UNIVERSIDAD AUTÓNOMA
DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE INFORMACIÓN Y SISTEMAS

UN MÉTODO PARA LA GENERACIÓN DE ÁRBOLES DE DECISIÓN BASADO EN ALGORITMOS CONCEPTUALES

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRA EN CIENCIAS COMPUTACIONALES

P R E S E N T A

ANILU FRANCO ARCEGA

ASESOR: DR. GUILLERMO SÁNCHEZ DÍAZ

PACHUCA DE SOTO, HIDALGO. JULIO DE 2006

Agradecimientos

A DIOS, por darme la fuerza para conseguir cada una de las metas propuestas en mi vida. Te quiero y siempre estarás conmigo.

A mi mami y papi por todo el amor y las enseñanzas inculcadas en mi, cada palabra de apoyo de su parte ha sido un aliciente para convertirme en la persona que hoy soy. Son grandes, los quiero mucho.

Bere, sin ti no hubiera logrado muchas de las cosas que he hecho, eres y siempre serás una parte fundamental en mi. Te quiero con todo mi corazón.

A toda mi familia, por el apoyo que cada uno a su manera me brindo.

Iván, tu compañía y tu apoyo representan una parte importante en mi vida. Eres una gran persona, te amo.

Dr. Memo, ha sido un gran apoyo para mi y este es un logro que comparto con usted. Gracias por todos sus consejos.

A mis amigos, por su amistad y por los momentos que hemos compartido, también me han ayudado mucho.

A los Doctores y Profesores que me han apoyado a concluir esta etapa de mi vida. Todos sus conocimientos han sido de gran ayuda.

Y por último, pero no por eso menos importante a la Universidad Autónoma de Estado de Hidalgo, por ser la institución que me preparó como profesionista.

Simplemente GRACIAS a todos ustedes!!

Resumen

En este trabajo de tesis se presenta un método para la generación de árboles de decisión, a diferencia de otros algoritmos que hacen uso de la matriz de aprendizaje, el método propuesto se basa en la utilización de conceptos por clase para la construcción.

Dentro de las tareas realizadas destaca el diseño de un criterio de división para la selección del subconjunto de atributos que definen a un nodo del árbol, dicho criterio no toma la entropía como medida de información, así como la implementan otros algoritmos.

Para mejorar el proceso de construcción del árbol, se realizó además una simplificación de los conceptos que caracterizan a las clases, definiendo de esta manera nuevos conjuntos de conceptos de cobertura mínima.

Entre las ventajas que presenta el método propuesto está también la capacidad de trabajar con conjuntos de datos cuantitativos y cualitativos. Además de poder procesar conjuntos de datos extensos, es decir, descritos por un gran número de objetos.

Se presentan de igual forma en este trabajo los resultados obtenidos al implementar este método en tareas de clasificación supervisada, mostrando un desempeño aceptable.

Índice general

Índice de figuras

Índice de Tablas

Introducción

Dentro del presente trabajo se analizaron los presupuestos, alcances y limitaciones de algunos algoritmos de generación de árboles desarrollados en los últimos años, esto con el objeto de definir un nuevo método de generación de árboles de decisión en el marco de la Clasificación Supervisada (CS) que supere algunas de las limitaciones actuales de estos algoritmos.

Objetivo General

Diseñar y programar un método para la generación de árboles de decisión que basado en conceptos obtenidos a partir de un algoritmo conceptual.

Actualmente existen métodos que permiten la generación de árboles de decisión para problemas de clasificación supervisada, algunos de estos métodos trabajan con conjuntos de datos descritos tanto por datos cualitativos como por cuantitativos pero procesan los objetos sólo como datos de tipo cuantitativo, como el caso de los algoritmos C4.5, CART (Classification and Regresion Trees) [JRQ99], [LB84]; además de los métodos FACT (Fast Algorithm for Classification Trees) y QUEST (Quick, Unbiased, Efficient, Statistical Tree) [WYL88], [WYL97].

Para tratar con datos cualitativos se cuenta con el método ID3, el algoritmo k-d (k-dimensional) y un algoritmo FDT (Fuzzy Decision Trees) que genera un árbol de decisión difuso basado en ID3, sin embargo cuando se desea procesar datos mezclados las opciones se limitan a la utilización de un algoritmo que trabaja con tipos de datos híbridos pero que desafortunadamente dentro de su funcionamiento se tiene una separación en la forma de manejar los datos cuantitativos y los cualitativos [PHKYJ05], ó al algoritmo LMDT (Linear Machine Decision Trees), que realiza una codificación de los datos para poder manejarlos.

Como una manera de superar la limitación de trabajar en forma separada o con una transformación a los datos cualitativos y cuantitativos se ha propuesto un método de generación de árboles, que permite el manejo de conjuntos de datos mezclados.

La importancia de manejar en forma conjunta los datos cuantitativos y cualitativos

se basa en la idea de que las descripciones de los objetos están determinadas por la totalidad de las variables que lo constituyen, y no sólo por un subconjunto de éstas.

El método ADT (Alternative Decision Trees), propuesto en este trabajo de tesis, se basa en el empleo de conceptos caracterizantes y excluyentes de clases, obtenidos mediante el Algoritmo Conceptual RGC (Refunion-Generalization-Conceptual), para la generación de árboles de decisión. Estos conceptos además fueron reducidos a conceptos de cobertura mínima como parte de un proceso para mejorar la construcción del árbol.

Una parte importante dentro del diseño de algoritmos para la generación de árboles de decisión, es la determinación de la regla o criterio de división que se utiliza para seleccionar el atributo con el cual se ha de crear un nuevo nodo en el árbol; por lo que dentro de este trabajo también se ha propuesto una regla que realice este proceso.

Por tanto, el método ADT se describe de manera general de la siguiente forma:

Como entrada recibe el conjunto de conceptos excluyentes que caracterizan a cada una de las clases presentes en el conjunto de datos. Posteriormente, se simplifican a la forma de conceptos de cobertura minima (CM). Este conjunto de conceptos es la entrada a una función recursiva que genera el árbol de decisión a partir de la regla de división propuesta.

Aportes del trabajo

Los *Aportes* que se presentan en este trabajo son:

- Los Árboles de Decisión generados por el método propuesto basan su construcción en conceptos, no en la matriz de aprendizaje como lo hacen los algoritmos desarrollados por otros investigadores y que son analizados en este trabajo. Esta característica contribuye a construir árboles de decisión más robustos que admiten y cubren cambios en la MA sin necesidad de volver a construir el árbol generado.
- Se propuso un criterio de separabilidad entre clases diferente al criterio clásico de entropía, en el cual se basan otros algoritmos para la selección del atributo en la construcción del árbol. Este criterio además de encontrar el valor de separabilidad existente entre todas las clases que intervengan en el nodo a construir, permite seleccionar un conjunto de variables (no solamente una como en el caso de la entropía) para formar la condición que define al nodo.
- Se propuso una forma de simplificar los conceptos obtenidos por el algoritmo RGC para utilizarlos de esta manera en la construcción del árbol. El concepto creado por RGC se compone de un conjunto de descripciones o características.

El concepto simplificado sólo contiene una sola descripción de conjunciones de variables que representan al concepto original.

- Se permite el manejo de datos mezclados por la forma en cómo el algoritmo RGC trabaja los datos.

Limitaciones

Las limitaciones encontradas en el método propuesto, y la forma en cómo se propone resolverlas, son:

- Los conceptos generados por RGC presentan en su descripción una característica que no se debe cumplir (existencia de l-complejos elementales asociados) para que el concepto no deje de ser excluyente a la clase que caracteriza. Esta característica resulta ser un inconveniente al momento de generar el árbol de decisión, es por eso que se propondrá desarrollar un nuevo operador de generalización para los conceptos obtenidos por RGC.
- El manejo de ausencia de información esta contemplado por RGC, sin embargo el algoritmo propuesto no ha considerado para la construcción del árbol esta característica, que puede estar presente en los datos. Como trabajo futuro se contempla desarrollar una manera de manejar esta ausencia de información.

Organización de la tesis

La organización de la tesis presentada es la siguiente:

El primer capítulo *"Conceptos Básicos"* presenta aquellos conceptos que es importante conocer, a fin de comprender el problema de Clasificación Supervisada en el Reconocimiento de Patrones y la estructuración de un árbol de decisión. En el segundo capítulo *"Algoritmos clásicos para la generación de árboles de decisión"* se muestra el análisis crítico de los algoritmos de generación de árboles de decisión más representativos, haciendo énfasis en sus principales características, presupuestos, ventajas y limitaciones.

Así mismo, en el tercer capítulo, titulado *"El Método Propuesto para la Generación de Árboles de Decisión"*, se propone una nueva alternativa para la generación de árboles de decisión que supera algunas de las limitaciones de los algoritmos ya estudiados en el segundo capítulo.

El cuarto capítulo *"Experimentación efectuada"* contiene los resultados obtenidos a partir de un conjunto de datos. Se muestra el árbol de decisión generado y el porcentaje de clasificación que se obtuvo con esos resultados. Además, se muestra la comparación

de ADT con otros algoritmos que permiten la generación de árboles de decisión. Las *Conclusiones y Trabajo Futuro* resumen los resultados más importantes obtenidos con este trabajo y presentan el trabajo futuro que se pretende iniciar a partir de lo ya desarrollado.

Finalmente, los *Apéndices A, B, C, D* presentan algunos de los conceptos básicos y los algoritmos de algunos de los métodos estudiados en el capítulo 2. El Apéndice E muestra la descripción de la base de datos Zoo utilizada en la experimentación de este trabajo.

Capítulo 1

Conceptos Básicos

El objetivo principal de este capítulo es mostrar los conceptos y definiciones que pueden ser de utilidad para comprender la clasificación de objetos y la generación de árboles de decisión, tópicos fundamentales en la realización de este trabajo.

1.1. Conceptos básicos del Reconocimiento de Patrones

El Reconocimiento de Patrones (RP) es una disciplina científica que se empieza a conformar a finales de los años 50 [Ros61]. Busca la sustitución de algunas actividades humanas con dispositivos electrónicos, como la computadora.

Es un conjunto de investigaciones científicas encaminadas a resolver problemas relacionados con la clasificación de objetos y fenómenos de naturaleza muy diversa; como el reconocimiento, la identificación, el diagnóstico, el pronóstico, etc.

También se utiliza para la clasificación de las formas o patrones que aparecen en una imagen de computadora. Tomando como resultado del análisis de patrones, la decisión sobre la pertenencia o no del patrón bajo análisis a las clases o estructuras conocidas. En muchas ocasiones se identifica esta disciplina con todo lo relacionado al procesamiento y análisis de imágenes exclusivamente. Sin embargo, abarca una cantidad mayor de problemas.

Existen diferentes enfoques para resolver los problemas de Reconocimiento de Patrones, tales como el Estadístico (ver por ejemplo [Fuk90]), el Sintáctico Estructural [Fu74] y el Lógico Combinatorio [Laz95b], [Mar99].

En esta tesis se trabaja con el Enfoque Lógico Combinatorio (ELC) debido a que permite el manejo de datos mezclados (cualitativos y cuantitativos) sin realizar cambios de espacio o transformaciones en la MA. Además, en el ELC está concebido el algoritmo conceptual RGC utilizado. El cual presenta diversas bondades que otros algoritmos no

poseen.

1.1.1. Problemas de Reconocimiento de Patrones

En RP existen esencialmente cuatro familias de problemas, las cuales son:

1. **Clasificación con aprendizaje (o supervisada)** donde se conoce un universo de objetos que se agrupa en un número de clases dado, y se tiene una muestra de objetos de cada clase. El problema consiste en dado un nuevo objeto poder establecer las relaciones con cada una de dichas clases.

   Como ejemplo de este tipo de problemas, se tiene el diagnóstico diferencial de enfermedades; teniendo por caso, el de bronquiolitis y asma en niños menores de dos años. En este caso, se tiene una población bien definida: los niños menores de dos años; además, se cuenta con sendos grupos de niños a los cuales se les ha practicado una serie de estudios que arrojó como conclusión el padecimiento de una enfermedad u otra. El problema es, dado un niño con un cuadro de sibilancia, determinar si es una bronquiolitis o un asma, para poder aplicar el tratamiento correcto en el momento indicado.

2. **Clasificación con aprendizaje parcial (parcialmente supervisada)** es análogo al anterior, excepto que hay una clase de objetos de la cual no se tiene muestra. Y el problema original sigue siendo el mismo: dado un nuevo objeto, relacionarlo con los ya clasificados.

   Como ejemplo, se puede citar el conjunto de zonas geológicas en México perspectivas y no perspectivas con respecto a la presencia de petróleo. Se tienen características de las zonas con petróleo, pero no de las que no tienen.

3. **Clasificación sin aprendizaje (no supervisada)** en esta familia se tiene un universo de objetos, pero no se conoce cómo se agrupan, y éste es precisamente el objetivo. El problema radica en encontrar cómo se estructura el universo de objetos mencionado anteriormente. La clasificación sin aprendizaje se puede dividir en restringida y libre. En la primera, se determina a priori el número de clases que se deben generar. En la segunda, no se determina el número de clases a formar.

   Por ejemplo, el determinar cuántas clases y qué objetos se agrupan en cada una de ellas cuando se analiza un conjunto de estudiantes de un curso dado y el objetivo principal es agruparlos con respecto a determinadas habilidades para alguna asignatura.

4. **Selección de rasgos** [1] se utiliza para reducir el número de rasgos con los cuales se deben describir los objetos en modo eficiente y también para encontrar los rasgos que inciden en el problema de manera determinante.

El problema de Clasificación Supervisada (y predicción) es muy importante cuando se quiere determinar la inclusión de nuevos objetos a conjuntos de datos ya existentes. Para poder establecer esa pertenencia a una clase en específico en dicho conjunto de datos, se requiere verificar el grado de semejanza de los objetos descritos y los objetos nuevos.

Existen muchos problemas en la vida real en los que se puede aplicar la clasificación supervisada. Por ejemplo en la clasificación de: pacientes con un cuadro sintomatológico en específico, fenómenos naturales o sociales, etc.

1.2. Matriz de Aprendizaje

Las definiciones y proposiciones incluidas en esta sección fueron obtenidas de varios materiales impresos, puede consultarse por ejemplo [Laz95b], [Laz95a], [Bar94], [Laz93] y [Laz94].

Se asume que se tiene un universo de objetos admisibles M y una partición $K_1, \ldots, K_r$ de M por subconjuntos propios $K_i \subset M$, $i = 1, \ldots, r$, $K_1 \cap K_2 \cap \ldots \cap K_{r-1} \cap K_r = \emptyset$. Sea $R = \{X_1, \ldots, X_n\}$ el conjunto de variables a través de los cuales se describen los objetos de M. Cada variable X_j, $j = 1, \ldots, n$ tiene asociado un conjunto D_j que se denomina *conjunto de valores admisibles de la variable* X_j.

A los subconjuntos $K_i \subset M$ se les denomina clases.

Definición 1. Una descripción estándar de un objeto es un n-uplo $I(O) = (X_1(O), \ldots, X_n(O))$ donde $X_i(O) \in D_i$, $i = \overline{1, n}$ es el valor de la variable X_i en el objeto O.

El tipo de una variable depende de la naturaleza de su conjunto de valores admisibles:

Si el conjunto D_i es un conjunto discreto cuyos valores admisibles son meramente nominales (carecen de orden), entonces se dice que la variable X_i es cualitativa nominal. La distinción de sexo masculino, femenino y el grupo sanguíneo A, B, AB, O de una persona son ejemplos de este tipo de variable.

[1] **También se manejarán para referirse a los rasgos, los nombres de variables y atributos.**

Una variable es cuantitativa si su conjunto de valores admisibles D_i está formado por números y preserva un orden total. El nivel de glucosa en sangre es un ejemplo de este tipo de variable, con el orden de los números reales.

Cuando se habla de variables cualitativas y cuantitativas se asume que son univaluadas, es decir, en su conjunto de valores admisibles sólo existen conjuntos unitarios.

Definición 2. La función $C_i : D_i \times D_i \longrightarrow F$ se denomina criterio de comparación de valores de X_i, donde $F = \{0, 1\}$, si C_i es un criterio de comparación booleano; $F = \{0, 1, \ldots, k-1\}$ cuando C_i es k-valente; $F = [0, 1]$ o $F = \Re$ si el criterio de comparación es aritmético o cuantitativo.

