UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS."

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

GABRIELA ARANGO GALVAN

DIRECTOR DE TESIS: DR. FELIPE LARA ROSANO

MEXICO, D. F. MARZO, 2000

"Los ideales son como las estrellas: nunca los alcanzamos, pero, igual que los marinos en altamar, trazamos nuestro camino siguiéndolos".

Jean Paul Sartre

Con agradecimiento y respeto al Dr. Felipe Lara Rosano por su apoyo incondicional, por la disponibilidad que siempre mostró para la realización de este trabajo y por su paciencia.

Al M. en I. Nicolás Kemper Valverde por proporcionarme todas las facilidades para que esto fuera posible y por darme la oportunidad de incursionar en el área de la Inteligencia Artificial.

Un reconocimiento especial al Fis. Raúl Jiménez Benítez por compartir conmigo su amistad y sus conocimientos así como por su valiosa colaboración en este trabajo.

A todas y cada una de las personas que con sus consejos y observaciones han enriquecido esta tesis.

A mi alma mater, por mi formación y por el enorme privilegio de ser universitaria

A MIS PADRES, por estar siempre a mi lado apoyándome en cada uno de mis pasos, por guiarme con su ejemplo y con su amor, por sus consejos, por su entrega en cada una de las cosas que hacen para mí, por enseñarme a ser libre, respetando siempre cada una de mis decisiones, porque son un ejemplo de que con esfuerzo y trabajo todo se puede lograr. Pero sobre todo por darme su confianza, la que espero nunca defraudar.

A mis hermanas, Angela y Claudia, por sus consejos, por su cariño, por su paciencia, por hacerme más agradables los momentos difíciles, por enseñarme tantas cosas en la convivencia cotidiana y por su ayuda en la revisión de mi trabajo.

Con todo mi amor a Luis mi compañero en este viaje, porque compartimos un proyecto de vida, por respetar mis elecciones, porque con su ternura y comprensión me anima a seguir adelante y porque con su entrega y generosidad me demuestra su amor.

A mis amigos que forman parte importante en mi vida, por compartir los momentos difíciles y también los alegres. Por alentarme a seguir siempre adelante.

A DIOS porque me ama, y a través de todos ustedes me bendice.

ÍNDICE TEMÁTICO

3. Nociones de la teoría de la información y fundamentos del algoritmo ID3

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El uso de la tecnología en las empresas transforma a la organización y cambia su estructura, un ejemplo de ello es el uso del correo electrónico, el intercambio electrónico de datos y el acceso a información externa por medio de redes como Internet. Las herramientas computacionales han ido evolucionando como respuesta a cada una de las necesidades generadas en su momento, desde el surgimiento de las bases de datos hasta las tecnologías de información de más impacto comercial como son el Almacén de datos o Data warehouse y la Minería de datos. Esto nos lleva a considerar lo siguiente:

- Las necesidades crecientes de las organizaciones han llevado a la evolución de las tecnologías de la información.

- Actualmente debido al "glut" o abundancia de información, se requiere de sistemas que sean capaces de almacenar ordenar y analizar dicha información.

- No sólo se requiere tener la información almacenada ordenada y disponible, sino que se necesita analizarla de tal manera que podamos darle un sentido útil a todo ese cúmulo de datos, para que finalmente, nos lleve a la obtención de *conocimiento*.

- La importancia de contar con información adecuada para la mejor toma de decisiones y el impacto que tiene esto dentro de una organización.

- De acuerdo con el Gartner Group, muchas empresas invierten hasta 1.5 millones de dólares en consultoría para la obtención y administración del conocimiento y se espera que dicha cifra se incremente a 5 millones de dólares para el 2001.

Este contexto, nos lleva a valorar la importancia que tiene obtener "conocimiento" de la enorme cantidad de datos a la que se tiene acceso hoy en día. Para un humano sería prácticamente imposible procesar todo el volumen de información que esto implica por lo que existe una urgente necesidad de una nueva generación de técnicas computacionales y herramientas para asistir a los humanos en la extracción de la información y de conocimiento, en especial si tomamos en cuenta el acelerado crecimiento de dicho volumen de los datos y la rapidez de los cambios que constantemente desafían el status del conocimiento adquirido.

Estas técnicas y herramientas son el objeto del surgimiento del *Descubrimiento de conocimiento en bases de datos KDD* (Knowledge Discovery in Databases). En el núcleo de este proceso se encuentra la aplicación de métodos de *Minería de datos* para descubrimiento de patrones en los datos, entendiendo como patrón un término usado para designar a los modelos extraidos de los datos, y el conjunto de datos visto como un objeto susceptible de contener conocimiento.

Dentro de un proceso de generación de conocimiento una de las primeras fases es la categorización o distribución del objeto a conocer en *clases* que agrupan a objetos semejantes. para su mejor estudio. por lo que la tarea de *clasificación* es de particular interés. Actualmente se trabaja en la unificación del marco conceptual del KDD lo que permitirá entender la variedad de actividades en este multidisciplinario campo.

OBJETIVOS DEL TRABAJO

- Mostrar un panorama general del Descubrimiento de Conocimiento en bases de datos o KDD, haciendo énfasis en su parte medular que es la Minería de Datos.

