgonum));
    if( (strcmp(val1,val2)) == 0)
    {  free(val1);
            free(val2);
            return(1);
    }
    else
    {
            free(val1);
            free(val2);
            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=1; 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(epsilon_fun == 0)
    if((rasgonum - cuenta_rasgo) == 0)
            return(1);
    else
```

```
            return(0);
  else
    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*l];
                else
                        T_i_1=T_i_1+mat_PI_PD[2*l];
              else
                if(clase == (j+1) )
                        T_j_0=T_j_0+mat_PI_PD[(2*l)+1];
                else
                        T_i_0=T_i_0+mat_PI_PD[(2*l)+1];
            }
            if (T_j_0+T_i_1 == 0.0)
              T_i_1 = 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;
  }
}


/********************    Funci¢n CONTRASTE()    ********************/
/* Esta funci¢n 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;
}
```

Bibliografía y Referencias

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

2.- Vega L.; "Desarrollo de un algoritmo para el análisis de datos y clasificación tipoTipicidad y Contraste"; Tesis de Licenciatura 1995. UNAM

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

4.- Ruiz, J; Lazo, M; "Reconocimiento de Patrones Y"; Diplomado de Computación, Colegio de Computación; Facultad de Ciencias Físico-Matemáticas, Benemérita Universidad de Puebla, 1994.

5.- Dubois D.; Prade H.; "Fuzzy sets and systems theory and applications"; Academic Press Inc., 1980.

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

7.- Chapa, S. ; " Herramientas para Consultas y Capturas basadas en el descriptor de archivos" Informe Técnico; Serie Amarilla; Nº 15. CINVESTAV, I.P.N; 1985.

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

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

10.- Kernighan, B; Ritchie, D.; "El lenguaje de Programación C"; Printece-Hall Hispanoamericana, S.A., 1985.

11.- 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 Central de Las Villas, Cuba. 1994

12.- Shalkoff, R.; "Pattern Recognition: Statistical, Structural and Neural Approaches"; Wiley & Sons, Inc. 1977

13.- Spiegel, M; "Estadística"; McGraw-Hill. 1970.

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

Los abajo firmantes, intregrantes de jurado para el examen de grado que sustentará la

Lic. Leticia Vega Alvarado, declaramos que hemos revisado la tesis titulada:

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

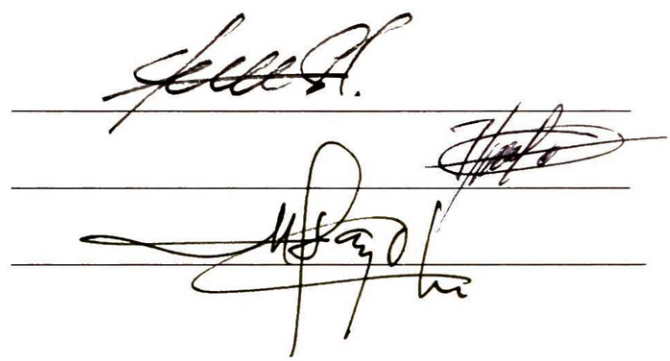
y consideramos que cumple con los requisitos para obtener el grado de Maestra en Ciencias, con especialidad en Ingeniería Eléctrica.

Atentamente

Dr. José Ruíz Shulcloper

Dr. Humberto Sossa Azuela

Dr. Manuel Lazo Cortés



AUTOR	VEGA ALVARADO, L.
TITULO	DESARROLLO DE UN ALGORITMO PARA EL ANALISIS DE DATOS Y CLASIFI..
CLASIF.	XM 96.9
RGTRO.	BI-14772

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
~~Hist~~ Sofía	6/3/97	7/3/97
Fotocopiado - Rep. Téc.	22abr97	23 abr 97