Definición 3. Se denominará r-uplo de pertenencia del objeto O_j al r-uplo $\alpha(O_j) = (\alpha_1(O_j), \ldots, \alpha_r(O_j))$, donde $\alpha_i(O_j) \equiv$ " $O_j \in K_i$ " siendo $\alpha_i(O_j) \in \{0, 1, *\}$, $i = 1, \ldots, r$ y $\alpha_i(O_j)$ simbolizará la pertenencia del objeto O_j al subconjunto K_i de M de la siguiente manera.

$$\alpha_i(O_j) = 1 \quad significa \quad O_j \in K_i$$

$$\alpha_i(O_j) = 0 \quad significa \quad O_j \notin K_i$$

$$\alpha_i(O_j) = * \quad significa \quad O_j \; ? \; K_i$$

Si $\alpha_i(O) \neq *$, $i = 1, \ldots, r$ se dirá que el r-uplo de pertenencia de O_j es completo.

Si $\alpha_i(O_j) = P_i(O_j)$ donde $P_i(O_j)$ es el predicado que describe correctamente la pertenencia del objeto O_j a la clase K_i, $i = 1, \ldots, r$ se considera que el r-uplo de pertenencia de O_j es correcto.

Definición 4. Por información estándar de las clases $K_1, \ldots, K_r$ se entiende

$$I_O(K_1, \ldots, K_r) = (I(O_1), \alpha(O_1), \ldots, I(O_m), \alpha(O_m))$$

donde $I(O_i)$ es la descripción de O_i y $\alpha(O_i)$, $i = 1, \ldots, m$ su r-uplo de pertenencia.

Una información estándar se denomina completa y correcta si y sólo si todos sus r-uplos de pertenencia sean completos y correctos.

En lo sucesivo se asumirá que se trabajará con información estándar correcta de las clases $K_1, \ldots, K_r$.

Suponiendo que se cuenta con:

(i) Un universo M de objetos,

(ii) $K_1, \ldots, K_r$ subconjunto de M, las cuales se denominan clases y conforman un cubrimiento de M,

(iii) $X_1, \ldots, X_n$ rasgos o variables a través de las cuales se describen los objetos de M,

(iv) $D_1, \ldots, D_n$ conjuntos de valores admisibles de las variables,

(v) $I_O(K_1, \ldots, K_r)$ información estándar correcta de las clases $K_1, \ldots, K_r$.

Esta información se puede representar en forma de matriz. A esta representación se le conoce como *matriz de aprendizaje* (MA) mostrada en la Figura 1.1.

Clases	Objetos	Rasgos
K_1	O_1	$X_1(O_1) \ \ldots \ X_n(O_1)$
	$\vdots$	$\vdots \qquad \vdots$
	O_i	$X_1(O_i) \ \ldots \ X_n(O_i)$
K_2	O_{i+1}	$X_1(O_{i+1}) \ \ldots \ X_n(O_{i+1})$
	$\vdots$	$\vdots \qquad \vdots$
	O_j	$X_1(O_j) \ \ldots \ X_n(O_j)$
	$\vdots$	$\vdots \qquad \vdots$
K_r	O_l	$X_1(O_l) \ \ldots \ X_n(O_l)$
	$\vdots$	$\vdots \qquad \vdots$
	O_m	$X_1(O_m) \ \ldots \ X_n(O_m)$

Figura 1.1: Estructura General de la Matriz de Aprendizaje

La información representada en forma de matriz, indica que se tienen m-objetos, cada uno con n-rasgos y divididos en r-clases.

Un ejemplo de una MA con descripciones de objetos determinadas por tipos de datos mezclados, se presenta en la figura 1.2:

	X_1	X_2	X_3	X_4	X_5
O_1	0.7	0	0	0	1
O_2	0.8	1	3	0	1
O_3	0.9	1	1	0	0
O_4	0.1	0	2	1	1
O_5	0.3	1	4	0	1

Figura 1.2: Ejemplo de Matriz de Aprendizaje

Donde: X_1 es variable real, X_3 variable 5-valente, X_2, X_4 y X_5 son variables boolenas.

Suponiendo que los criterios de comparación de las variables son:

$$C_1(X_1(O_i), X_1(O_j)) = \begin{cases} 0 & \text{si } |X_1(O_i) - X_1(O_j)| < 0.5 \\ 1 & \text{en otro caso.} \end{cases}$$

$$C_3(X_3(O_i), X_3(O_j)) = \begin{cases} 0 & \text{si } X_3(O_i) \in A_k \text{ y } X_3(O_j) \in A_k, \\ & k = 1, 2 \\ 1 & \text{en otro caso} \end{cases}$$

Con $A_1 = \{0, 3\}$, y $A_2 = \{1, 2, 4\}$

$$C_k(X_k(O_i), X_k(O_j)) = \begin{cases} 0 & \text{si } X_k(O_i) = X_k(O_j) \\ & \text{k = 2,4,5} \\ 1 & \text{en otro caso} \end{cases}$$

En general, la matriz de aprendizaje así como los criterios de comparación entre los objetos de ésta, son proporcionados por el especialista del problema.

1.3. Conceptos Básicos de Árboles de Decisión

Un árbol de decisión (AD) es una representación gráfica de un proceso que permite la clasificación o evaluación de un objeto en específico [Utg98]. Estos AD's se han utilizado en muchas áreas de forma exitosa, tales como la clasificación de señales de radar,

el reconocimiento de caracteres y de lenguaje, para diagnósticos médicos y en sistemas expertos, por mencionar algunas.

Los AD son comúnmente usados debido a que en ellos se muestra de forma clara cómo llegar a tomar una decisión; ésta es quizás su característica más importante, la capacidad de convertir un proceso de decisión complejo en una colección de simples decisiones que proveen una solución de fácil interpretación [Mit97].

A continuación se presentan algunas definiciones básicas para la comprensión de una estructura de un AD, las cuales fueron obtenidas de diversos materiales, de los cuales pueden consultarse [Lan91], [Qui86], [Mar03], [Qui93].

Definición 5. Un grafo G = (V, E) consiste en un conjunto V de nodos (o vértices) finito y no vacío y un conjunto E de arcos. Si los arcos son pares ordenados de vértices (v, w) entonces se dice que el grafo es dirigido $E = \{(v, w) | v \in V \wedge w \in V\}$.

Definición 6. Un camino en un grafo es una secuencia de arcos de la forma:

$$(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$$

Se dice que el camino de v_1 a v_n es de longitud $n - 1$.

Definición 7. Un grafo dirigido sin ciclos es llamado grafo dirigido acíclico. Un árbol dirigido es un grafo dirigido acíclico que satisface las siguientes propiedades:

1. Sólo hay un nodo, llamado raíz, que no tiene arcos de entrada. Este nodo contiene todas las etiquetas de las clases.
2. Cada nodo excepto la raíz tiene un arco de entrada.
3. Sólo existe un único camino de la raíz a cada nodo.

Definición 8. Si (v, w) es un arco en un árbol, v es considerado padre de w, y w es considerado hijo de v. Si existe un camino de v a w, $v \neq w$, entonces v es antecesor de w y w es descendiente de v.

Definición 9. Un nodo sin descendientes se considera hoja o nodo terminal. Todos los demás nodos (a excepción de la raíz) son llamados nodos internos.

Definición 10. La profundidad de un nodo v en un árbol es la longitud del camino de la raíz a v. La altura del nodo v es la longitud del camino más largo de v a una hoja. La altura de un árbol es la altura de su raíz. El nivel de un nodo v en un árbol es la altura del árbol menos la profundidad de v.

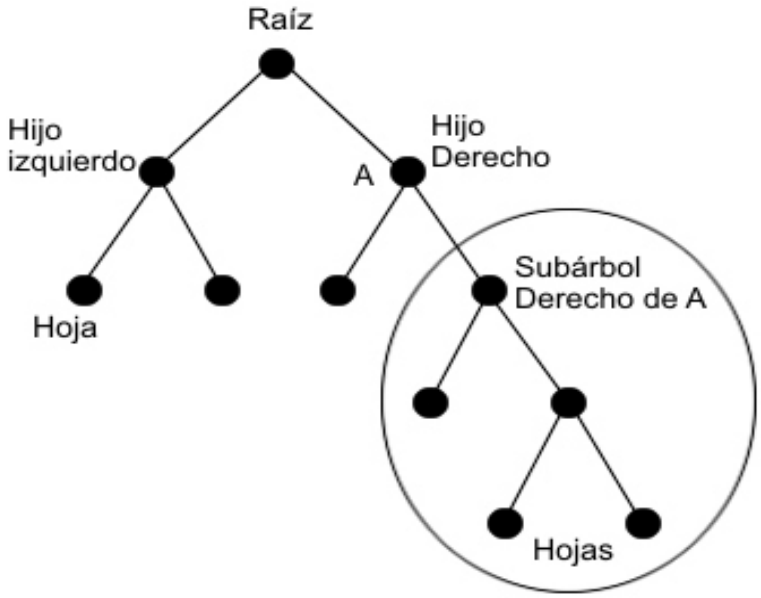

Figura 1.3: Estructura de un Árbol

Definición 11. Un árbol ordenado es un árbol en el cual los hijos de cada nodo son ordenados (normalmente de izquierda a derecha).

Definición 12. Un árbol binario es un árbol ordenado tal que

1. Cada hijo de un nodo es distinguido también como un hijo izquierdo o como un hijo derecho,

2. Ningún nodo tiene más de un hijo izquierdo ni más que un hijo derecho.

Árboles con más de dos hijos pueden ser reducidos a árboles binarios de una manera simple [LD96], sólo se asocia un hijo izquierdo con cada nodo seleccionando el hijo más viejo en la lista de hijos. Posteriormente, el hijo derecho de un nodo es el siguiente hermano. La figura 1.4 muestra un árbol (no binario) ordenado, sus hijos son ordenados del más viejo al más joven. La figura 1.5 es el árbol binario correspondiente al árbol de la figura 1.4.

Definición 13. El balance de un nodo v en un árbol binario es $(1 + L)/(2 + L + R)$, donde L y R son el número de nodos en los subárboles izquierdo y derecho de v. Un árbol binario es $\alpha - balanceado$, $0 < \alpha \leq 1$, si cada nodo tiene balance entre α y $1 - \alpha$. Un árbol binario $1/2$ - balanceado es también conocido como un árbol completo.

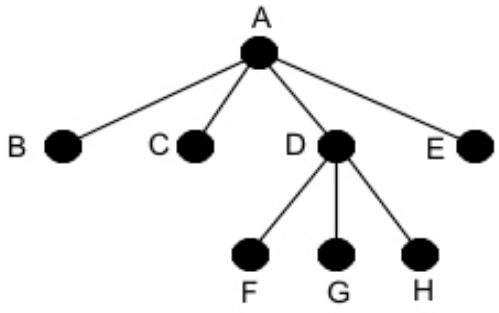

Figura 1.4: Árbol Ordenado

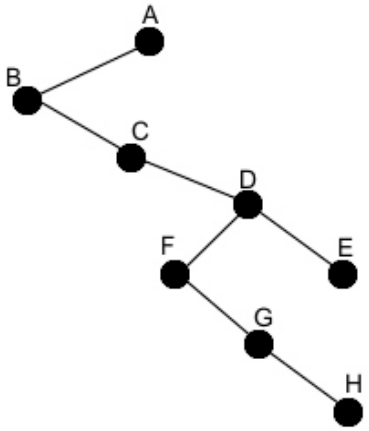

Figura 1.5: Árbol Binario

Definición 14. Dado un conjunto de objetos MI, donde cada objeto es descrito por un conjunto de variables $X_1, X_2, \cdots, X_n$ y dado un conjunto de clases $C = C_1, \cdots, C_m$, un árbol de decisión o árbol de clasificación es un árbol asociado a MI y cumple las siguientes propiedades:

- Cada nodo interno es etiquetado con un atributo, X_i.
- Cada arco es etiquetado con un predicado, aplicado al atributo asociado al padre.
- Cada nodo hoja es etiquetado con una clase, C_j.

Un árbol de decisión es típicamente diseñado y construido en forma recursiva siguiendo una estrategia descendente, es decir, desde conceptos generales hasta ejemplos particulares. La familia de algoritmos de construcción "Top-Down Induction on Decision

Trees (TDIDT)"se forma de métodos que generan este tipo de árboles y que siguen la filosofía "divide y vencerás", la cual se basa en los siguientes puntos [Gal02]:

- Si existen uno o más casos en el conjunto de entrenamiento y todos ellos corresponden a objetos de una misma clase c, el árbol de decisión es una hoja etiquetada con la clase c. Se ha alcanzado un nodo puro.
- Si no se encuentra ninguna forma de seguir ramificando el árbol o se cumple alguna condición de parada no se sigue expandiendo el árbol por la rama actual. Se crea un nodo hoja etiquetado con la clase más común del conjunto de casos de entrenamiento que corresponden al nodo actual. Si el conjunto de casos de entrenamiento queda vacío, la clasificación adecuada ha de determinarse utilizando información adicional.
- Cuando en el conjunto de entrenamiento hay casos de distintas clases, éste se divide en subconjuntos que sean o conduzcan a agrupaciones uniformes de casos, entendiendo por estos conjuntos de casos correspondientes a una misma clase. Utilizando los casos de entrenamiento disponibles, se selecciona una pregunta para ramificar el árbol de decisión. Dicha pregunta, basada en los valores que toman los atributos predictivos en el conjunto de entrenamiento, ha de tener dos o más respuestas alternativas mutuamente excluyentes R_i. De todas las posibles alternativas, se selecciona una empleando una regla heurística a la que se denomina regla de división. El árbol de decisión resultante consiste en un nodo que identifica la pregunta realizada del cual cuelgan tantos hijos como respuestas alternativas existan. El mismo método utilizado para el nodo se utiliza recursivamente para construir los subárboles correspondientes a cada hijo del nodo, teniendo en cuenta que al hijo H_i se le asigna el subconjunto de casos de entrenamiento correspondientes a la alternativa R_i.

Definición 15. Se denomina regla de división a cualquier pregunta que divida el conjunto de casos de entrenamiento en al menos dos subconjuntos no vacíos lo cual conducirá a la construcción de un árbol de decisión.

Cada posible pregunta debe evaluarse mediante alguna medida de información y, dado que los algoritmos de construcción de árboles de decisión suelen ser de tipo greedy (es decir, siempre va a elegir la mejor opción), esta medida desempeña un papel esencial en la construcción del árbol: una vez que se ha escogido una pregunta para expandir un nodo, no se vuelven a considerar otras alternativas.

Las medidas estadísticas empleadas intentan favorecer las divisiones que discriminan mejor unas clases de otras. Ejemplos muy conocidos de estas medidas son la ganancia de información usada por ID3, el criterio de proporción de ganancia de C4.5 o el índice de diversidad de Gini empleado en CART.

Capítulo 2

Algoritmos clásicos para la generación de AD

En este capítulo se presenta un análisis crítico de algunos de los algoritmos previamente desarrollados que generan árboles de decisión. El objetivo de este análisis crítico es mostrar, además del estado del arte, los alcances y las limitaciones que cada algoritmo tiene para su aplicación en la práctica.

A partir de esas limitaciones y alcances de los algoritmos revisados, se propone generar un método que supere algunas de las limitaciones presentadas, utilizando nuevas técnicas para la construcción del árbol de decisión.

Los árboles de decisión constituyen probablemente el modelo de clasificación más popular y utilizado [Qui96]. Un árbol de decisión puede utilizarse para clasificar un ejemplo, comenzando a preguntar en su raíz y siguiendo el camino determinado por la respuesta en él, esto lo llevará a los nodos internos hasta llegar a una hoja del árbol, donde se encontrará una decisión. Su funcionamiento es análogo al de una aguja de ferrocarril: cada caso es dirigido hacia una u otra rama de acuerdo con los valores de sus atributos al igual que los trenes cambian de vía según su destino (las hojas del árbol) en función de la posición de las agujas de la red de ferrocarriles (los nodos internos).

Los árboles de clasificación son útiles siempre que los ejemplos a partir de los que se desea aprender se puedan representar mediante un conjunto prefijado de atributos y valores, ya sean éstos discretos o continuos.

Entre los métodos de generación de árboles de decisión más conocidos se encuentran el ID3, el C4.5, CART, aunque existen muchos métodos más que se han desarrollado.

2.1. Algoritmos para datos cualitativos

Existen diversos conjuntos de datos descritos sólo por variables de tipo nominal (discretas). El algoritmo más conocido para generar árboles de decisión basándose en familias de datos de este tipo es el ID3, desarrollado por Quinlan.

2.1.1. ID3

La estructura básica del ID3 corresponde a un algoritmo iterativo. Un subconjunto del universo de objetos MI, el cual se conoce como MA (matriz de aprendizaje), es seleccionado de manera aleatoria y a partir de él se forma un árbol de decisión; este árbol clasificará de manera correcta todos los objetos en MA.