- Ilustrar la Minería de datos a través de uno de sus algoritmos de clasificación, presentando una aplicación del árbol de decisión ID3 en la solución de un problema.

DESCRIPCIÓN DEL CONTENIDO

La información que se presenta en este trabajo ha sido organizada de la siguiente manera:

En la primera parte se presenta el contexto en el que se ha dado la evolución de los datos, hasta convertirse en información y conocimiento, conceptos como bases de datos, sistemas de información, data warehouse y minería de datos son tratados como una introducción al Descubrimiento de Conocimiento en Bases de datos. El capítulo 2, se inicia con cuestiones filosóficas que fundamentan la obtención del conocimiento, posteriormente se introduce el concepto de aprendizaje como parte importante de un algoritmo de minería de datos, y se da un panorama general del descubrimiento de conocimiento en bases de datos (KDD), y de la minería de datos (data mining), enfocando ésta última hacia una de sus funciones principales que es la clasificación .

La segunda parte del trabajo se refiere a la ejemplificación de la minería de datos a través del algoritmo ID3, en el capítulo 3 se dan los fundamentos teóricos requeridos, aquí se contemplan conceptos como Teoría de la Información, entropía, la forma en que trabaja el ID3, la ganancia etc. El capítulo 4 describe un problema y se muestra como fue implementado el algoritmo ID3 para su solución (los detalles de programación están contenidos en el apéndice A). En la última parte se presentan las apreciaciones finales del trabajo.

1. GENERALIDADES

1. GENERALIDADES

1.1 LA IMPORTANCIA DE LA INFORMACIÓN

1.1.1 INTRODUCCIÓN

Cultura, tecnología, cualificación e información son las nuevas claves de la competitividad en las organizaciones, dentro de un contexto de mercado cada vez más global, con mayores exigencias de servicio, de calidad y menores ciclos de vida de los productos. En el ambiente empresarial complejo y competitivo de hoy en día, a los ejecutivos y gerentes encargados de establecer las estrategias de mercado, planear campañas y determinar la ubicación y racionalización de recursos limitados, les interesa mucho comprender y medir la estabilidad de sus negocios, dentro del contexto de sus propios datos empresariales, la industria y la economía y provocar eventos futuros en beneficio de su organización. Para lograr el éxito es necesario que se cuente con información de calidad que apoye las decisiones que deben tomarse. Parte de esa calidad se logra al contar con una información precisa, oportuna, completa y concisa, en la figura 1.1 podemos apreciar la importancia de la información.

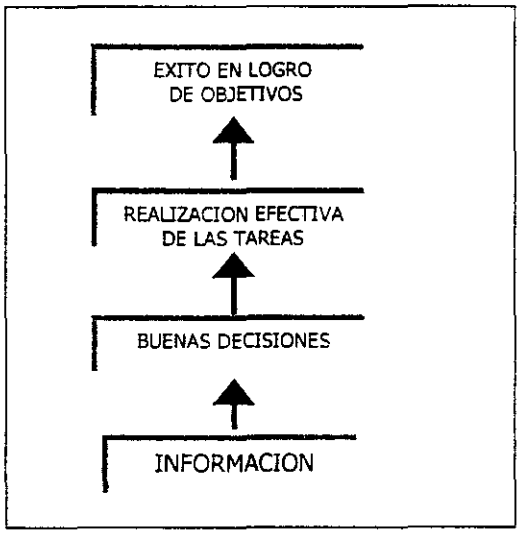

Fig. 1.1 La Integración de la Información para la mejor toma de decisiones

Necesariamente la toma de decisiones debe ser un proceso racional. Para que este proceso sea racional los tomadores de decisiones deben comprender claramente los cursos de acción, es decir, las diferentes alternativas, para ello deben considerar grandes volúmenes de

información así como los mecanismos involucrados para analizarlos detalladamente y así evaluar dentro de esas alternativas la más viable en términos de los objetivos planteados.

1.1.2 LOS DATOS Y LA INFORMACIÓN

En cualquier organización, se necesita saber qué *información* es la más valiosa, sin embargo este término frecuentemente se confunde con el término *dato*. Al hablar de los términos dato e información en una organización frecuentemente se tiende a hacer un uso indiscriminado de ellos, aunque sus aportaciones al sistema son diferentes. *Los datos* consisten en hechos no procesados, tales como nombres de empleados o número de horas trabajadas en la semana, inventario de números de parte u órdenes de venta. Los diferentes tipos de datos que existen como el alfanumérico, el de imagen, audio o video nos ayudan a representar estos hechos. Cuando estos hechos son organizados u ordenados de manera significativa, mediante el establecimiento de relaciones entre ellos, llegan a ser entonces información, por lo tanto *Información* es una colección de hechos o datos organizados de tal forma que ellos adquieren un valor adicional que va más allá del valor de los hechos por sí mismos[1]. Por ejemplo para el administrador de un negocio puede ser más útil para sus propósitos (es decir más valioso) conocer el número total de ventas mensuales que saber las ventas de cada vendedor en particular.