Posteriormente se clasificarán todos los objetos restantes de MI usando el árbol y si éste proporciona una correcta clasificación para todos ellos entonces este árbol se ha generado de forma correcta para todo objeto de MI y el proceso termina. Si no, se seleccionan los objetos que no hayan sido clasificados correctamente, se añaden a MA y se continúa el proceso.

Este algoritmo se basa en la metodología de construcción top-down y la idea central que sigue es seleccionar el atributo que clasifique mejor los objetos de MA para cada uno de los nodos que componen el árbol mediante una medida estadística llamada ganancia de la información [Mit97] (ver Apéndice A). Su construcción comienza por elegir del conjunto total de atributos ese que mejor clasifique para colocarlo como nodo raíz, el nodo tendrá tantas ramas como valores admisibles tenga el atributo, a partir de ahí se formarán nuevos nodos (internos) tomando en cuenta solo las descripciones de objetos que cumplan con el valor de la rama en cuestión. Se continuará el proceso hasta encontrar solo descripciones que correspondan a una misma clase y en ese momento se construirá un nodo terminal (hoja). El algoritmo ID3 se presenta en el Apéndice A.

La principal limitación de ID3 es que es un algoritmo concebido sólo para tratar con conjuntos de descripciones de datos de tipo cualitativo. Todas las bases de datos que contengan datos de tipo cuantitativo no podrán ser procesadas por este algoritmo. Además, debido a que este algoritmo basa su construcción del AD en la MA, puede resultar complicado cuando se presenten conjuntos de datos extremadamente grandes.

Otra limitación que presenta es cuando llega una nueva descripción de un objeto a la MA. ID3 tiene que volver a generar el AD si es que quiere considerar esa nueva muestra dentro del árbol.

2.1.2. Algoritmo k-d

El algoritmo k-d fue propuesto como un algoritmo de clasificación supervisada para minería de datos, el cual genera un árbol de decisión llamado k-dimensional, donde cada nodo es expandido eligiendo el atributo que produce menos confusión en la MA[AG98].

La forma de construcción del árbol es similar al ID3, la salida eventual es un árbol en el cual cada hoja lleva un nombre de clase y cada nodo interior especifica un atributo con una correspondiente rama a cada posible valor de este atributo. La diferencia principal entre ID3 y este algoritmo es el criterio de selección de atributos, ya que el primero basa, esta selección, en la suposición de que la complejidad del árbol está fuertemente relacionada con la cantidad de información que expresa el atributo, y el segundo sólo basa la selección en la confusión que puede inducir un atributo, es decir, si dos objetos de clases distintas poseen el mismo valor en algún atributo.

El atributo que se elige es el que produce menor confusión en la MA, éste será el que tiene el menor número de pares de objetos que tienen el mismo valor en ese atributo pero que pertenecen a clases distintas. El algoritmo k-d es presentado en el Apéndice B de esta tesis.

Entre las deficiencias que presenta este algoritmo es que sólo trata con atributos que poseen un número pequeño de valores (booleanos, k-valentes, intervalos y enteros pequeños), es decir, trabaja con atributos que contengan valores discretos y cuyos conjuntos de valores admisibles sean de limitada extensión, si se presenta un atributo con un número grande de valores, el experto tendrá que dividirlo en intervalos para transformarlo a k-valente.

Tomando en cuenta que su regla de división revisa por pares de objetos para determinar la confusión que induce un atributo, se encontró que si se presenta una MA de extensión grande, puede resultar complejo el proceso de construcción del AD.

2.1.3. FDT

Existe un algoritmo para generar árboles de decisión difusos que trabaja siguiendo la metodología presentada por ID3 [Jan98]. A partir de ejemplos previamente clasificados (MA) se genera un árbol de decisión que ayude a clasificar nuevos objetos.

Este algoritmo maneja tanto datos cualitativos como cuantitativos, pero las descripciones de los valores para cada atributo las representa mediante funciones de pertenencia, las cuáles poseen un número finito de valores lingüísticos, por lo que de alguna manera se discretizan las variables cuantitativas en este proceso de representación, y de esta manera todas las variables presentes en MA son tomadas como cualitativas.

Cada nodo que se vaya construyendo tendrá tantas ramas como valores lingüísticos tenga la variable elegida para formar ese nodo. Las variables son elegidas tomando el mismo criterio de división que maneja ID3: ganancia de información [Mar98]. La diferencia principal entre este algoritmo e ID3 es que este método usa el criterio de división basándose en restricciones difusas, por lo que un ejemplo de la MA puede ser encontrado en más de un nodo pero con cierto grado de pertenencia.

Las limitaciones que presenta este método son básicamente que basa su construcción en la MA, además de representar las variables cuantitativas como cualitativas con valores lingüísticos, el poder determinar cuales y cuantos serían esos valores puede representar un problema, no solo para el método sino para el experto al momento de determinarlos.

2.2. Algoritmos para datos cuantitativos

En conjuntos de datos reales, no todos los datos con los que nos encontramos son de tipo nominal o categórico, sino que también suele haber datos de tipo numérico. Costes, precios de venta, tamaños o pesos son algunos ejemplos de cantidades que pueden aparecer como atributos numéricos en cualquier base de datos.

Resulta necesario, por tanto, disponer de métodos que permitan trabajar con atributos cuantitativos a la hora de construir un árbol de decisión. Entre los algoritmos más populares que permiten la manipulación de este tipo de datos se encuentran el C4.5 y el CART.

2.2.1. C4.5

C4.5 pertenece a una sucesión de algoritmos para generar árboles de decisión que trazan sus orígenes en los trabajos de Hunt y otros autores a finales de los 50s y principios de los 60s [JRQ99].

El algoritmo C4.5 es una extensión de ID3 [Mit97], [RD00]. En este método variables definidas con valores reales pueden ser tratadas, sólo que para poder procesar este tipo de variables es necesario discretizar mediante rangos los valores admisibles de ella. Para esto se evalúan los rangos que pueden ser utilizados mediante un umbral θ. El rango que mejor resultado obtenga será con el que se use la variable. Este método procesará tanto variables discretas como continuas de la misma manera y utilizará el criterio de proporción de ganancia para seleccionar el atributo de división (ver Apéndice C).

Los árboles generados por C4.5 se caracterizan por ser completos y consistentes, cubriendo todos los ejemplos presentes en la matriz de aprendizaje. Dicha circunstancia provoca que al ajustarse demasiado al conjunto de entrenamiento pueda aparecer

un mal comportamiento al clasificar nuevos ejemplos [JC].

La manera frecuente de limitar este problema es empleando mecanismos de poda. Los criterios basados en el error estiman el porcentaje de error tomando en cuenta la exactitud de los subárboles [DJ97]. Se pueden distinguir dos tipos de poda que se pueden utilizar en este método:

- *Prepoda*. El proceso se lleva a cabo durante la construcción del árbol. Se trata de determinar el criterio de parada a la hora de seguir especializando una rama o regla.
- *Postpoda*. El proceso se lleva a cabo después de la construcción del árbol. Se trata de eliminar nodos de abajo hacia arriba del árbol hasta un cierto límite.

El algoritmo C4.5 se caracteriza por combinar prepoda y postpoda. Este mecanismo de poda, mejora la capacidad de generalización del árbol y reduce su tamaño. El inconveniente es que hay que indicar el criterio de parada en prepoda y el límite de eliminación en postpoda.

La ausencia de información es permitida en C4.5. El algoritmo la toma en cuenta al momento de decidir qué variable deberá formar el nodo a construir, esa ausencia de información reducirá la medida de Proporción de Ganancia de acuerdo al número de casos que la tengan (ver Apéndice C).

Entre las principales deficiencias que presenta este algoritmo se encuentran, que las variables son discretizadas para poder procesarlas y que es complejo indicar la decisión de parada al momento de aplicar la poda al árbol generado. Además después de aplicar este proceso de poda al árbol construido se puede perder la generalidad que por principio se generó al construir un árbol completo, es decir, puede resultar que con el AD final no se cubran todos los objetos descritos en MA.

2.2.2. CART

En Classification and Regresion Trees (CART) la metodología para la construcción del árbol es la misma empleada en C4.5, la metodología de divide y vencerás [LB84]. Las diferencias presentadas entre estos dos algoritmos radican en la estructura del árbol construido, el criterio de división que utilizan, el método de poda y la forma en cómo es tratada la ausencia de información.

CART permite la manipulación de conjuntos de datos descritos por variables cuantitativas. Si las variables son descritas por valores reales, antes de proceder con el criterio de división para la construcción del árbol, se requiere discretizar dichas variables con rangos que contengan cada uno de los posibles valores admisibles con los que cuenta el

conjunto de datos.

CART construye sólo árboles de tipo binario [RD00]. Resultado de esto es que este algoritmo puede generar una partición óptima de las clases si los atributos se describen de forma binaria. Sin embargo, para atributos no binarios este tipo de criterio de división puede no resultar tan bueno y la interpretabilidad del árbol puede reducirse cuando múltiples particiones (del mismo atributo) son hechas en diferentes niveles.

Este algoritmo utiliza el índice de diversidad Gini como criterio de división (ver Apéndice D) [JRQ99] [RD00] . Este índice trata de minimizar la impureza existente en los subconjuntos de casos de entrenamiento generados al ramificar el árbol de decisión.

La técnica de poda que CART utiliza es la Poda del Costo de Complejidad (ver Apéndice D) [RD00] [Kun04], la cual puede reemplazar directamente una estructura compleja de un subárbol por una hoja. Uno de los beneficios que presenta este tipo de poda es que evita el llamado "efecto horizonte", el cual se produce a partir del criterio de parada de división.

A diferencia de C4.5, CART no penaliza el criterio de división durante la construcción del árbol si existen casos en los que el atributo usado en la partición tiene valores desconocidos (ausencia de información). El criterio toma en cuenta sólo a aquellos casos donde los valores se conocen.

Al igual que C4.5 la deficiencia principal que presenta CART es que discretiza los atributos numéricos para poder construir el árbol. Además al generar un árbol de tipo binario, la altura del árbol obtenido puede resultar demasiado alta, sobretodo cuando se presenten conjuntos de datos demasiado extensos y con un número grande en los conjuntos de referencia de sus variables.

2.2.3. FACT

El árbol producido por FACT [WYL88] basa su construcción en la metodología top-down induction. El árbol es construido particionando en forma recursiva la MA en la cuál se conocen las etiquetas de clasificación (es decir las clases a las que pertenecen los objetos) y los valores que describen a cada atributo para cada objeto.

FACT construye cada uno de los nodos que componen el árbol en dos pasos, esto se debe a que el algoritmo no combina el proceso de selección de atributos con el proceso de encontrar las diversas características que posee el atributo seleccionado para poder formar las ramas. Esos dos pasos son:

- Realizar un análisis de varianza entre grupos (ANOVA) para calcular un valor

para cada variable, la que posea el mayor valor será la variable seleccionada.

- Aplicar un análisis discriminante lineal (LDA) para encontrar el conjunto de valores, para ese atributo seleccionado, que mejor separe a las clases de MA. Esto asume que las clases deben ser linealmente separables.

Una de las características que FACT reporta es que no es inestable al momento de ocurrir ligeros cambios en MA, ya que con LDA se tiene un pequeño límite de variabilidad en los datos. Sin embargo, como los métodos descritos anteriormente, basa la construcción del árbol en la MA, y cuando se tiene una de gran extensión puede resultar complejo el proceso.

El conjunto de valores a encontrar por LDA esta en dependencia del número de clases de MA, ya que FACT divide cada nodo de acuerdo al número de clases presentes en ese nodo. Esto puede presentar una desventaja para el método, ya que si se cuenta con un número de clases grande, la MA puede ser cubierta por un árbol de poca profundidad que puede llegar a no mostrar atributos que sean relevantes para entender los datos.

Dentro de las limitaciones que presenta este algoritmo es que las variables de tipo cualitativo las transforma a variables de tipo cuantitativo para poder procesarlas y así poder elegir el atributo que mejor diferencie a las clases. Esta transformación la realiza usando la información discriminatoria de la variable cualitativa para definir el espacio de los valores transformados. Una dificultad que se tiene en este proceso es que no todos los valores de las variables de MA están presentes en todos los nodos, debido a que con los nodos padres construidos se ha ido reduciendo el número de objetos cubiertos por esa rama.

2.2.4. QUEST

QUEST [WYL97] para construir el árbol de decisión se basa en la idea dada por el método FACT. La diferencia principal que presentan estos dos métodos es el número de ramas que derivan de cada nodo del árbol, para este caso, se producen árboles binarios, es decir, de los nodos sólo se derivarán dos ramas.

Debido a esta característica, este algoritmo agrupa en dos super-clases, cuando se tiene una MA descrita por más de dos clases, antes de aplicar el análisis discriminante. Además utiliza una modificación del Análisis Discriminante Cuadrático (QDA) sobre estas dos super-clases para obtener el valor que determina la condición a cumplir en el nodo que se está construyendo.

Para obtener estas super-clases se aplica un algoritmo de agrupamiento k-means (k=2). La clase con mayor número de objetos se denomina super-clase A y la otra

forma la super-clase B. Al igual que FACT, QUEST transforma las variables de tipo cualitativo en cuantitativo, este proceso lo realiza utilizando una matriz de tamaño igual al número de valores presentes en el conjunto de referencia de esa variable, todos los componentes serán 0 excepto en aquellos donde el valor l este presente en los objetos cubiertos por esa rama, ahí el valor será 1.

QUEST también presenta la propiedad de ser estable cuando ocurren pequeños cambios en la MA, ya que el método QDA conserva la propiedad de LDA al momento de establecer un límite de variación.

Dentro de las limitaciones encontradas en este algoritmo se encuentra que, al igual que CART, genera árboles de decisión binarios, lo cual puede resultar en un árbol de gran profundidad y por lo cuál se tiene que aplicar un proceso de poda para reducirlo. Además basa su construcción en la MA y hace una transformación de los datos cualitativos para poder procesar a todas las variables como cuantitativas.

2.3. Algoritmos para datos mezclados

En el problema de clasificación existen diversos métodos que son apropiados sólo para ciertos tipos de datos. Sin embargo, en la vida cotidiana existen muchos conjuntos de datos que consisten no sólo de atributos nominales (discretos) sino también de atributos numéricos (continuos). Se dice entonces, que este tipo de conjuntos de datos están descritos por datos mezclados o por tipos de datos híbridos.

Entre los métodos ya desarrollados para la generación de árboles de decisión existe uno que permite la manipulación de conjuntos de datos mezclados [PHKYJ05].

2.3.1. Model Trees for Classification of Hybrid Data Types

Este algoritmo genera un árbol de decisión, a partir de un conjunto de datos híbrido, incorporando Máquinas de Soporte Vectorial (SVM - por sus siglas en inglés) al proceso de construcción del árbol.

Por lo regular los métodos que pueden tratar con atributos numéricos utilizan procesos de discretización, que muchas veces pueden llegar a transformar los datos de tal manera que perjudiquen su naturaleza. Durante la construcción del árbol, este modelo, utiliza SVM para reemplazar esos procesos de discretización, dando como resultado atributos booleanos sintéticos.

Una de las limitaciones que presenta este algoritmo es precisamente que realiza un cambio de espacio en las variables numéricas cuando usa SVM, es decir, las procesa de

tal manera que las convierte en variables booleanas.

En la generación del árbol de decisión este algoritmo va construyendo nodo a nodo del árbol, en cada uno de esos nodos SVM es aplicado a cada atributo numérico para definir una decisión multiple, es decir transformar el atributo a booleano sintético.

Una vez que los atributos son procesados con SVM, la decisión sobre cuál atributo debe utilizarse se puede basar en diversos criterios de división, como la ganancia de información, la proporción de ganancia, entre otros.

El funcionamiento básico de este algoritmo es procesar los atributos numéricos con SVM y los atributos nominales con el algoritmo ID3. Por lo tanto, se observó que para determinar qué atributo se utilizará en la construcción de los nodos, no se toman en cuenta, de manera simultánea, los datos mezclados, debido a que cada tipo es procesado de manera diferente.

2.3.2. LMDT

Este algoritmo LMDT [CEB95], [PEU91] aprende a partir de un conjunto de entrenamiento (MA) con el objetivo de formar o seleccionar una generalización de esa muestra. Las principales características de LMDT es que permite manejar multiples clases, decisiones multivaluadas en cada nodo y permite además trabajar con datos tanto de tipo cualitativo como cuantitativo. También puede procesar descripciones de objetos imprecisas o etiquetadas incorrectamente.

Basa la construcción del árbol en la metodología top-down y utiliza una máquina lineal como criterio de división, la cual elimina variables de manera controlada para determinar la prueba multivariada que contendrá el nodo a construir. Una máquina lineal es un conjunto de R funciones discriminantes lineales que son usadas en forma colectiva para asignar una variable a una de las clases R.

La forma en cómo trabaja las variables es codificándolas. Cada una de ellas las representa como un vector Y que consta de un umbral y valores codificados que describen a la variable. Esta codificación la realiza LMDT para cada uno de los nodos para todas las variables disponibles. Esta información codificada se utilizará posteriormente para poder clasificar nuevos objetos con ayuda del árbol de decisión resultante.

Cuando se codifica una variable cualitativa se toman los valores 1 y -1, para determinar si ese valor de la variable esta presente en algún objeto en específico. Este esquema de codificación es el que permite el uso de los datos mezclados, sin embargo se puede observar que de alguna manera las variables son transformadas a booleanas, ya que solo

utiliza dos valores para determinar si está o no presente algún valor en particular de la variable.

La ausencia de información también esta considerada por LMDT, el método maneja los valores perdidos tomándolos con valor 0 en la codificación de las variables.

La eliminación de las variables, para obtener la condición que contendrá el nodo, lo hará no tomando en cuenta a aquellas variables que no contribuyan con exactitud a la clasificación de la MA en ese nodo. Durante el proceso de eliminación la máquina lineal más precisa con el menor número de variables será guardada y así sucesivamente, cuando la eliminación termine, la condición para el nodo la contendrá la máquina lineal guardada. El objetivo es formar esa condición con un número de variables que permitan diferenciar las clases con mayor precisión.

Entre las limitaciones que presenta el algoritmo es que hace una codificación de los datos (tanto de datos cualitativos como cuantitativos) y con esto puede llegar a perder la naturaleza de las descripciones de los objetos presentes en MA. Además basa su construcción en esta muestra de objetos, y como se ha observado pueden existir cambios cuando ésta se modifique.

2.4. Otros Algoritmos

Existen otros algoritmos que incluyen árboles de decisión y algoritmos conceptuales en su funcionamiento, sin embargo, el procedimiento los contempla por separado, es decir, son independientes el uno del otro. Uno de los algoritmos de este tipo estudiado es el D2MS (Data Mining with Model Selection) descrito a continuación.

2.4.1. D2MS

D2MS es un sistema que permite al usuario poder elegir de entre varios algoritmos a los que mejor resuelvan un problema en específico, además se pueden comparar los modelos visualizando su desempeño, y de esta manera determinar si la combinación de algoritmos elegida fue la adecuada.

Este sistema esta compuesto por varios módulos que completan su funcionamiento, cada uno de ellos funciona de la siguiente manera:

- La interfaz gráfica para el usuario ayuda a elegir las combinaciones de algoritmos que se pueden utilizar.
- La interfaz de los datos recibe la entrada al sistema, el conjunto de datos.

- El preprocesamiento a los datos sólo se realiza si es necesaria una transformación de ellos.
- Minería de datos provee al usuario de métodos para clasificación supervisada, no supervisada y conceptualización de datos.
- Post-procesamiento y evaluación contiene herramientas que permiten manipular y procesar los resultados obtenidos por los algoritmos elegidos.
- Para verificar que se realizó lo que el usuario quería se utiliza un plan de manejo.
- Visualización de los datos y del conocimiento muestra árboles y estructuras jerárquicas para los datos y los resultados.
- Aplicación es el modulo que contiene utilidades que ayudan al usuario a usar modelos que le permitirán predecir la salida de ciertos algoritmos.

Entre los algoritmos que D2MS tiene definidos para problemas de clasificación supervisada se encuentran: el k-vecino, el bayesiano, un método para generar árboles de decisión llamado CABRO, y un algoritmo que permite generar reglas denominado CABROrule. Los algoritmos para clasificación no supervisada, o agrupamiento, se tiene al k-means con sus variantes, y el algoritmo OSHAM es el algoritmo conceptual que se utiliza para obtener la descripción de los datos en forma de conceptos.

Como se puede observar este método incluye tanto árboles de decisión como conceptos, pero ninguno tiene que ver con el otro, son algoritmos que se utilizan con un fin distinto cuando se esta procesando un conjunto de datos con este sistema.

2.5. Limitaciones encontradas

Con base en las limitaciones presentadas por los algoritmos estudiados en este trabajo se plantea dar una solución a algunos de los problemas como:

- Basar la construcción del AD en la MA. El método propuesto basa la construcción en conceptos generados por un Algoritmo Conceptual.
- Aplicar un proceso de poda al AD final, sin conocer con exactitud la condición de parada. El método propuesto no incluye proceso de poda.
- Manejo de datos mezclados no en forma simultánea, a partir del algoritmo RGC empleado en el ADT se manejan todos los datos en forma simultánea.
- Procesos complejos cuando se trata de una MA extensa debido a que basan la construcción del árbol en las descripciones de objetos representadas en ella.

- Hacer cambio de espacio de las variables para poder procesarlas, en el caso del algoritmo que trabaja con SVM se realiza un preprocesamiento en las variables cuantitativas para poder manejarlas.

Capítulo 3

El Método Propuesto para la Generación de Árboles de Decisión

Teniendo en cuenta el análisis crítico del capítulo anterior acerca de los métodos tradicionales para la generación de Árboles de Decisión, se propone un nuevo método que genere AD (denominado ADT) a partir de un conjunto de datos descrito por variables de naturaleza mezclada, que supere algunas de las limitaciones presentadas por los métodos analizados.

Como se pudo observar, todos los métodos estudiados se basan en la MA para generar el AD. Este trabajo se basa, para la construcción del AD, en los conceptos obtenidos de un Algoritmo Conceptual, para este caso se eligió el Algoritmo RGC [APP00].

RGC fue elegido para obtener los conceptos que se utilizan en la generación del árbol por el método ADT, debido a que muestra más capacidades que otros algoritmos conceptuales. Entre otras, RGC permite el manejo de información mezclada (es decir, objetos descritos por variables de tipo cualitativas y cuantitativas en forma simultánea), además incluye en su funcionamiento la permisibilidad de ausencia de información en las descripciones de los objetos de la MA.

Otra de las características importantes de este algoritmo conceptual elegido es que obtiene conceptos que son totalmente caracterizantes y excluyentes a las clases definidas en la matriz de aprendizaje.

En este trabajo de tesis también se propone una regla de división que elige un subconjunto de variables que describen a cada nodo que componen al AD generado por el método propuesto. El esquema general del método ADT se presenta en la figura 3.1, donde se muestra el funcionamiento completo del método así como las entradas y la salida de éste.

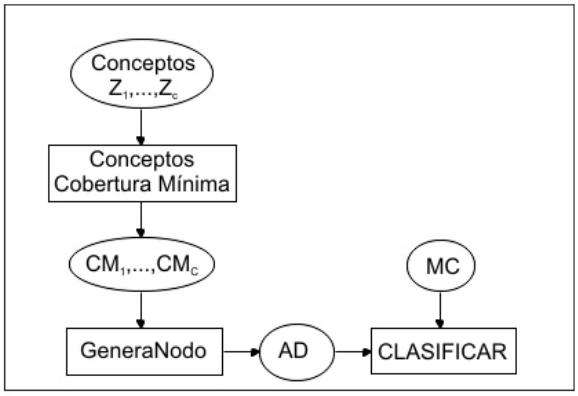

Figura 3.1: Esquema General del Método ADT

Este capítulo contiene dos secciones principales, la primera describe al algoritmo RGC y sus conceptos básicos. La segunda, muestra el método propuesto para la generación del árbol de decisión, la regla de división propuesta y la simplificación hecha a los resultados obtenidos por RGC.

3.1. Algoritmo RGC

El objetivo principal de este algoritmo es que dado un conjunto de descripciones de objetos se encuentre una estructuración conceptual de dichos objetos en el espacio de representación inicial.

Este algoritmo está concebido originalmente para trabajar en el marco de la Clasificación No Supervisada y consta de dos etapas: una de determinación por extensión y otra de determinación por intensión.

En la primer etapa, determinación extensional, lo que se busca son agrupamientos creados usando alguna medida de similaridad entre objetos y un criterio de agrupamiento para generar la estructuración. Debido a que este trabajo basa su problemática en el marco de la Clasificación Supervisada, esta etapa del algoritmo es omitida ya que los conjuntos de datos que se procesan tienen de antemano una clasificación establecida.

Por lo tanto sólo se trabaja con la segunda etapa del algoritmo RGC, donde los conceptos asociados a cada clase son construidos y generalizados.

3.1.1. Conceptos Básicos

Definición 16. Sea un subconjunto de rasgos $\tau = \{X_{i_1}, \ldots, X_{i_s}\}$. Sea además $\omega = (\omega_1, \ldots, \omega_n)$ un n-uplo booleano, donde $\omega_i = 1$ significa que $X_i \in \tau$ y $\omega_i = 0$, que no pertenece. Se llama $\omega - parte$ de la descripción de un objeto, y se denota como $\omega I(O)$, a la subdescripción de O en términos sólo de los rasgos X_i para los cuales $\omega_i = 1$, es decir, $\omega I(O) = \{X_{i_1}(O), \ldots, X_{i_s}(O)\}$.

Definición 17. Se denomina conjunto de apoyo a un subconjunto no vacío de rasgos en términos de los cuales se analizarán los objetos, es decir, un conjunto de $\omega - partes$.

Definición 18. Un l-complejo booleano es una expresión relacional de la forma $\alpha = \underset{i \in I}{\wedge} [X_i = R_i]$, donde $I \subseteq \{1, \ldots, n\}$ es el conjunto de índices, $R_i \subseteq D_i$ es un conjunto de referencias y $R_i \neq \emptyset$.

A los l-complejos booleanos también se les conoce como l-complejos. Un s-complejo S_α se define, por Michalski, como el conjunto de los objetos que satisfacen al l-complejo α.

Definición 19. Sea $I(O) = (X_1, \ldots, X_n)$ un objeto de MA. Se denomina l-complejo elemental asociado y se denota por α_e, $\alpha_e = [X_1(I(O)) = \{R_1\}] \ldots [X_n(I(O)) = \{R_n\}]$.

Definición 20. Sea K una clase de una partición $P = \{K_1, \ldots, K_c\}$ y α un l-complejo. Se dice que α es un l-complejo excluyente para K si $\exists O \in K,\ v(\alpha(O)) = 1 \wedge \forall\ O' \in MA \backslash K,\ v(\alpha(O')) = 0$. Se llama conjunto desacuerdo de α en K al conjunto de todas las combinaciones de valores presentes en los conjuntos de referencia que le impiden al l-complejo α ser excluyente para K.

Donde $v(f)$ significa el valor veritativo de f. Si $v(\alpha(O')) = 1$, se dice que O' satisface al l-complejo α o que α cubre a O'. Obviamente, si un l-complejo α es excluyente para una clase K, entonces su conjunto desacuerdo es vacío.

Definición 21. Sea $\tau = \{X_{i_1}, \ldots, X_{i_s}\}$ un conjunto de apoyo. Se llama estrella de K_i respecto a $K_1, \ldots, K_{i-1}, K_{i+1}, \ldots, K_c$ en términos de τ, y se denota por $G_\tau(K_i \backslash K_1, \ldots, K_{i-1}, K_{i+1}, \ldots, K_c)$ al conjunto de los l-complejos maximales bajo la inclusión que cubren a los objetos de K_i y son excluyentes para K_i de la forma $\underset{x \in \tau}{\wedge} \left[X = R|_{K_i}\right]$, donde $R|_{K_i}$ es un subconjunto del conjunto de valores de X en K_i.

Los conjuntos $R|_{K_i}$ de los l-complejos que conforman la estrella se construyen aplicando el operador de refunión extendida, el cual determina el conjunto de valores que

cada variable toma en esos l-complejos. Este conjunto de valores se denomina conjunto de referencia R para esa variable.

Definición 22. El operador de refunión extendida (RUE) transforma un conjunto de objetos y/o l-complejos en un conjunto de l-complejos excluyentes y se define de la siguiente manera: $RUE : 2^{\Omega}x2^{L(\Omega)} \rightarrow 2^{L(\Omega)}$, donde $L(\Omega)$ es el conjunto de l-complejos sobre Ω, tal que dada una clase a caracterizar C de una partición $P = \{K_1, \ldots, K_c\}$ y un conjunto de apoyo $\tau = \{X_{i_1}, \ldots, X_{i_s}\}$:

- Se considera un primer objeto $O_1 \in K$ y su descripción $I_\tau(O_1) = \{X_{i_1}(O_1), \ldots, X_{i_s}(O_1)\}$ y se forma un l-complejo α_1 de la forma $\bigwedge_{j=1}^{s} \left[X_{i_j} = R_{i_j}^1\right]$, donde $R_{i_j}^1 = \left\{X_{i_j}(O_1)\right\}$, $j = 1, \ldots, s$. Sea además, q la cantidad de l-complejos construidos e inicialmente $q = 1$.
- Para cada objeto $O_p \in K$ y su subdescripción $I_\tau(O_p) = \{X_{i_1}(O_p), \ldots, X_{i_s}(O_p)\}$, $p = 2, \ldots, |K|$ hacer

  g = 0

  Para cada l-complejo α_h construido, $h = 1, \ldots, q$ hacer

  En $\alpha_h = \bigwedge_{j=1}^{s} \left[X_{i_j} = R_{i_j}^h\right]$ agregar a cada $R_{i_j}^h$ el valor $X_{i_j}(O_p)$, $j = 1, \ldots, s$.

  Si α_h no es excluyente para K, entonces

  Hallar el conjunto desacuerdo de α_h en K.

  Marcar en cada $R_{i_j}^h$ los valores pertenecientes a las combinaciones del conjunto desacuerdo de α_h en K.

  g = g + 1.

  Formar un nuevo l-complejo β_g de la forma $\bigwedge_{j=1}^{s} \left[X_{i_j} = R_{i_j}^g\right]$, donde $R_{i_j}^g = \left\{X_{i_j}(O_p)\right\} \cup \left\{v/v \in R_{i_j}^h, v \text{ no está marcado}\right\}$, $j = 1, \ldots, s$.

  Eliminar de cada $R_{i_j}^h$ en α_h el valor añadido $X_{i_j}(O_p)$, $j = 1, \ldots, s$.

  Desmarcar en los conjuntos de referencia de α_h todos los valores marcados.

  Si $g > 0$ entonces

  Agregar al conjunto de l-complejos los l-complejos β_g.

  q = q + g.

Debido a que se trabaja con variables de cualquier naturaleza los l-complejos que se obtienen aplicando el operador RUE pueden ser extensos (en especial para las variables cuantitativas), por lo que es necesario compactarlos, es decir, se necesita disminuir los cardinales de los conjuntos $R_{i_j}^h$. Para ello se utiliza un operador de generalización que simplifique y generalice los l-complejos, sin perder la propiedad de ser excluyentes (que no cubran a objeto alguno de los otros agrupamientos).

Definición 23. Sean K una clase de una partición $P = \{K_1, \ldots, K_c\}$, α un l-complejo excluyente para K, X una variable presente en α, D_x su conjunto de valores admisibles y R_x su conjunto de referencia en α. El operador de generalización GEN trasforma al conjunto R_x en un conjunto R'_x mas general.

Si X es cuantitativa, se aplica la regla de cerrar el intervalo de la siguiente manera: R'_x estará formado por un conjunto de intervalos disjuntos que contengan a todos los valores de R_x, de tal forma que R'_x sea conservativo para α en K (mantiene la propiedad de ser excluyente para K).

3.1.2. Algoritmo

El algoritmo RGC se describe de la siguiente manera:

Entrada: Una colección de objetos MA.

Criterios de comparación C_i para cada variable X_i, $i = 1, \ldots, n$.

Un sistema de conjuntos de apoyo W.

Salida: Conceptos que caracterizan a cada clase de MA.

Paso 1: Etapa intensional.

Para cada conjunto de apoyo τ de W hacer

Se forma una matriz de aprendizaje MA' a partir de las subdescripciones de
MA según τ y de las clases $K_1, \ldots, K_c$.
Para cada clase K_i, $i = 1, \ldots, c$ hacer

Calcular la estrella $G_\tau(K_i \backslash K_1, \ldots, K_{i-1}, K_{i+1}, \ldots, K_c)$. Para el cálculo de los $R|_{K_i}$ en cada $X \in \tau$ se emplea el operador de refunión extendida RUE.

Aplicar el operador de generalización GEN a todas las variables de cada l-complejo obtenido en el paso anterior y construir los l-complejos generalizados $\alpha_1, \ldots, \alpha_q$.