Entonces podemos afirmar que los datos son la materia prima de la información y para que estos sean de utilidad, deberán estar combinados bajo determinados criterios, este proceso puede hacerse manual o automatizado, la fuente de los datos o cómo son procesados es importante, pero el objetivo de este proceso es que los resultados sean *útiles y valiosos* para un tomador de decisiones.

1.1.3 CARACTERÍSTICAS DE LA INFORMACIÓN VALIOSA

Para que la información tenga valor para un administrador o un tomador de decisiones, debe tener ciertas características, mismas que la harán valiosa para la organización. Si la información no es exacta o completa, las decisiones que puedan tomarse serán pobres o incorrectas, teniendo como consecuencia costos millonarios para la organización, por ejemplo, si un pronóstico erróneo de la demanda a futuro de algún producto, indica que las ventas serán muy altas cuando en realidad es lo contrario, una organización invertirá millones de pesos en una nueva planta que no necesita.

[1] Cfr. Star Ralph. Principles of Information Systems a managerial Approach. p. 5-9.

Esto nos da una idea de lo importante que resulta contar con la información correcta y de manera oportuna. A continuación se presenta la tabla 1.1 que contiene dichas características.

CARACTERÍSTICAS	DESCRIPCIÓN
Precisa	La información precisa es la que está libre de error. En algunos casos la información inexacta es consecuencia de datos imprecisos, esto comúnmente es llamado GIGO (garbage in garbage out).
Completa	La información completa contiene todos los hechos más importantes
Económica	La información debe producirse de manera relativamente económica, los tomadores de decisiones deben siempre balancear el valor de la información con el costo de producirla.
Flexible	La flexibilidad implica que la información pueda ser usada para una variedad de propósitos.
Confiable	La fiabilidad de la información depende muchas veces de los datos y de las relaciones establecidas, y en otros casos de las fuentes mismas de los datos.
Relevante	Una información relevante es aquella que para el tomador de decisiones es importante.
Simple	La información no debe ser difícil de entender, a veces la información sofisticada y muy detallada puede no ser necesitada, de hecho también demasiada información puede saturar al tomador de decisiones y puede perder de vista que es lo realmente importante.
Oportuna	La información debe ser entregada justo cuando es requerida, ya que si se conoce demasiado tarde las decisiones tomadas pueden ser erróneas.
Verificable	Finalmente la información debe ser verificable. Esto significa que se pueda comprobar si es correcta, por ejemplo revisando muchas fuentes de la misma información.

Tabla 1.1 Características de una buena información

1.2 EL PROCESAMIENTO DE DATOS

1.2.1 CONCEPTOS BÁSICOS

La importancia de la información en la mayoría de las organizaciones y por tanto el valor de los datos ha llevado al desarrollo de una gran cantidad de conceptos y técnicas para el procesamiento eficiente de dichos datos.

La colección de datos estructurados normalmente denominada *base de datos* contiene una información determinada. Una base de datos es un conjunto autodescriptivo de registros integrados.

"Un *sistema de gestión de datos* (DBMS database management system) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. El objetivo primordial de un DBMS es proporcionar un entorno en el que pueda almacenarse y recuperarse información de forma conveniente y eficiente"[2]. Los sistemas de bases de datos están diseñados para gestionar grandes bloques de información.

1.2.2 BREVE HISTORIA DEL PROCESAMIENTO DE BASES DE DATOS

1.2.2.1 EL CONTEXTO ORGANIZACIONAL

A mediados de los años 60, las grandes corporaciones estaban produciendo datos a grandes velocidades en los sistemas de procesamiento de archivos pero los datos se volvían difíciles de manejar y los nuevos sistemas estaban siendo cada vez más difíciles de desarrollar. Además, la administración quería relacionar los datos de un sistema de archivo con los de otro. Las limitaciones en el procesamiento de archivos evitaron la fácil integración de los datos. Sin embargo, la tecnología de base de datos ofrecía la promesa de una solución a esos problemas y las compañías fuertes empezaron a desarrollar bases de datos organizacionales. Las compañías centralizaron sus datos operativos pedidos, inventarios y datos de contabilidad en estas bases de datos. Las aplicaciones fueron inicialmente sistemas de transacción y procesamiento a nivel de toda la organización.

Como todo principio, cuando la tecnología recién apareció, las aplicaciones de bases de datos eran difíciles de desarrollar y había distintas fallas. Incluso las que funcionaban eran lentas y poco confiables: el hardware no podía manejar el volumen de transacciones con rapidez, la gente de desarrollo aún no había descubierto formas más eficientes de almacenar y obtener datos, de un modo gradual, la situación mejoró.

Los ingenieros de hardware y software aprendieron a construir sistemas suficientemente poderosos para dar soporte a varios usuarios a la vez y tan rápidos como para mantener la carga de trabajo diaria de las transacciones.

[2] Korth, Henry, Et al. Fundamentos de Base de datos. p.1.

A mediados de los años 70, las bases de datos podían procesar muy bien las aplicaciones de una organización. Con varios problemas anteriores resueltos, la administración puso su atención en el descubrimiento de nuevos usos para este conjunto nuevo e inmenso de datos organizacionales. Los administradores aprendieron que de alguna forma, todos los datos podían proporcionar información para tomar decisiones tácticas de corto plazo y estratégicas de largo plazo, sin embargo, para lograrlo debían acceder a los datos ellos mismos, no podían esperar semanas o meses para que los programadores obtuvieran la información de la computadora, pero los usuarios no tenían tiempo ni recursos para volverse programadores profesionales. Para entonces los datos seguían ahí en espera de que se les diera un mejor uso.

1.2.2.2 EL MODELO RELACIONAL

En 1970, E.F. Codd publicó un interesante artículo en el que aplicaba los conceptos de una rama de las matemáticas llamada álgebra relacional, a los problemas de almacenar enormes cantidades de datos. El artículo de Codd en pocos años condujo a la definición del modelo de *bases de datos relacionales*. La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las filas y las tablas son visibles en los datos. Hay que recordar que las bases de datos no solo almacenan los datos sino también las relaciones entre ellos. En este aspecto el modelo relacional cambió la situación respecto a lo planteado antes ya que una de las partes fundamentales de dicho modelo es almacenar las relaciones de los datos de un modo visible para el usuario[3].

En las tablas 1.2 y 1.3 observamos como las relaciones entre los estudiantes y los cursos se almacenan en el campo de identificación del estudiante (ID_estudiante), es posible recuperar un registro de cursos y determinar cuales estudiantes tomaron ese curso examinando su contenido.

Cuando se usa el modelo relacional el usuario debe especificar cuales registros quiere procesar, por ejemplo todos los registros de cursos para el estudiante 100 o los nombres de los estudiantes que completaron el curso MA100.

[3] Cfr. Kroenke, David. Procesamiento de Bases de Datos. Passim.

ID_ESTUDIANTE	NOMBRE_ESTUDIANTE	TELEFONO
100	Estrada, Claudia	323-0098
200	Morales, Luis	232-9987
300	Torres, Javier	887-4484

Tabla 1.2 Datos del estudiante

CLAVE_CURSO	NOMBRE_CURSO	SEMESTRE	GRUPO	ID_ESTUDIANTE
MA100	Calculo II	2	2100	100
CS303	Intro CS	1	1303	200
BD201	Bases de datos	2	2201	200
FIS150	Estática I	1	1150	300
MA105	Algebra I	1	1105	200

Tabla 1.3 Datos del Curso

Los sistemas de bases de datos relacionales pueden usarse en la mayor parte de las aplicaciones, incluyendo el procesamiento de transacciones de las bases de datos organizacionales. Para los Sistemas de Apoyo para la Toma de Decisiones (DSS por sus siglas en inglés) este tipo de modelo ha resultado muy útil.

1.2.2.3 APLICACIONES CLIENTE-SERVIDOR

De mediados a finales de los 80 los usuarios empezaron a conectar sus microcomputadoras separadas usando un nuevo tipo de capacidad de comunicaciones en las computadoras llamado red de área local (LAN) estas redes permitieron a las computadoras enviar datos de uno a otro equipo a velocidades antes no imaginadas. Las primeras aplicaciones de esta tecnologia compartieron periféricos como discos de gran capacidad e impresoras, con el tiempo los usuarios finales quisieron compartir también sus bases de datos, lo que condujo al desarrollo de *aplicaciones multiusuario* de las bases de datos en las redes de área local y a este nuevo estilo de procesamiento se le denominó *arquitectura de base de datos cliente servidor.*

1.2.2.4 PROCESAMIENTO DISTRIBUIDO

Las aplicaciones en bases de datos organizacionales controlan los problemas de procesamiento de archivos y permiten un procesamiento más integrado de los datos en la organización.

Los sistemas de bases de datos personales y de grupos de trabajo acercan todavia más la tecnología al usuario, permitiéndole el acceso a bases de datos administradas localmente. Las bases de datos distribuidas combinan estos tipos de procesamiento permitiendo que las bases de datos personales, de grupos de trabajo y de organizaciones se combinen en sistemas integrados pero distribuidos, como tales ofrecen un acceso de datos y un procesamiento todavía más flexibles, la esencia de las bases de datos distribuidas es que todos los datos de la organización están dispersos en varias computadoras: micros, servidores LAN y macrocomputadoras, que se comunican a medida que procesan los datos.

Los objetivos de este sistema son hacer que cada usuario sienta que es el único usuario de los datos de la organización y proporcionar la misma consistencia, precisión y oportunidad que tendria si nadie más estuviera usando la base de datos distribuida. Algunos de los problemas que presenta este esquema son la seguridad y el control, son tareas complicadas permitir que muchos usuarios tengan acceso a la base de datos y controlar las acciones que ellos ejecutan en la misma.

1.3 LOS SISTEMAS DE INFORMACIÓN

A lo largo del siglo XX el ritmo de cambio acelerado que vive la sociedad (en tecnología, medicina, economía etc.) está obligando a las organizaciones a asumir un proceso de cambio continuo, una permanente búsqueda para mejorar su competitividad. Si las empresas no asumen un planteamiento de sus sistemas informáticos como Sistemas de Información (SI), es decir, más allá de un simple software de funciones administrativas y no aprovechan su verdadero potencial como generadores de ventajas competitivas desde una perspectiva de apoyo a la estrategia de la organización, sin duda su posicionamiento en el sector se verá perjudicado con el tiempo.