Determinar los objetos $O_1, \ldots, O_t \in MA \backslash K_i$ que satisfacen a la propiedad $\alpha_1 \vee \ldots \vee \alpha_q$.

El concepto que caracteriza a la clase K será:
$Z_i = (\alpha_1 \vee \ldots \vee \alpha_q) \wedge \neg (\alpha_{O_1} \vee \ldots \vee \alpha_{O_t})$, donde α_{O_j}, $j = 1, \ldots, t$ son los l-complejos elementales asociados a los objetos $O_1, \ldots, O_t$.

3.2. Solución Propuesta: Método ADT

Como se describió anteriormente la generación del AD del método propuesto en este trabajo se basa en los conceptos generados por el Algoritmo RGC, no en la MA como los métodos tradicionales estudiados en el capítulo 2. Gracias a esto es que se pueden trabajar conjuntos de datos extensos sin mayor problema, ya que las clases de esa MA son caracterizadas por un sólo concepto.

Como se expuso en la sección anterior el concepto obtenido para cada clase K_i es construido en base a un conjunto de l-complejos, que en su totalidad son caracterizantes para dicha clase.

Desafortunadamente, al aplicar el operador GEN a esos l-complejos se aumenta su dispersión (es decir, puede aumentar el número de objetos cubiertos por él) ya que, a pesar de que al aplicar el operador GEN en cada variable del l-complejo se exige que no pierda su condición de ser excluyente para la clase, al unir todas las generalizaciones de las variables en el l-complejo generalizado, se puede perder esta propiedad.

Es por eso, que una vez generalizados los l-complejos es necesario determinar los objetos de las restantes clases que los satisfacen y construir los l-complejos elementales asociados para así generar el concepto final que caracteriza a cada clase de MA.

Para poder utilizar los conceptos como base para la construcción del AD, es necesario que al aplicar el operador GEN ningún objeto $O \in MA \backslash K$ satisfaga la propiedad $\alpha_1 \vee \ldots \vee \alpha_q$, es decir, que no se formen l-complejos elementales asociados.

Una vez que se obtuvieron los conceptos $Z_1, \ldots, Z_c$ descritos sólo en términos de $\alpha_1 \vee \ldots \vee \alpha_q$, se observó que al construir el AD basándose en ese conjunto de l-complejos el árbol generado poseía un nivel de profundidad muy alto. A partir de esto es que se propone una simplificación de esos conceptos obtenidos por el algoritmo RGC.

Definición 24. Un concepto de cobertura mínima CM_i se forma de simplificar el conjunto de l-complejos excluyentes y generalizados que forman a un concepto Z_i de la siguiente manera:

- Para cada Z_i, $i = 1, \ldots, c$ hacer

  Crear $CM_i = \emptyset$.

  Para cada $X_j \in \alpha_{i_1}$, $j = 1, \ldots, s$, donde $\alpha_{i_1} \in Z_i$

  Si $\forall \alpha_{i_m}$, $m = 2, \ldots, q$, $R^1_{i_j} = R^m_{i_j}$, entonces

  $$CM_i = CM_i \wedge \left[X_j = R^1_{i_j}\right]$$

  Para cada α_{i_m}, $m = 1, \ldots, q$ hacer

  $\forall (X_j \notin CM_i)$

  $$CJ_m = \bigwedge_{j=1}^{s} \left[X_{i_j} = R^m_{i_j}\right]$$

  $$CM_i = CM_i \wedge \{CJ_1 \vee \ldots \vee CJ_m\},$$

donde, $R^m_{i_j}$ es el conjunto de referencia de la variable j en el l-complejo m del Concepto i.

3.2.1. Regla de División

Cuando se realizó el análisis crítico de los métodos de generación de AD presentados en el Capítulo 2, se estudiaron de igual manera las reglas de división que cada uno de ellos empleaba al momento de construir el AD.

La mayoría de estas reglas se basan en una medida de entropía que permite determinar la confusión que induce un atributo al momento de clasificar, con respecto a los valores que puede tomar en la MA. Es decir, buscan la homogeneidad que existe en cada clase. Tomando en cuenta esta característica, y observando que los subconjuntos de la MA que se van generando nodo a nodo en un AD pueden llegar a contener el número total de clases de MA, se propone una nueva regla de división.

Esta regla tiene como propósito encontrar la separabilidad que existe entre las clases del conjunto de datos que se esta procesando. Además el proceso y el resultado obtenido por la regla se basará en los conceptos de cobertura mínima no en la MA, como en el caso de los demás métodos.

El algoritmo para obtener el resultado de esta regla se sigue como se describe a continuación:

Entrada: $CCM = \{CM_1, \ldots, CM_c\}$

Salida: VS

Paso 1: Crear $CJT = \emptyset$.
Paso 2: Para cada CM_i hacer
Para cada $CJ_p \in CM_i$, $p = 1, \ldots, card(CM_i)$ hacer
$CJT = CJT \cup CJ_p$
Paso 3: Para cada $CJ_t \in CJT$, $t = 1, \ldots, card(CJT)$ hacer
$VS_t = ST(CJ_t)$,

donde, CCM es el conjunto de conceptos de cobertura mínima (CM) que caracterizan a cada clase de MA, CJT contiene a todas las posibles conjunciones (CJ) que hay que revisar para el nodo a generar, c es el número de conceptos que están presentes en el nodo, CJ es un subconjunto de variables que forman una conjunción descrita en CM_i, $card(CJ) = ca$, $ca = 1, \ldots, s$, es decir que CJ puede estar descrito solo por una variable o por el conjunto total de variables descritas en el conjunto de apoyo, y VS contiene los valores de separabilidad para CJ_t.

El valor ST para cada conjunción CJ se calcula aplicando el siguiente procedimiento:

$$ST(CJ_t) = \begin{cases} CL(CJ_t) & si \; \exists \; R_i^{CJ_t} \cap R_j^{CJ_t} = \emptyset, \;\; i \neq j \\ S(CJ_t) & en \; otro \; caso \end{cases} \tag{3.1}$$

En 3.1 $R_i^{CJ_t}$ es el conjunto de referencia de CJ_t que se encuentra en el concepto i.

CL(CJ_t) se define de la siguiente forma:

$$CL(CJ_t) = \frac{1}{c} \sum_{i=1}^{c} V_i \tag{3.2}$$

donde:

$$V_i = \begin{cases} 1 & si \; R_i^{CJ_t} \cap R_q^{CJ_t} = \emptyset, \;\; q \neq i \\ 0 & en \; otro \; caso \end{cases} \tag{3.3}$$

3.2 es aplicada a $CJ_t \in CJT$ si es que encuentra al menos una clase que es separada del resto con CJ_t. Y 3.3 sumará un valor de uno cada vez que el conjunto de referencia para CJ_t en el concepto i es totalmente distinto del resto de los conjuntos de referencia, es decir, de los restantes conceptos de CCM, por lo que $q = 1, \ldots, c$.

Si CJ_t no separa al menos a una clase con alguno de sus conjuntos de referencia, entonces se aplica la segunda parte de 3.1, donde:

$$S(CJ_t) = \frac{1}{c}\sum_{i=1}^{c}\left(\frac{card(R_i^{CJ_t} \cap R_q^{CJ_t})}{card(R_i^{CJ_t} \cup R_q^{CJ_t})}\right), \quad q \neq i \tag{3.4}$$

q recorre todos los conceptos de CCM, es decir que compara al concepto i contra el resto de ellos. El resultado proporcionado será un valor entre 0 y 1, donde el valor máximo indica la mayor separabilidad entre clases, por lo tanto esa CJ es la que se debe utilizar para construir el nodo.

En 3.4, la expresión

$$card(R_i^{CJ_t} \cap R_q^{CJ_t}) \tag{3.5}$$

representa la cardinalidad de la intersección entre los conjuntos de referencia de CJ_t en los conceptos i y q, así como también

$$card(R_i^{CJ_t} \cup R_q^{CJ_t}) \tag{3.6}$$

representa la cardinalidad de la unión que existe entre esos conjuntos.

Cuando se traten de variables cualitativas (discretas) dentro de CJ_t, 3.5 y 3.6 son utilizadas como en 3.4. Mediante la división de 3.5 con 3.6 se obtiene la proporción de datos que no distinguen a una clase con la otra, es decir, determina cuántos valores están presentes en ambas clases.

Cuando las variables sean cuantitativas, los conjuntos de referencia estarán descritos por números y/o intervalos de números, entonces la división de 3.5 con 3.6 en 3.4 es reemplazada por 3.7.

$$\sum_{v=1}^{ne}\left(\frac{InD_v(CJ_t)}{InT_v(CJ_t)}\right) \tag{3.7}$$

En 3.7 ne representa el número de elementos que existen en $R_i^{CJ_t}$ multiplicado por el número de elementos en $R_q^{CJ_t}$, por lo que se comparan cada uno de los elementos del conjunto de referencia de CJ_t en el concepto i contra cada uno de los elementos del conjunto de referencia de CJ_t en el concepto q.

Además, la ecuación

$$InD_v(CJ_t) = abs(\text{mín}(E_{i_a}^{CJ_t}) - \text{mín}(E_{q_b}^{CT_t})) + abs(\text{máx}(E_{i_a}^{CJ_t}) - \text{máx}(E_{q_b}^{CT_t})) \quad (3.8)$$

representa la diferencia que existe entre los intervalos $E_{i_a}^{CJ_t}$ y $E_{q_b}^{CJ_t}$. Entonces $E_{i_a}^{CJ_t}$ es el elemento a de $R_i^{CJ_t}$, donde $a = 1, \ldots, ti$ y ti es el número total de elementos en ese conjunto de referencia; y $E_{q_b}^{CJ_t}$ es el elemento b de $R_q^{CJ_t}$, $b = 1, \ldots, tq$ y tq es el número total de elementos en ese conjunto de referencia,

y con

$$InT_v(CJ_t) = \text{máx}(E_{i_a}^{CJ_t}, E_{q_b}^{CT_t}) - \text{mín}(E_{i_a}^{CJ_t}, E_{q_b}^{CT_t}) \quad (3.9)$$

se obtiene el intervalo total que existe entre los elementos $E_{i_a}^{CJ_t}$ y $E_{q_b}^{CT_t}$.

Esta regla de división mide entonces, para cada subconjunto de variables - CJ presentes en los conceptos a procesar en el nodo, el grado de separabilidad entre clases de la MA.

3.2.2. Algoritmo

La forma de construcción para el AD propuesto en este trabajo sigue la idea de la metodología TDIDT.

Se selecciona la CJ que mejor separe las clases del conjunto de datos mediante la regla de división propuesta en la sección anterior. El número de ramas que tendrá el nodo dependerá de cuántas variables formen a esa CJ. Si está formada sólo por una variable, el número de ramas será de acuerdo al numero de valores admisibles para esa variable. Si está formada por más de una variable, dependerá de cuántas descripciones de CJ se pueden formar en cada concepto.

El AD se forma mediante una función recursiva que permite ir creando cada uno de los nodos (inicial, internos y terminales) que contendrá el AD final. Comienza por formar la raíz y para cada una de las ramas generadas crea un nuevo nodo intermedio, así sucesivamente hasta llegar a una rama donde sólo se procesará a un concepto, es ahí donde se crea un nodo terminal u hoja.

El subconjunto de conceptos que incluirá cada una de la ramas (para posteriormente ser ocupado al crear el nuevo nodo) se formará de aquellos conceptos que cumplan la

condición de tener los valores descritos en la rama para la CJ elegida en el nodo.

El algoritmo general ADT se describe de la siguiente manera:

Entrada: Un conjunto de conceptos $\{Z_1, \ldots, Z_c\}$.

Salida: AD basado en conceptos.

Paso 1: Obtener el conjunto de los conceptos de cobertura mínima
$CCM = \{CM_1, \ldots, CM_c\}$.

Paso 2: Generación de AD
Aplicar GeneraNodo(CCM)

Paso 3: Clasificar objetos de MC con el árbol generado.

La función recursiva GeneraNodo toma como parámetro el conjunto de conceptos de cobertura mínima de los cuales se basa para generar cada uno de los nodos. El procedimiento de esta función se describe de la siguiente forma:

GeneraNodo(CCM)
Si $card(CCM) = 1$ entonces
Crea hoja, con valor i, donde i es la clase cubierta por CM_i.
en otro caso
VS = ReglaDivisión(CCM).
CJ_E = máx(VS)
Crear nodo, con valor CJ_E, donde CJ_E es la CJ elegida por tener el máximo valor en VS.
Crea ramas para CJ_E.
Para cada rama r hacer
CCM' = CreaSubconjuntoCM(CCM,r)
GeneraNodo(CCM')

La función CreaSubconjuntoCM genera el subconjunto de conceptos que son cubiertos por la rama r, es decir, que para las variables de CJ_E en el concepto CM_i se cumplen las propiedades de los conjuntos de referencia descritos en r.

Capítulo 4

Experimentación efectuada

En este capítulo se presentan los resultados obtenidos por el método de generación de árboles de decisión ADT. La base de datos utilizada fue la base de datos Zoo [Rep].

4.1. Datos de Prueba - Zoo

Esta base de datos contiene información sobre 101 animales clasificados en 7 clases y caracterizados por 16 variables cualitativas booleanas y una variable numérica.

Las variables booleanas son las siguientes:

- Pelo (hair) - X1
- Plumas (feathers) - X2
- Huevos (eggs) - X3
- Leche (milk) - X4
- Vuela (airborne) - X5
- Acuático (aquatic) - X6
- Depredador (predator) - X7
- Dentado (toothed) - X8
- Columna vertebral (backbone) - X9
- Respiración pulmonar (breathes) - X10
- Venenoso (venomous) - X11
- Aletas (fins) - X12

- Cola (tail) - X14
- Doméstico (domestic) - X15
- Felino (catsize) - X16

La variable numérica es:

- Número de Patas (legs) - X13, toma valores en el conjunto $\{0, 2, 4, 5, 6, 8\}$

De todas las descripciones de objetos de la base de datos del Zoo se tomaron, de manera aleatoria, 91 muestras como MA y 10 muestras como matriz de control (MC). En el Anexo E se presentan las dos matrices donde se describen los 101 animales caracterizados por las 16 variables.

4.1.1. Conceptos generados por RGC

Se tomó como sistema de conjuntos de apoyo $\Omega = \{\Re\}$, donde $\Re$ es el conjunto total de los rasgos, para la generación de los conceptos. Los conceptos obtenidos se muestran a continuación, se exponen los l-complejos que forman a cada uno de los conceptos finales de las clases descritas en MA:

Clase 1:

$$\alpha_1 = [X1 = \{1,0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{0,1\}] \wedge [X4 = \{1\}] \wedge [X5 = \{0,1\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{1,0\}] \wedge [X8 = \{1,0\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0\}] \wedge [X12 = \{0,1\}] \wedge [X13 = \{[0,4]\}] \wedge [X14 = \{0,1\}] \wedge$$
$$[X15 = \{0,1\}] \wedge [X16 = \{1,0\}]$$

Clase 2:

$$\alpha_1 = [X1 = \{0\}] \wedge [X2 = \{1\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{1,0\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0\}] \wedge [X12 = \{0\}] \wedge [X13 = \{2\}] \wedge [X14 = \{1\}] \wedge$$
$$[X15 = \{1,0\}] \wedge [X16 = \{0,1\}]$$

Clase 3:

$$\alpha_1 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1,0\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{1\}] \wedge [X8 = \{1\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1,0\}] \wedge$$
$$[X11 = \{1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{0\}] \wedge [X14 = \{1\}] \wedge$$
$$[X15 = \{0\}] \wedge [X16 = \{0\}]$$

$$\alpha_2 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1,0\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{0\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0,1\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1,0\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{[0,4]\}] \wedge [X14 = \{1\}] \wedge$$
$$[X15 = \{0\}] \wedge [X16 = \{1,0\}]$$

Clase 4:

$$\alpha_1 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{1\}] \wedge [X7 = \{1,0\}] \wedge [X8 = \{1\}] \wedge [X9 = \{1\}] \wedge [X10 = \{0\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{1\}] \wedge [X13 = \{0\}] \wedge [X14 = \{1\}] \wedge$$
$$[X15 = \{0,1\}] \wedge [X16 = \{0,1\}]$$

Clase 5:

$$\alpha_1 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{1\}] \wedge [X7 = \{1,0\}] \wedge [X8 = \{1\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{4\}] \wedge [X14 = \{0,1\}] \wedge$$
$$[X15 = \{0\}] \wedge [X16 = \{0\}]$$

Clase 6:

$$\alpha_1 = [X1 = \{0,1\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0,1\}] \wedge$$
$$[X6 = \{0\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{0\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{6\}] \wedge [X14 = \{0\}] \wedge$$
$$[X15 = \{0,1\}] \wedge [X16 = \{0\}]$$

Clase 7:

$$\alpha_1 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1,0\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{0\}] \wedge [X10 = \{0,1\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{[0,8]\}] \wedge [X14 = \{0,1\}] \wedge$$
$$[X15 = \{0\}] \wedge [X16 = \{0,1\}]$$

$$\alpha_2 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1,0\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{0\}] \wedge [X10 = \{1,0\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{[0,5],8\}] \wedge [X14 = \{0,1\}] \wedge$$
$$[X15 = \{0\}] \wedge [X16 = \{0,1\}]$$

4.1.2. Conceptos de Cobertura Mínima

Una vez obtenidos los conceptos por el algoritmo RGC, se procedió a aplicar el algoritmo para obtener los Conceptos de Cobertura Mínima (CM) propuesto en el Capítulo 3. Como se puede observar sólo la clase 3 y la clase 7 forman su concepto final por más de un l-complejo, así que sólo estos conceptos son afectados por este algoritmo, el resto quedan de la misma forma.