1.3.1 CONCEPTO DE SISTEMA DE INFORMACIÓN

Como punto de partida nos basaremos en la definición propuesta por Andreu, Ricart y Valor y entenderemos por SI "al conjunto integrado de procesos, principalmente formales, desarrollados en un entorno usuario-computadora, que operando sobre un conjunto de datos estructurados (Base de datos) de una organización, recopilan, procesan y distribuyen selectivamente la información necesaria para la operatividad habitual de la organización y las

actividades propias de la dirección de la misma"[4]. Analizando los distintos componentes de la definición tenemos:

- Conjunto integrado de procesos, principalmente formales: Se refiere a este tipo de procesos porque son los que la organización conoce y sabe como utilizar. No obstante, los informales, aunque menos aplicables, no dejan por ello de ser importantes y por tanto, en la medida de lo posible, de formar parte del mismo. Se le da el término "integrado" dado que normalmente existen aplicaciones aisladas desarrolladas por diferentes grupos de usuarios (cubriendo de manera rápida y flexible sus propias necesidades de desarrollo), pero si no existe integración de datos y procesos, las aplicaciones individuales pueden llegar a ser incompatibles con el global e impedir la construcción de la base del SI.

- Operando sobre un conjunto de datos estructurados (Bases de datos):
  Hay una gran diferencia entre las bases de datos que están diseñadas específicamente para ser estables y los ficheros que se han usado tradicionalmente en procesos de datos. Debe recordarse que no hay empresas estáticas y por lo tanto las percepciones por parte de dirección de las necesidades de información cambian, esto conlleva a la necesidad de aislar los programas, de los cambios en las estructuras en los datos, es decir, "independencia de datos", misma que se obtiene a través de los sistemas de gestión de datos (DBMS).

- Recopilan, procesan y distribuyen selectivamente la información necesaria (evitando sobrecargas) para la operatividad habitual de la organización y las actividades propias de la dirección de la misma. No es lo mismo cantidad de información que calidad de la información, lo que conlleva criterios de selección, ordenamiento etc.

El SI permitirá operatividad habitual de la organización, apoyando el análisis, la planificación, el proceso de toma de decisiones, todo esto facilitando un adecuado sistema de interrogación a la base de datos con procedimientos "ad hoc" (es decir permitiendo la explotación de datos eligiendo el criterio y las relaciones entre éstos según las circunstancias específicas del momento y del usuario), para así mejorar las tareas propias de gestión de la organización. La fig. 1.2 nos muestra la estructura básica de un Sistema de Información.

[4] Cfr.Gil Pechuán Ignacio. Sistemas y tecnologías de la información para la gestión. p.23.

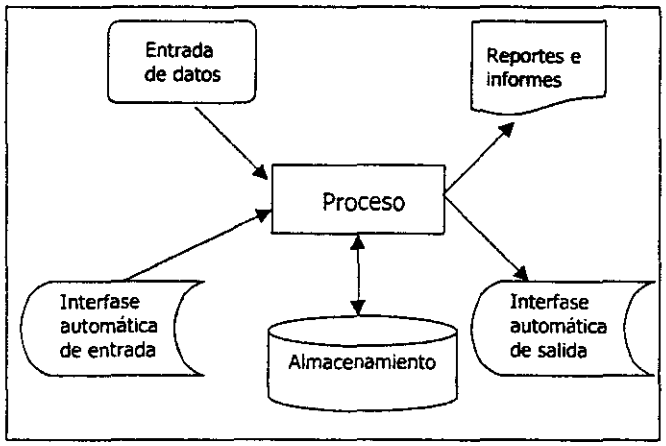

Fig. 1.2 Diseño Conceptual de un sistema de información

1.3.2 TIPOS DE SISTEMAS DE INFORMACIÓN

La figura 1.3 ilustra los tipos de Sistemas de Información que se dan en una organización.

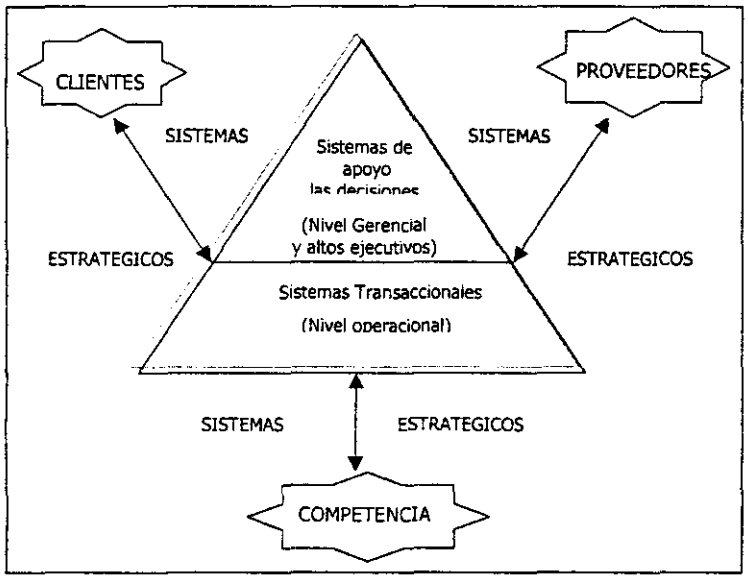

Fig. 1.3 Tipos de Sistemas de Información [5]

[5] Figura tomada de Scott et al. Sistemas de Información.

1.3.2.1 SISTEMAS DE INFORMACIÓN TRANSACCIONALES

Se refiere a los sistemas que logran automatización de procesos operativos dentro de una organización, ya que su función primordial consiste en procesar transacciones tales como pagos, cobros, pólizas, entradas, salidas, etcétera.

- A través de éstos suelen lograrse ahorros significativos de mano de obra debido a que automatizan tareas operativas de la organización.
- Con frecuencia son el primer tipo de sistema de información que se implanta en las organizaciones .
- Sus cálculos y procesos suelen ser simples y poco sofisticados. Estos sistemas requieren mucho manejo de datos, por lo que también generan grandes volúmenes de información.
- Tienen la propiedad de ser recolectores de información es decir, a través de estos sistemas se cargan las grandes bases de información para su explotación posterior.
- Son fáciles de justificar ante la dirección general, ya que sus beneficios son visibles y palpables.

1.3.2.2 SISTEMAS DE APOYO A LAS DECISIONES

Estos sistemas surgen como herramientas de productividad para los operadores de conocimiento (directivos), aportando una ayuda de tipo informático a los procesos de toma de decisiones humanas, de lo anterior podemos afirmar que el objetivo básico de un Sistema de Apoyo a las Decisiones (DSS: Decision Support Systems) es complementar las capacidades de decisión del ser humano, valiéndose de la potencia que aportan las computadoras para el procesamiento de datos. Diversos autores a lo largo del tiempo han aportado aquellos elementos indispensables que debe contemplar su definición, tal como considera la definición aportada por Freyenfeld en 1984:

"Un Sistema de apoyo para la toma de decisiones (DSS) es un proceso de datos interactivo y un sistema de representación visual (entorno gráfico) que es usado para ayudar en el proceso de toma de decisiones y debe reunir las siguientes características:

- Ser sencillo para que lo pueda utilizar personalmente el tomador de las decisiones.
- Debe mostrar la información en formato y terminología familiar para el usuario.
- Ser selectivo en su provisión de información (evitando sobrecarga al usuario)."En un sentido amplio, se define a los Sistemas de apoyo a las decisiones como un conjunto de

En un sentido amplio, se define a los Sistemas de apoyo a las decisiones como un conjunto de programas y herramientas que permiten obtener de manera oportuna la información que se requiere durante el proceso de la toma de decisiones que se desarrolla en un ambiente de certidumbre. A lo anterior se agrega que, en la mayoría de los casos, lo que constituye el detonante de una decisión es el tiempo límite o máximo en que se debe tomar. Así al tomar cualquier decisión siempre se podrá pensar que no se tiene toda la información requerida; sin embargo, al llegar al límite del tiempo se debe tomar una decisión. Esto implica necesariamente que el verdadero objetivo de un Sistema de Apoyo a las Decisiones sea proporcionar la mayor cantidad de información relevante en el menor tiempo posible, con el fin de decidir lo más adecuado[6].

TIPOS DE SISTEMAS DE APOYO A LAS DECISIONES

Una clasificación básica y sumamente práctica es la propuesta por Finlay, consistente en dividir a los sistemas de apoyo a decisiones en aquellos que aportan información (Manager Information System) y los que aportan inteligencia (Manager Intelligent System) la diferencia fundamental entre ambos se explica a continuación:

a) Los que aportan información (Manager Information System) :
   Obtienen y dan información del pasado.
   Dan información del futuro simplemente extrapolando la del pasado.
   Intimamente relacionado con el concepto de "eficacia": orientado a la consecución del Fin.

b) Los que aportan inteligencia (Manager Intelligent System) :
   Existe un proceso de selección de alternativas.
   El que decide utiliza el sistema para desarrollar una visión global de su escenario.
   Muy relacionado con el concepto de "eficiencia"; orientado al procedimiento adecuado para conseguir el fin.

1.4 LA IMPORTANCIA DE LOS S.I. ESTRATÉGICOS

La perspectiva estratégica considera a los Sistemas de información como una herramienta para mejorar la estructura competitiva de la empresa, por lo que tienen su área de influencia en el medio ambiente de la organización a través de nuevos servicios a clientes, nuevos

[6] Cfr.Gil Pechuán Ignacio Op. Cit.Cap 3.

productos y mercados, adquisiciones de nuevos negocios y oportunidades de inversión entre otros. También pueden influenciar la manera en que la organización desarrolla su trabajo internamente, incrementando la productividad o reduciendo costos. Wiseman define la visión estratégica como la necesidad de entender de que forma la tecnología de la información es utilizada para soportar o dar forma a la estrategia competitiva de la empresa. Diversas organizaciones han decidido invertir cantidades considerables de recursos económicos en aplicaciones específicas de la tecnología de la información tal es el caso de los siguientes sistemas:

- Sistema de código de barras y punto de venta
- Transferencia electrónica de fondos (EFT)

1.4.1 APLICACIÓN DE UN SISTEMA DE INFORMACIÓN ESTRATÉGICO

Un ejemplo significativo de la aplicación de la tecnología de la información para el logro de ventajas competitivas se encuentra en la implantación de los sistemas SABRE y APOLLO de las líneas aéreas American Airlines y United Airlines, respectivamente[7]. El sistema de reservación de vuelos que utilizan estas líneas aéreas y las agencias de viajes realiza las reservaciones de los pasajeros y asigna los asientos mediante la actualización de bases de datos centralizadas.