Clase 1:

$$CM_1 = [X1 = \{1,0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{0,1\}] \wedge [X4 = \{1\}] \wedge [X5 = \{0,1\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{1,0\}] \wedge [X8 = \{1,0\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0\}] \wedge [X12 = \{0,1\}] \wedge [X13 = \{[0,4]\}] \wedge [X14 = \{0,1\}] \wedge$$
$$[X15 = \{0,1\}] \wedge [X16 = \{1,0\}]$$

Clase 2:

$$CM_2 = [X1 = \{0\}] \wedge [X2 = \{1\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{1,0\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0\}] \wedge [X12 = \{0\}] \wedge [X13 = \{2\}] \wedge [X14 = \{1\}] \wedge$$
$$[X15 = \{1,0\}] \wedge [X16 = \{0,1\}]$$

Clase 3:

$$CM_3 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1,0\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X9 = \{1\}] \wedge [X10 = \{1,0\}] \wedge [X12 = \{0\}] \wedge [X14 = \{1\}] \wedge [X15 = \{0\}] \wedge$$
$$[\{([X6 = \{0,1\}] \wedge [X7 = \{1\}] \wedge [X8 = \{1\}] \wedge [X11 = \{1\}] \wedge [X13 = \{0\}] \wedge$$
$$[X16 = \{0\}]) \vee ([X6 = \{0\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0,1\}] \wedge [X11 = \{0,1\}] \wedge$$
$$[X13 = \{[0,4]\}] \wedge [X16 = \{1,0\}])\}]$$

Clase 4:

$$CM_4 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{1\}] \wedge [X7 = \{1,0\}] \wedge [X8 = \{1\}] \wedge [X9 = \{1\}] \wedge [X10 = \{0\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{1\}] \wedge [X13 = \{0\}] \wedge [X14 = \{1\}] \wedge$$
$$[X15 = \{0,1\}] \wedge [X16 = \{0,1\}]$$

Clase 5:

$$CM_5 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{1\}] \wedge [X7 = \{1,0\}] \wedge [X8 = \{1\}] \wedge [X9 = \{1\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{4\}] \wedge [X14 = \{0,1\}] \wedge$$
$$[X15 = \{0\}] \wedge [X16 = \{0\}]$$

Clase 6:

$$CM_6 = [X1 = \{0,1\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0,1\}] \wedge$$
$$[X6 = \{0\}] \wedge [X7 = \{0,1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{0\}] \wedge [X10 = \{1\}] \wedge$$
$$[X11 = \{0,1\}] \wedge [X12 = \{0\}] \wedge [X13 = \{6\}] \wedge [X14 = \{0\}] \wedge$$
$$[X15 = \{0,1\}] \wedge [X16 = \{0\}]$$

Clase 7:

$$CM_7 = [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1,0\}] \wedge [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge$$
$$[X6 = \{0,1\}] \wedge [X8 = \{0\}] \wedge [X9 = \{0\}] \wedge [X10 = \{1,0\}] \wedge [X11 = \{0,1\}] \wedge$$
$$[X12 = \{0\}] \wedge [X14 = \{0,1\}] \wedge [X15 = \{0\}] \wedge [X16 = \{0,1\}] \wedge$$
$$[\{([X7 = \{1\}] \wedge [X13 = \{[0,8]\}]) \vee ([X7 = \{0,1\}] \wedge [X13 = \{[0,5],8\}])\}]$$

4.1.3. AD generado

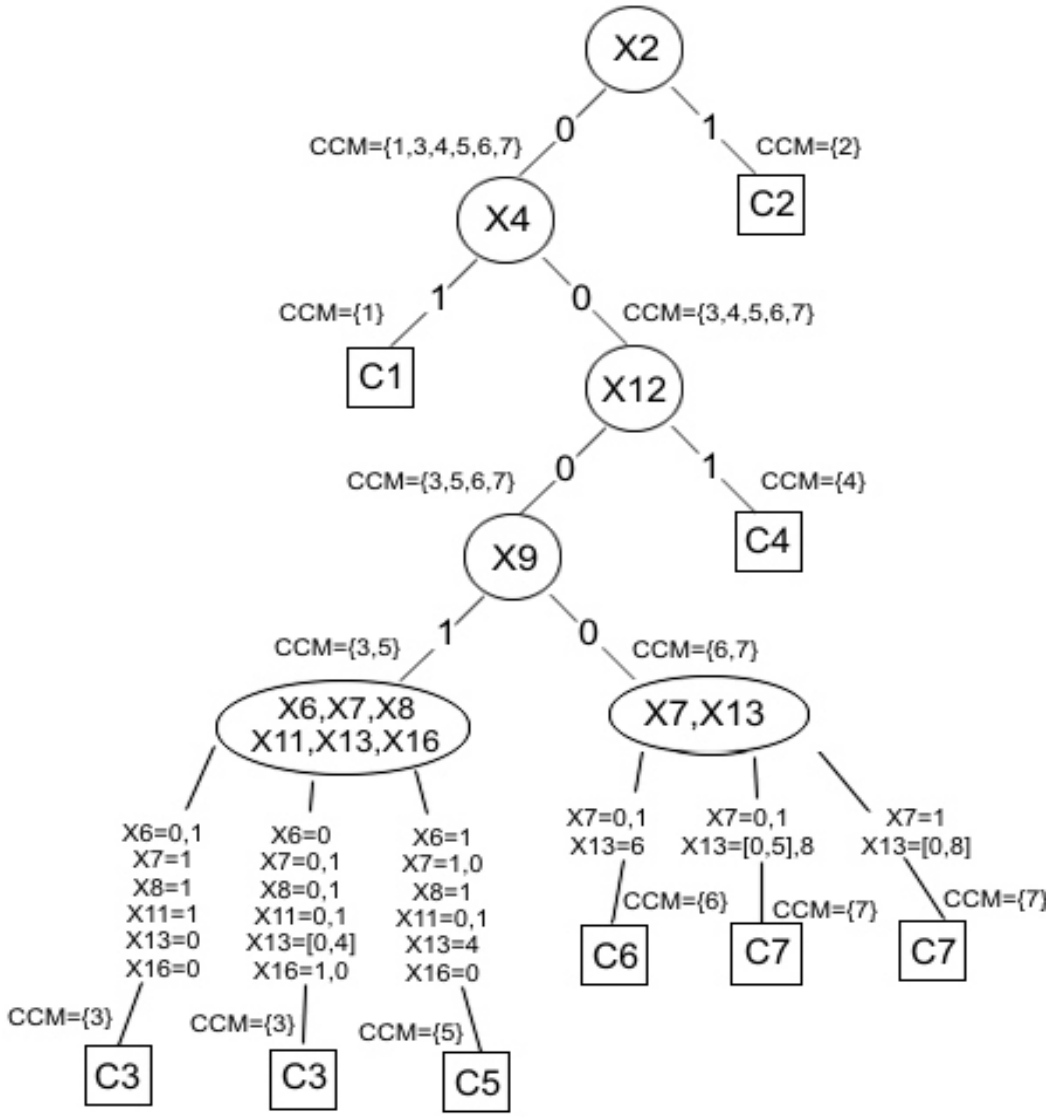

Figura 4.1: Árbol generado por ADT para Zoo

Como se puede observar se generó un árbol con una altura igual a 5, lo cual implica que el AD generado es pequeño y por lo tanto con sólo considerar pocas características de los objetos se puede llegar a una clasificación satisfactoria.

CCM, en cada rama de un nodo, representa al subconjunto de conceptos que se utilizarán en la generación del próximo nodo. Este subconjunto se obtiene de verificar

en los conceptos, cuáles de ellos cumplen la propiedad descrita en esa rama en específico.

De acuerdo a ese subconjunto de conceptos se puede verificar la propiedad de separabilidad entre clases que proporciona la regla de división propuesta en este trabajo. Cada rama separará a un grupo de clases del resto y cuando se llega a tener sólo una clase en el subconjunto de conceptos se crea, entonces, una hoja.

Los objetos que componen la MC, mostrada en el Anexo E, se clasificaron con el AD generado por el método propuesto, los resultados se muestran en la tabla 4.1.

Objeto en MC	Clasificación ADT	Clasificación Real
1	1	1
2	1	1
3	1	1
4	1	1
5	2	2
6	2	2
7	3	3
8	4	4
9	6	6
10	7	7
Clasificación Correcta:	100 %	

Tabla 4.1: Resultados con la Base de Datos Zoo

Como se puede observar el árbol generado por el método propuesto en esta tesis, logró clasificar correctamente a la totalidad de objetos descritos en la MC.

4.1.4. Comparación con ID3

Se aplicó el algoritmo ID3 al conjunto de datos Zoo para comparar resultados con el método propuesto ADT. El árbol mostrado en la figura 4.2 es el resultado obtenido por ID3. Como se observa el árbol es de nivel 4, sin embargo, está compuesto por 9 nodos y además el criterio de división utilizado no genera una separabilidad entre clases, por ejemplo, la clase 7 está presente en 5 caminos del árbol, mientras que en el árbol construido por ADT, la clase 7 sólo se encuentra en un camino.

Se clasificó la MC establecida para probar el desempeño del ADT. ID3 clasificó, de igual manera, la muestra al 100 %.

El objeto mostrado en la tabla 4.2 es una descripción nueva de MA, por lo que ésta no ha sido tomada en cuenta para generar los árboles. Cuando este objeto llega a MA

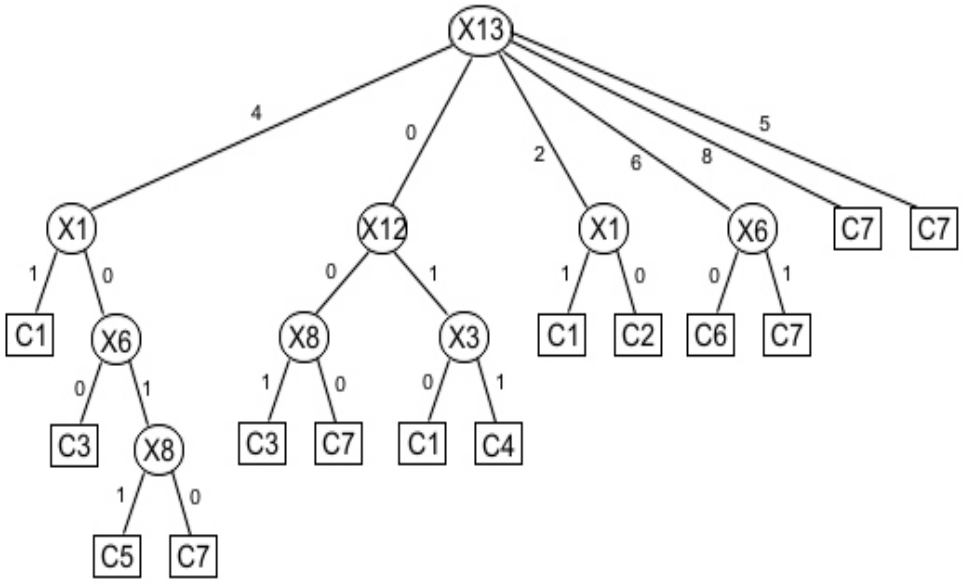

Figura 4.2: Árbol generado por ID3 para Zoo

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$	$X11$	$X12$	$X13$	$X14$	$X15$	$X16$	C
O	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	3

Tabla 4.2: Objeto Nuevo de Zoo

se tiene que verificar que el árbol, previamente construido, cubra a la nueva descripción, es decir, que una rama del árbol la clasifique correctamente para que de esta manera se asevere que el árbol satisface a todos los objetos que componen a MA, incluyendo al objeto nuevo.

El objeto pertenece a la clase 3, así que se verificó, en el árbol construido por ID3, que el objeto estuviera cubierto por él. Los resultados no fueron satisfactorios ya que el objeto fue clasificado incorrectamente. En la figura 4.3 se muestra el nodo que llega a la conclusión errónea. Debido a esto ID3 tiene que volver a construir el árbol, este resultado se muestra en la figura 4.4, el árbol ahora construido satisface al nuevo objeto, y al resto de los objetos descritos en la MA de Zoo.

En el caso del método ADT, el árbol construido clasifica correctamente al nuevo objeto (ver figura 4.1), por lo que no es necesario que se vuelva a aplicar el proceso de construcción. De esta manera se muestra una de las ventajas de utilizar los conceptos como base para la construcción y no la MA, éstos son más robustos y por lo tanto no siempre es necesario volver a construir el árbol que satisfaga a las muestras de la MA

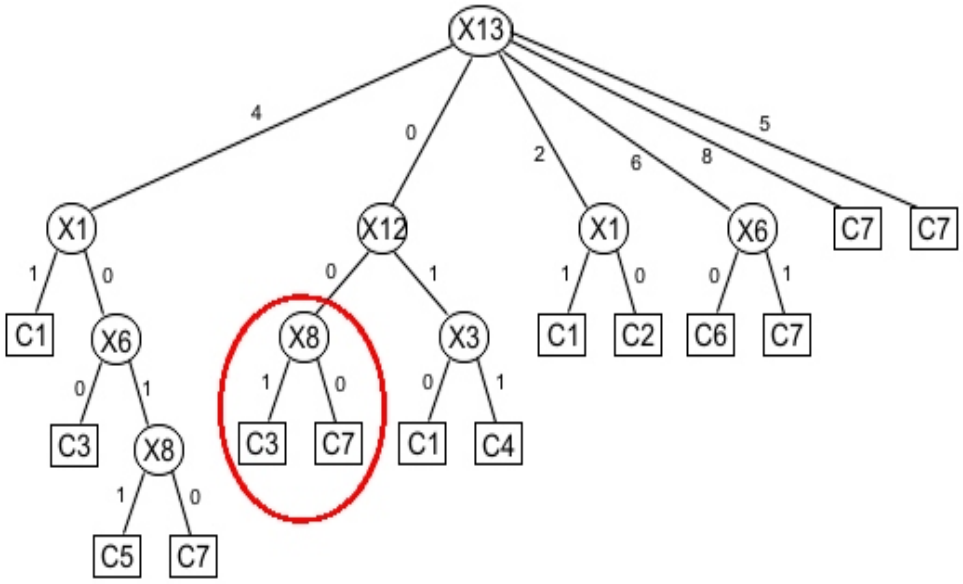

Figura 4.3: Árbol generado por ID3 con error

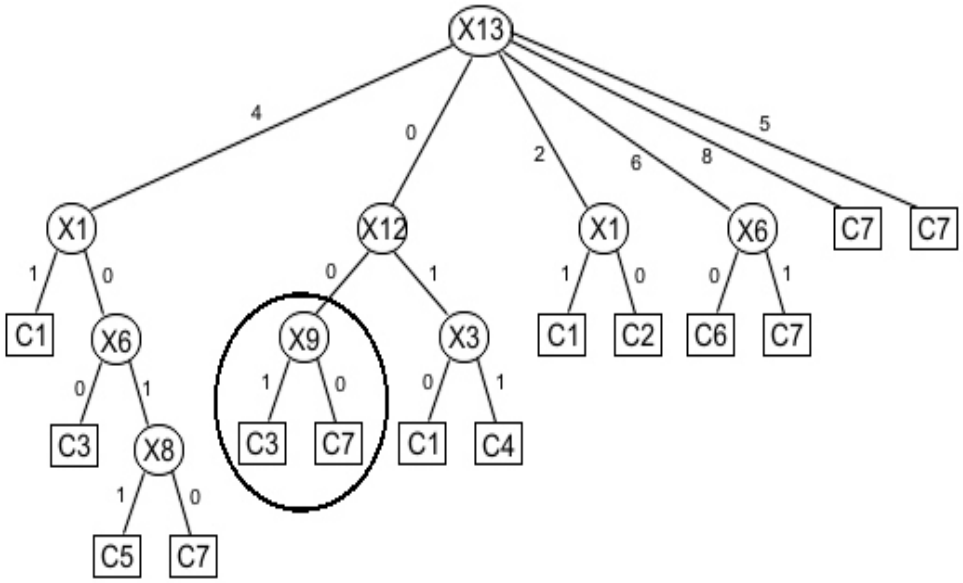

Figura 4.4: Árbol generado por ID3 para Zoo Modificado

modificada.