El sistema SABRE fue implantado en 1976 y en la actualidad cuenta con más de 100 000 estaciones de trabajo e impresoras con capacidad para procesar cerca de 1500 transacciones por segundo. El sistema APOLLO se implantó seis meses después. En la actualidad American y United no solo realizan negocios a través de la transportación de pasajeros, sino que estos sistemas de información estratégicos contribuyen a las utilidades en forma directa. Lo anterior es posible, ya que cada vez que alguna agencia de viajes utiliza los servicios SABRE y APOLLO, el sistema hace un cargo por una cantidad determinada. El sistema SABRE dio a American Airlines una clara ventaja competitiva, la cual duró seis meses debido a la introducción de APOLLO. En la actualidad muchas líneas aéreas cuentan con servicios similares. American y United lograron capturar un mercado adicional de pasajeros con el apoyo de sus sistemas de información, pero es inevitable que la competencia persiga la implantación de innovaciones similares como estrategia defensiva contra los competidores.

[7] Cfr. Ibid p.140-141.

1.5 EVOLUCIÓN DE LOS SISTEMAS DE INFORMACIÓN EN LAS ORGANIZACIONES

Con frecuencia se implantan en forma inicial los Sistemas Transaccionales, y posteriormente, se introducen los Sistemas de apoyo a las Decisiones. Por último se desarrollan los Sistemas Estratégicos que dan forma a la estructura competitiva de la empresa.

1.5.1 LA EVOLUCIÓN DE LOS SISTEMAS DE INFORMACIÓN EN PYOSA

A continuación se describe la evolución que ha tenido la función de Informática en una empresa manufacturera de clase mundial. PYOSA es una compañía mexicana que fabrica productos químicos con un enfoque hacia el color, la cerámica y óxidos de plomo y colorantes para alimentos. Cuenta con plantas en Monterrey y San Nicolás y ha logrado ventas de 80 millones de dólares al año, de los cuales 40% son exportaciones que realiza a los cinco continentes. Tiene oficinas de ventas en Monterrey, México, Guadalajara, Los Angeles y Nueva Jersey PYOSA ha dado considerable importancia al desarrollo de la Informática y su evolución lo demuestra[6].

En 1979 surgió el departamento de Sistemas en PYOSA y en ese momento se contaba con una computadora central para realizar los procesos de nómina, contabilidad y manufacturación. Se utilizaban tarjetas perforadas y la computadora se empleaba como una herramienta para generar reportes y no como auxiliar en el trabajo. En 1980 se laboró un análisis del área de informática, del que resultó la necesidad de un cambio hacia un ambiente interactivo, que evitaría en la medida de lo posible intermediarios en la captura de la información. Se empezaron manejar los conceptos de bases de datos y el lenguaje Cobol para el desarrollo de aplicaciones.

En 1985 se introdujeron las microcomputadoras para emular terminales de la máquina HP 3000. Se actualizó el software en la máquina central, se agregaron sistemas adicionales y un volumen mayor de información. Existían alrededor de 80 terminales conectadas a la computadora central. En 1987 se implantó a nivel empresa el sistema MRP II para lo cual se adquirió equipo exclusivo para ello, se instaló una red de área amplia (WAN) para establecer comunicación vía satélite con la oficina de México. En 1993 se elaboró el Plan Estratégico de Informática con el objetivo de alinear todas las inversiones en Informática con los objetivos del negocio.

[6] Cfr. Scott George. Op.cit. pp.56-58.

La idea de dicho plan surgió debido a que los costos de operación de los sistemas crecieron en forma desproporcionada y a que el tiempo de respuesta de los nuevos requerimientos era cada vez más lento. Los riesgos inherentes a las aplicaciones del usuario eran la pérdida, la poca confiabilidad y la inconsistencia de la información. El Plan Estratégico de Informática contiene los objetivos del negocio (que son su base), la identificación de los procesos básicos del negocio (primarios y de apoyo), un inventario de los sistemas actuales que incluye: qué sistema es, qué área apoya, en dónde esta desarrollado, fecha de última modificación, lenguaje, puntos criticos del sistema, principales quejas y número de programas. Este plan incluye también el análisis de la situación actual de los Sistemas de Información y de la manera en que apoyan a cada uno de los procesos del negocio, del costo de operación de dichos sistemas y de los tiempos de respuesta para hacer cambios en los requerimientos. El resultado final de este plan fue la recomendación de proyectos en tres áreas: infraestructura, aplicaciones y mejora.