4.2. Comparación con otros algoritmos

Para realizar la comparación del desempeño del método ADT con otros algoritmos se tomó como referencia [LTS00b]. En este trabajo se procesaron diversos conjuntos de datos para realizar la comparación de los resultados de clasificación y el tiempo de procesamiento entre varios algoritmos. Entre éstos se encuentran algunos que generan árboles de decisión y reglas, algunos algoritmos estadísticos y dos más que trabajan con redes neuronales. En [LTS00a] se muestran las tablas de resultados obtenidas para cada conjunto de datos.

Los algoritmos que se tomaron en cuenta para comparar los resultados, fueron los que se estudiaron en el Capítulo 2. Los conjuntos de datos utilizados y los resultados obtenidos son:

- Wisconsin breast cancer (*bcw*). Este es uno de los conjuntos de datos descrito en [Rep]. El problema consiste en predecir si una muestra de tejido tomada del pecho de un paciente, es maligna o benigna. Se compone de 2 clases, 9 atributos numéricos y 699 observaciones. 16 de esas observaciones poseen ausencia de información, por lo que fueron borradas de la muestra. De los 683 objetos a utilizar, 615 conforman la MA y 68 la MC. La tasa de error fue estimada usando un método de validación denominado ten-fold cross-validation.

  Los resultados obtenidos para este conjunto de datos se muestran en la tabla 4.3.

Método	Tasa de error
ADT	0.0571
QUEST	0.0454
FACT	0.0498
C4.5	0.0425
CART	0.0468
LMDT	0.0351

Tabla 4.3: Comparación de resultados para *bcw*

- Congressional voting records (*vot*). Este conjunto de datos [Rep] proporciona los votos de cada miembro del $98^{a}vo$ Congreso de Representantes de E. U., descritos en 16 claves. El problema es clasificar a los miembros del Congreso como Demócratas o Republicanos basándose en esos 16 votos. Existen 2 clases, 16 atributos categóricos, con tres categorías cada uno, y 435 observaciones, de las cuales 391 forman parte de la MA y 44 de la MC.

La tasa de error mostrada en la tabla 4.4 se estimó por ten-fold cross validation.

Método	Tasa de error
ADT	0.0681
QUEST	0.0435
FACT	0.0435
C4.5	0.0480
CART	0.0480
LMDT	0.0483

Tabla 4.4: Comparación de resultados para *vot*

- StatLog heart disease (*hea*). Este conjunto de datos se basa en la información proporcionada por la Fundación Clínica de Cleveland. El problema se basa en la predicción de la presencia o ausencia de algún padecimiento en el corazón, dados los resultados de varias pruebas médicas aplicadas a diversos pacientes. Se tienen 2 clases, 7 atributos numéricos, 5 atributos cualitativos y 270 objetos. La tasa de error se estimó usando el método de validación ten-fold cross-validation.

Método	Tasa de error
ADT	0.355
QUEST	0.244
FACT	0.233
C4.5	0.196
CART	0.219
LMDT	0.163

Tabla 4.5: Comparación de resultados para *hea*

Como se puede observar en las tablas 4.3, 4.4 y 4.5, la tasa de error obtenida entre los diversos algoritmos comparados es similar, esto debido a que las diferencias entre esos valores son estadísticamente insignificantes. La diferencia entre los métodos ADT y FACT, para el conjunto de datos *bcw*, difiere por 0.0073, lo cual implica una mínima diferencia que es insignificante en términos prácticos. Y la diferencia de 0.0198 entre ADT y LMDT para *vot*, representa también una pequeña diferencia en el desempeño de ambos algoritmos. Tal vez la diferencia más grande que existe, entre los conjuntos de datos probados, es con *hea*, donde se puede observar un valor de 0.111 entre los algoritmos ADT y QUEST, sin embargo, este valor se puede considerar muy pequeño en casos prácticos.

La tasa de error de ADT resultó menor que algunos algoritmos presentados en [LTS00b] no incluidos en este trabajo. Algunos de ellos son: OC1, CAL5, T1.

Si estas pequeñas diferencias no son importantes en aplicaciones reales, el usuario podría seleccionar un algoritmo basándose en otros criterios, tales como el tiempo de entrenamiento o la interpretabilidad que ofrece un clasificador [LTS00b]. En el caso de ADT, la robustez proporcionada por el árbol generado permite cambios en la MA (introducir, modificar o quitar objetos del conjunto de datos) que en algunos casos, no resultará necesario reconstruir el árbol porque los conceptos utilizados cubrirán estas modificaciones. Como se observa en el ejemplo con la base de datos Zoo presentada anteriormente. Sin embargo, con los algoritmos que basan la generación del árbol en la MA, se tiene una mayor probabilidad de tener que reconstruir el árbol al ocurrir esos cambios.

Conclusiones y Trabajo Futuro

La clasificación supervisada es una de las familias de problemas dentro del Reconocimiento de Patrones, la cuál tiene aplicación en diversas áreas. Una de sus principales tareas consiste en poder determinar la pertenencia de una nueva muestra a un conjunto de datos en dependencia de las características que presenta en su descripción.

Existen diversas familias de algoritmos que permiten dar solución a este problema, una de ellas son los árboles de decisión. Dentro de los métodos que existen para la generación de árboles se encontraron algunas limitaciones en su funcionamiento. A partir de esas limitaciones encontradas se propuso un método que supera algunas de éstas.

Como se observó en el análisis de los métodos ya existentes, desarrollado en este trabajo, la totalidad de los algoritmos utilizan como base para la construcción de los árboles la matriz de aprendizaje que describe a los objetos de estudio. Cuando una MA es extensa en cantidad de objetos, muchos de los algoritmos pueden volverse complicados.

Debido a que esos métodos de generación se basan exclusivamente en las muestras aportadas por la MA, cuando ocurre un cambio en esta muestra de datos, es necesario volver a aplicar el método para generar un nuevo árbol que satisfaga todas las propiedades del nuevo conjunto de datos.

El método propuesto supera estas limitaciones utilizando como base para la construcción del árbol de decisión el conjunto de conceptos que caracterizan a cada clase de la MA, de tal manera que sólo se tenga un concepto por cada clase no importando la extensión de ella, además cuando llega una muestra no contemplada en el conjunto inicial, no resulta necesario generar un nuevo árbol si esta nueva descripción es cubierta por el concepto que caracteriza a la clase a la que pertenece.

Otra característica aportada por el método ADT es el criterio de división aplicado a la generación de cada nodo del AD, el cuál a diferencia de los empleados en la mayoría de los métodos revisados, no basa su operación en la medida de entropía que busca la homogeneidad presente en el conjunto de datos. Este criterio implementado encuentra el grado de separabilidad que existe entre las clases de la MA con respecto a las con-

junciones de atributos presentes en los conceptos que intervienen en la generación de cada nodo.

Los conceptos generados por RGC constan de dos elementos. El primero es un conjunto de l-complejos que poseen la propiedad caracterizante del concepto, y el segundo es un conjunto de l-complejos elementales asociados (no necesariamente presentes) que permiten volver excluyente al concepto.

En primera instancia se consideró el primer conjunto de l-complejos como base para la generación del árbol, sin embargo los resultados mostraban un árbol, que aunque caracterizaba a la totalidad de objetos descritos en MA, poseía un alto nivel de profundidad lo cuál representaba una desventaja en el método. A partir de esto, surgió la necesidad de crear un procedimiento que simplifica a ese conjunto de l-complejos, por lo que se propuso uno que generará el conjunto de conceptos de cobertura mínima.

Sin embargo existe el inconveniente de no poder generar el árbol cuando existe el segundo conjunto de l-complejos, ya que las descripciones de los dos conjuntos presentes en los resultados de RGC coinciden en variables y algunas veces en sus conjuntos de referencia, y de alguna manera resulta inapropiado generar el árbol a partir de esta propiedad.

Se probó el método ADT obteniendo resultados satisfactorios para la generación de un árbol de decisión compacto que representará a la totalidad de objetos descritos en la MA; además durante el proceso de clasificación de las muestras de la MC el desempeño del árbol construido resulto eficiente.

Analizando los resultados obtenidos con la comparación realizada con otros algoritmos de construcción de árboles de decisión se muestra que ADT esta situado en el promedio de otros algoritmos en el proceso de clasificación.

Como parte del trabajo por desarrollar, se tiene contemplado formular un nuevo operador de generalización para RGC que permita construir conceptos excluyentes sin la necesidad de generar l-complejos elementales asociados.

Además se tiene considerado proponer una forma para manejar la ausencia de información al momento de construir el AD, así como también para el proceso de clasificación.

Bibliografía

[AG98] G. Martínez G.Ñúñez Adolfo Guzman, A. García. Inducción de árboles de decisión, algoritmo k-d. *Simposium Internacional de Computación, IPN, México*, 1998.

[APP00] J. Martínez-Trinidad A. Pons-Porrata, J. Ruiz-Shulcloper. Refunion - generalization - conceptual clustering algorithm. *Memorias del V Simposio Iberoamericano de Reconocimiento de Patrones, Lisboa, Portugal*, 2000.

[Bar94] J. Ruiz-Shulcloper; E. Alba; N. López; M. Lazo; E. Barreto. *Tópicos acerca de la teoría de testores.* Serie Amarilla No 134, Sección de Publicaciones Técnicas del Departamento de Ingeniería Eléctrica, CINVESTAV, México, 1994.

[CEB95] P. E. Utgoff C. E. Brodley. Multivariate decision trees. *Machine Learning, 19*, 1995.

[DJ97] M. Schmill D. Jensen. Adjusting for multiple comparisons in decision tree pruning. *American Association for Artificial Intelligence*, 1997.

[Fu74] K S. Fu. *Syntactic Methods in Pattern Recognition.* New York, Academic Press, 1974.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* 2da. edición. Academic Press, 1990.

[Gal02] F. Berzal Galiano. *ART. Un método alternativo para la construcción de árboles de decisión.* Tesis, Granada, 2002.

[Jan98] Cesary Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 28, No. 1*, 1998.

[JC] Manuel Lozano J. Cano, F. Herrera. Seleccion evolutiva estratificada de conjuntos de entrenamiento para la obtención de bases de reglas con un alto equilibrio entre precisión e interpretabilidad.

[JRQ99] Ron Konavi J R. Quinlan. Decision tree discovery. 1999.

[Kun04] Ludmila Kuncheva. *Combining Pattern Classifiers.* John Wiley and Sons, INC, 2004.

[Lan91] Rasoul Safavian; David Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3), 1991.

[Laz93] J. Ruiz-Shulcloper; E. Alba; M. Lazo. *Reconocimiento de Patrones I.* FCFM, BUAP, Puebla, México, 1993.

[Laz94] E. Alba; J. Ruiz-Shulcloper; M. Lazo. *Reconocimiento de Patrones IV.* FCFM, BUAP, Puebla, México, 1994.

[Laz95a] J. Ruiz-Shulcloper; E. Alba; M. Lazo. *Introducción a la Teoría de Testores.* Serie Verde No. 51, Sección de Publicaciones Técnicas del Departamento de Ingeniería Eléctrica, CINVESTAV, México, 1995.

[Laz95b] J. Ruiz-Shulcloper; E. Alba; M. Lazo. *Introducción al Reconocimiento de Patrones.* Serie Verde No. 51, Sección de Publicaciones Técnicas del Departamento de Ingeniería Eléctrica, CINVESTAV, México, 1995.

[LB84] R. Olshen L. Breiman, J. Friedman. *Classification and Regression Trees.* Wadsworth International Group, 1984.

[LD96] G. Lugosi L. Devroye, L. Györfi. *A Probabilistic Theory of Pattern Classification.* Springer, 1996.

[LTS00a] Shih Yu-Shan Lim Tjen-Sien, Loh Wei-Yin. Appendix to: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning, 40*, 2000.

[LTS00b] Shih Yu-Shan Lim Tjen-Sien, Loh Wei-Yin. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning, 40*, 2000.

[Mar98] Christophe Marsala. Application of fuzzy rule induction to data mining. *Lecture Notes in Computer Science*, 1998.

[Mar99] J. Ruiz-Shulcloper; A. Guzman; F. Martinez. *Enfoque Lógico Combinatorio al Reconocimiento de Patrones.* Centro de Investigación en Computación, IPN , México, 1999.

[Mar03] H. Dunham Margaret. *Data Mining, Introductory and Advanced Topics.* Prentice Hall, 2003.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw Hill, Inc. U.S.A., 1997.

[PEU91] C. E. Brodley P. E. Utgoff. Linear machine decision trees. *COINS Technical Report 91-10, Amherst, MA: University of Massachusetts, Department of Computer and Information Science*, 1991.

[PHKYJ05] Chang Shou-Chih Pau Hsing-Kuo and Lee Yuh-Jye. Model trees for classification of hybrid data types. *Intelligent Data Engineering and Automated Learning - IDEAL: 6th International Conference, Brisbane, Australia*, 2005.

[Qui86] J R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1), 1986.

[Qui93] J R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Qui96] J R. Quinlan. Learning decision tree classifiers. *ACM Computing Surveys*, 28(1), 1996.

[RD00] D. Stork R. Duda, P. Hart. *Pattern Classification*. John Wiley and Sons, Inc., 2000.

[Rep] UCI Repository. ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

[Ros61] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books, Washington, D.C., 1961.

[Utg98] P E. Utgoff. *Decision trees*. Wilson and Keil (Eds.), The MIT encyclopedia of cognitive sciences. Bradford, 1998.

[WYL88] Nunta Vanichsetakul Wei-Yin Loh. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association, Vol. 83, No. 403*, 1988.

[WYL97] Yu-Shan Shih Wei-Yin Loh. Split selection methods for classification trees. *Statistica Sinica, Vol. 7*, 1997.

Glosario de términos

Siglas	**Significado**
AD	Árbol de Decisión
ADT	Alternative Decision Trees
$ANOVA$	Analysis of Variance
$C4.5$	Método de generación de AD
$CART$	Classification and Regression Trees
CM	Concepto de Cobertura Mínima
CS	Clasificación Supervisada
$D2MS$	Data Mining with Model Selection
ELC	Enfoque Lógico Combinatorio
$FACT$	Fast Algorithm for Classification Trees
FDT	Fuzzy Decision Trees
GEN	Operador de Generalización
$ID3$	Método de generación de AD
$k-d$	Árbol k-dimensional
LDA	Linear Discriminant Analysis
$LMDT$	Linear Machine Decision Trees
MA	Matriz de Aprendizaje
MC	Matriz de Control
QDA	Quadratic Discriminant Analysis
$QUEST$	Quick, Unbiased, Efficient, Statistical Tree
RGC	Refunion-Generalization-Conceptual
RP	Reconocimiento de Patrones
RUE	Operador de Refunión Extendida
SVM	Support Vector Machine - Máquina de Soporte Vectorial
$TDIDT$	Top down induction on Decision Trees

Apéndice A

ID3

A.1. Ganancia de Información

Se denomina ***entropía*** a la medida que caracteriza la pureza de un conjunto de objetos determinado. Dado un conjunto S, que contiene ejemplos positivos y negativos respecto a un concepto, la entropía de S para una clasificación booleana es:

$$Entropia(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{A.1}$$

Donde $p_{\oplus}$ es la proporción de ejemplos positivos en S y $p_{\ominus}$ es la proporción de ejemplos negativos en S.

Ejemplo 1. Suponga que se tiene un conjunto S compuesto de 14 ejemplos, 9 de los cuales son positivos y 5 son negativos. La entropía de S sería entonces:

$$\begin{aligned} Entropia(9+, 5-) = & \quad \text{- } (9/14) \log_2(9/14) \text{ - } (5/14) \log_2(5/14) \\ = & \quad 0.4097 + 0.5305 \\ = & \quad 0.9402 \end{aligned}$$

Este es un caso especial, donde sólo se tienen dos clases en la MA, sin embargo existen muchos conjuntos de datos que contienen un número c de clases, entonces la entropía S de esta clasificación sería:

$$Entropia(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{A.2}$$

Donde p_i es la proporción de los ejemplos de S que pertenezcan a la clase i.

Se denomina ***ganancia de información*** a la medida que determina la efectividad de un atributo al momento de clasificar la MA.

La ganancia de información de un atributo A con respecto a un conjunto de objetos S, se define como:

$$GananciaInf(S,A) = Entropia(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{A.3}$$

Donde $Valores(A)$ es el conjunto de valores admisibles del atributo A, y S_v es el subconjunto de S donde el atributo A tiene el valor v.

A.2. Algoritmo

El algoritmo ID3 para construir un árbol de decisión a partir de un conjunto de atributos no categóricos discretos $C_1, C_2, \cdots, C_n$, un atributo C categórico y un conjunto de entrenamiento T es el siguiente:

Función ID3 (R: Conjunto de atributos no categóricos,
C: Atributo categórico,
S: Conjunto de entrenamiento) regresa un árbol de decisión;
Empieza
Si S es vacío, regresa un nodo con valor Error;
Si S consiste de objetos, todos con el mismo valor para el atributo categórico,
Regresa un nodo con ese valor;
Si R es vacío, entonces
Regresa un nodo con el valor más frecuente de los valores, del atributo categórico, que son encontrados en los objetos de S;
Sea D el atributo con la mayor Ganancia(D,S) entre los atributos en R;
Sean $\{dj | j = 1, 2, \ldots, m\}$ los valores del atributo D;
Sean $\{Sj | j = 1, 2, \ldots, m\}$ los subconjuntos de S, los cuales consistirán, respectivamente, de los objetos con el valor dj para el atributo D;
Regresa un árbol con raíz D y con m arcos, marcados con $d1, d2, \ldots, dm$, respectivamente.
$ID3(R - \{D\}, C, S1)$, $ID3(R - \{D\}, C, S2)$, ..., $ID3(R - \{D\}, C, Sm)$;
Termina ID3.

Apéndice B

Algoritmo k-d

B.1. Pseudocódigo

T - conjunto de atributos

Si *Cardinalidad_T* >1
{
    atributo = SelAtrib-Confusión(MA,T)
    raíz = CreaRaiz(atributo)
    Para i=1 hasta No. Valores de atributo
        CreaRama(raíz, atributo, *valor_i_atributo*, MA, T)
}

Donde

CreaRama(*nodo_i*, atributo, *valor_i_atributo*, MA, T)
{
    Etiquetar la rama atributo
    Si todos los objetos que tienen ese *valor_i_atributo* como atributo pertenecen a la misma clase
        hoja = CreaNodo(*nodo_i*, *clase_pertenecen_objetos*)
    de otro modo
    {
        SubMatriz = CreaSubmatriz(MA, T, atributo, *valor_i_atributo*)
        T = T - atributo
        Si *Cardinalidad_T* >1
        {
            Atributo = SelAtrib-Confusión(SubMatriz, T)
            nodo = CreaNodo(*nodo_i*, atributo)

```
                Para j=1 hasta No. de valores de atributo
                    CreaRama(nodo_i, atributo, valor_i_atributo, SubMatriz, T)
            }
        }
}

SelAtrib-Confusión(MA, T)
{
        Encontrar el atributo que menos confusión produce en la MA
        Retornar atributo encontrado
}

CreaSubmatriz(MA, T, atributo, valor_i_atributo)
{
        Seleccionar todos los objetos que tienen valor_i_atributo en atributo y
        eliminar la columna atributo de estos objetos
        Retornar matriz compuesta de los objetos seleccionados
}
```

Apéndice C

C4.5

C.1. Criterio de Proporción de Ganancia

Aunque usando la ganancia de información se obtienen buenos resultados al construir árboles de decisión, este criterio favorece a las preguntas que tienen más de un resultado posible. Por ejemplo, si cada caso va acompañado de un atributo que lo identifica unívocamente, se elegirá este atributo en la raíz del árbol de forma que cada nodo hijo corresponderá a un único caso. Es decir, se tiende a construir árboles de decisión en los que se utilizan claves o casi claves para ramificar.

Para modificar esto se recurre a la Teoría de la Información para normalizar de algún modo la ganancia obtenida. El contenido de un mensaje que indique la respuesta a la pregunta realizada (no la clase a la que pertenece cada caso) es igual a:

$$-\sum p(A_{ij}) \log_2 p(A_{ij})$$

Así, utilizando este resultado se redefine el criterio de división:

$$R(A_i) = \frac{H(C) - \sum_{j=1}^{M_i} p(A_{ij}) H(C|A_{ij})}{-\sum_{j=1}^{M_i} p(A_{ij}) \log_2 p(A_{ij})} \tag{C.1}$$

Este criterio de división es el utilizado en C4.5 [Qui93]. Cuando la división realizada del conjunto de casos de entrenamiento es trivial, el denominador de $R(A_i)$ es cercano a cero. Por tanto, se ha de escoger el atributo que maximice el cociente $R(A_i)$ siendo su ganancia, al menos, tan grande como la ganancia media de todas las alternativas analizadas.

Dado que en la práctica se disponen de muchos más casos de entrenamiento que clases diferentes, el criterio de proporción de ganancia evitará la construcción de árboles de decisión que clasifiquen los casos utilizando sus claves.

Se ha observado que el criterio de proporción de ganancia tiende a la construcción de árboles poco balanceados, característica que hereda de la regla de división de la que se deriva (la ganancia de información). Ambas heurísticas se basan en una medida de entropía que favorece particiones del conjunto de entrenamiento muy desiguales en tamaño cuando alguna de ellas es de gran pureza (todos los casos que incluye corresponden a una misma clase) aún siendo poco significativa (es decir, aun abarcando muy pocos casos de entrenamiento).

C.2. Ausencia de Información

La ausencia de información en los atributos de un conjunto de datos es muy común que suceda. Cuando un árbol de decisión es construido a partir de un conjunto de entrenamiento S, el algoritmo divide y vencerás selecciona una muestra con la cual se puede dividir S. Sea B una muestra basada en el atributo A_a con valores admisibles $b_1, b_2, \cdots, b_t$.

Se denota a S_0 como el subconjunto de casos en S donde los valores de A_a son desconocidos. S_i denota a aquellos casos donde los valores b_i son conocidos. La ganancia a partir de B es reducida porque se asume que no se aprende nada acerca de los casos en S_0.

La medida de ganancia quedaría entonces:

$$G(S, B) = \frac{|S - S_0|}{|S|} G(S - S_0, B) \tag{C.2}$$

Apéndice D

CART

D.1. Índice de Diversidad de Gini

La función empleada en el Índice de Diversidad de Gini es la siguiente:

$$G(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) G(C|A_{ij}) \tag{D.1}$$

donde:

$$G(C|A_{ij}) = -\sum_{k=1}^{J} p(C_k|A_{ij}) p(\neg C_k|A_{ij}) = 1 - \sum_{k=1}^{J} p^2(C_k|A_{ij}) \tag{D.2}$$

donde A_i es el atributo utilizado para ramificar el árbol, J es el número de clases, M_i es el número de valores diferentes del atributo A_i, $p(A_{ij})$ es la probabilidad de que A_i tome su j-ésimo valor, $p(C_k|A_{ij})$ es la probabilidad de que un ejemplo pertenezca a la clase C_k cuando su atributo A_i toma su j-ésimo valor y $p(\neg C_k|A_{ij})$ es $1 - p(C_k|A_{ij})$.

El índice de Gini es una medida de la diversidad de clases en un nodo del árbol que se utiliza como criterio de división. Al igual que las dos medidas heurísticas anteriores (ganancia de información y criterio de proporción de ganancia), el índice de Gini es una medida de impureza muy utilizada en distintos algoritmos de construcción de árboles de decisión. En concreto, esta es la medida que se utiliza en CART.

Ejemplo. Se considera un problema de dos variables. El método de partición (división) subdivide recursivamente el espacio de entrada (ver figura D.1). La representación de esta partición en un árbol se presenta en la figura D.2. Como se puede observar, la primer división ocurre para la variable x_1 con el valor s_1, de la cual se obtienen dos regiones. La segunda división subdivide una de esas regiones a través de la variable x_2 con el valor s_2, y así sucesivamente.

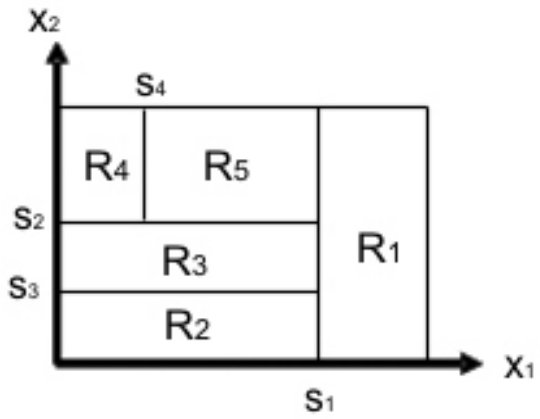

Figura D.1: Partición del espacio x

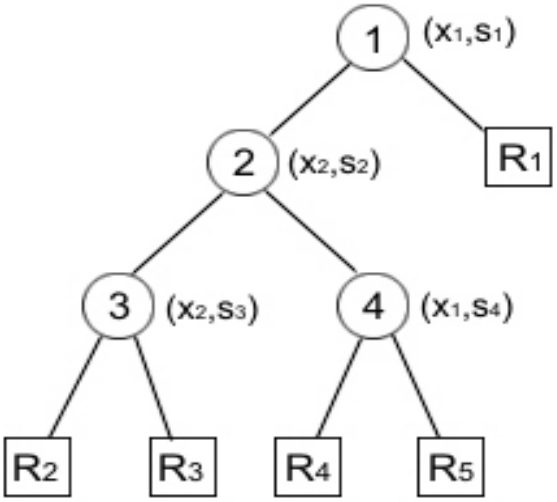

Figura D.2: Árbol resultante

Este método también utiliza el criterio de división llamado "twoing", el cual puede ser usado para problemas de multi-clases. Para cada nodo, las clases son separadas en dos superclases que contienen clases disjuntas y mutuamente excluyentes.

D.2. Método de Poda - Costo de Complejidad

El CCP (por sus siglas en Inglés - Cost-Complexity Pruning) es un método conocido como el algoritmo de poda que utiliza CART. En su primer paso, una familia de árboles con parámetros es generada, la cual se denota como $T = \{T_0, \cdots, T_K\}$. El árbol T_{i+1} es obtenido podando el árbol T_i de acuerdo a ciertos criterios y asociado con un parámetro α_{i+1}. Por ejemplo, α puede ser incrementado con el porcentaje de error por hoja, y puede ser calculado como:

$$\alpha = \frac{e(t) - \sum_{l \in L_t} e(l)}{n(|L_t| - 1)} \tag{D.3}$$

El error adicional, si se reemplaza t por una hoja y se divide por el número de hojas, se consigue. Si para el nodo t, α es más pequeño que un umbral, entonces la alta complejidad y la reducción del error del subárbol en el nodo t no están justificados y por lo tanto t es reemplazado por una hoja.

T_0 es obtenido podando el árbol completo (el final) en aquellas ramas donde $\alpha = 0$ (no añaden error) y T_K es la raíz del árbol. Suponiendo que T_0 es el árbol completo, para construir T_1 se calcula α para T_0 con la ecuación anterior para todos los nodos que no sean hojas, se identifica el mínimo valor de α y éste será utilizado como α_1. Después se reemplazan todos los nodos donde $\alpha = \alpha_1$ por hojas.

Ahora se identifica el α más pequeño en T_1 y se sigue el mismo procedimiento. El segundo paso, después de la construcción de la familia T, es encontrar el mejor árbol. Esto puede ser realizado usando un conjunto independiente de poda o a través de un método de validación.

Apéndice E

Base de Datos Zoo

Las tablas E.1 y E.2 corresponden a la base de datos Zoo utilizada como Matriz de Aprendizaje para generar el árbol de decisión a partir del método propuesto y desarrollado en este trabajo.

La primer columna de estas tablas representa el número de objeto de la MA, las siguientes dieciséis columnas describen a los objetos de acuerdo a las variables mencionadas en el Capítulo 4, y la última columna corresponde a la clase a la que pertenece el objeto descrito.

La tabla E.3 indica los objetos que fueron utilizados como Matriz de Control para verificar el desempeño del árbol generado. Se eligió al azar el 10 % de los objetos de cada clase para formar esta matriz. Debido a que la clase 5 solo contiene 4 objetos en su descripción, no se tomó ninguna muestra para que la conceptualización de la clase fuera lo más general posible.

La última columna muestra la clase a la que pertenece el objeto, con ella se puede verificar que el árbol clasifica correctamente a estos objetos.

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$	$X11$	$X12$	$X13$	$X14$	$X15$	$X16$	C
1	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
2	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
3	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
4	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
5	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
6	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
7	1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	1
8	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
9	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	1
10	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
11	1	0	0	1	1	0	0	1	1	1	0	0	2	1	0	0	1
12	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
13	1	0	0	1	0	0	1	1	1	1	0	0	2	0	1	1	1
14	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
15	1	0	0	1	0	0	0	1	1	1	0	0	2	0	0	1	1
16	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	0	1
17	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
18	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
19	1	0	0	1	0	1	1	1	1	1	0	0	4	1	0	1	1
20	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	0	1
21	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
22	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	0	1
23	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
24	1	0	1	1	0	1	1	0	1	1	0	0	4	1	0	1	1
25	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
26	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
27	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	1
28	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
29	1	0	0	1	0	0	1	1	1	1	0	0	4	1	1	1	1
30	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
31	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
32	1	0	0	1	0	1	1	1	1	1	0	1	0	0	0	1	1
33	1	0	0	1	0	1	1	1	1	1	0	1	2	1	0	1	1
34	1	0	0	1	1	0	0	1	1	1	0	0	2	1	0	0	1
35	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	0	1
36	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	1	1
37	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
38	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
39	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	2
40	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
41	0	1	1	0	1	1	0	0	1	1	0	0	2	1	0	0	2
42	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	1	2
43	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	2
44	0	1	1	0	0	0	1	0	1	1	0	0	2	1	0	0	2
45	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
46	0	1	1	0	0	0	0	0	1	1	0	0	2	1	0	1	2
47	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
48	0	1	1	0	0	1	1	0	1	1	0	0	2	1	0	1	2
49	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
50	0	1	1	0	0	0	1	0	1	1	0	0	2	1	0	1	2
51	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
52	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
53	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
54	0	1	1	0	1	1	0	0	1	1	0	0	2	1	0	1	2
55	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
56	0	0	1	0	0	0	1	1	1	1	1	0	0	1	0	0	3
57	0	0	0	0	0	1	1	1	1	0	1	0	0	1	0	0	3
58	0	0	1	0	0	0	0	0	1	1	0	0	4	1	0	1	3
59	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	0	3

Tabla E.1: Matriz de Aprendizaje de Zoo

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$	$X11$	$X12$	$X13$	$X14$	$X15$	$X16$	C
60	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
61	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	4
62	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
63	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
64	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
65	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
66	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
67	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
68	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
69	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
70	0	0	1	0	0	1	1	1	1	0	1	1	0	1	0	1	4
71	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
72	0	0	1	0	0	1	1	1	1	1	0	0	4	0	0	0	5
73	0	0	1	0	0	1	1	1	1	1	1	0	4	0	0	0	5
74	0	0	1	0	0	1	1	1	1	1	0	0	4	1	0	0	5
75	0	0	1	0	0	1	0	1	1	1	0	0	4	0	0	0	5
76	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	6
77	0	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6
78	1	0	1	0	1	0	0	0	0	1	1	0	6	0	1	0	6
79	0	0	1	0	1	0	1	0	0	1	0	0	6	0	0	0	6
80	1	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6
81	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	6
82	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	0	6
83	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	7
84	0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	7
85	0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	7
86	0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	7
87	0	0	1	0	0	1	1	0	0	0	0	0	8	0	0	1	7
88	0	0	0	0	0	0	1	0	0	1	1	0	8	1	0	0	7
89	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	7
90	0	0	1	0	0	1	1	0	0	0	0	0	5	0	0	0	7
91	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7

Tabla E.2: Matriz de Aprendizaje de Zoo (continuación)

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$	$X11$	$X12$	$X13$	$X14$	$X15$	$X16$	C
1	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
2	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	0	1
3	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	0	1
4	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
5	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
6	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	1	2
7	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0	3
8	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
9	1	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6
10	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7

Tabla E.3: Matriz de Control de Zoo