En lo referente a infraestructura se decidió cambiar al sistema operativo UNIX, utilizar la tecnología cliente-servidor, establecer estándares de comunicaciones, instalar redes de área local (LAN) cambiar las computadoras centrales y enlazar todas las micromputadoras. En el área de aplicaciones se optó por cambiar todos los sistemas existentes por un solo sistema integrado: el ERP (Enterprise Resourse Planning) como siguiente paso en el MRP II. En el área de mejoras se implantaron dos filosofías: Control Total de Calidad (TQC: Total Quality Control) y reingenieria de procesos. En 1994 se realizó el proceso de selección del software que formaría parte del ERP y se inició la instalación de la infraestructura que se tenía planeada. Debido a la competencia que tiene PYOSA con industrias transnacionales, requiere de herramientas que le ayuden a competir no solo en la fabricación, sino también en la administración y en la necesidad de alinear la planeación de informática con los objetivos del negocio.

1.6 EL PAPEL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN (TI) EN LAS ORGANIZACIONES.

Es importante, ponderar a las Tecnologías de la Información por la utilidad que reportan éstas a las empresas, ya que constituyen un medio de apoyo que les permite alcanzar sus objetivos reales a corto, medio y largo plazo de manera más eficaz y eficiente.

Por esta razón conviene remarcar el hecho de que el papel que las TI juegan en toda organización deberá estar contemplado en términos de las necesidades del negocio o del cumplimiento de sus objetivos[9].

La estrategia de negocio define las necesidades de información (SI) y éstas a su vez definen las necesidades de las tecnologías de la información (fig. 1.4). La función de las tecnologías de la información en el desarrollo competitivo de las organizaciones es de tal magnitud que incluso, mediante un adecuado planteamiento y gestión de las mismas se puede llegar a cambiar las bases competitivas del sector en el que la empresa opera diferenciándose ampliamente de la competencia, creando nuevos productos, nuevas barreras de entrada, etc.

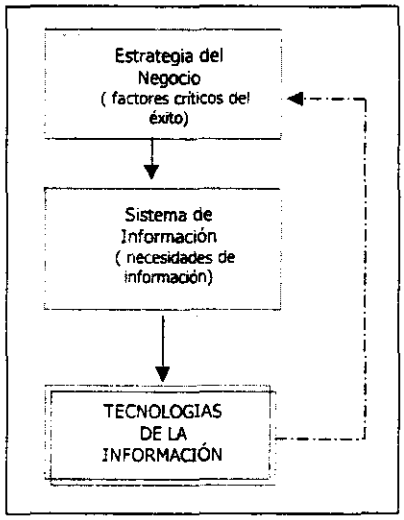

Fig. 1.4 Las Tecnologías de la Información en la empresa [10]

A medida que las organizaciones vayan conociendo y asumiendo el papel que estas tecnologías de la información juegan en su funcionamiento habitual, la propia definición de objetivos irá contemplando a su vez nuevos criterios de utilidad de los sistemas de información en cada organización.

[9] Gil Pechuán, Ignacio. Op.cit. pp. 22-23.
[10] Ibid.

1.7 EL ALMACÉN DE DATOS O DATA WAREHOUSE

1.7.1 INTRODUCCIÓN

El concepto de Data Warehouse o Almacén de datos nace de la combinación de dos conjuntos de necesidades que normalmente no son asociadas, pero que analizadas juntas permiten tener otra perspectiva de la razón fundamental del problema y por lo tanto nos ofrecen una nueva posibilidad de solución. Estas necesidades se refieren a lo siguiente:

a) Los negocios requieren la perspectiva de la información como un elemento clave para crecer.
b) La necesidad de los Sistemas de Información para administrar los datos de la empresa de la mejor forma.

Tomando solo la demanda del negocio, el tener información puede ser una ventaja para obtener soluciones que estén basadas en permitir que los usuarios tengan acceso a cualquier dato en cualquier lugar que éste resida, sin embargo, esta solución puede ser un tanto simplista pues está ignorando la distinción fundamental entre datos e información.

En realidad los usuarios requieren información (frecuentemente definida como datos en un contexto de empresa) y la mayoría de las aplicaciones de acuerdo a como han sido desarrolladas, sólo contienen datos separados del contexto del negocio, y además estos rara vez son consistentes lo largo de toda la compañía. Los datos presentados de esta manera, no son útiles o convenientes par ser usados directamente por los usuarios finales, por otro lado los Sistemas de Información necesitan mejorar la administración de sus datos, ya que cuando se pretende bajo esta situación, presentar un sistema de reducción de costos o un proyecto de implementación de nueva tecnología, teniendo en cuenta el tamaño y la complejidad que ello implica, el costo implementación y el retorno de la inversión, esta propuesta se va al final de la lista de prioridades del negocio.

De lo anterior surge el siguiente escenario : El Sistema de información necesita para la adecuada administración de los datos dónde direccionarlos, y dónde poder encontrarlos fácilmente, así como contar con información útil para los usuarios finales[11].

[11] Derlin, Barry. Data Warehouse from architecture to implementation. Cap.2 Passim

Es entonces a partir de esta coyuntura establecida por el usuario final y el Sistema de Información que surge el concepto de Almacén de Datos (Data Warehouse) como una respuesta eficiente que cumple las premisas mencionadas. En la figura 1.5 se muestran las diferentes etapas de la evolución de esta tecnología.

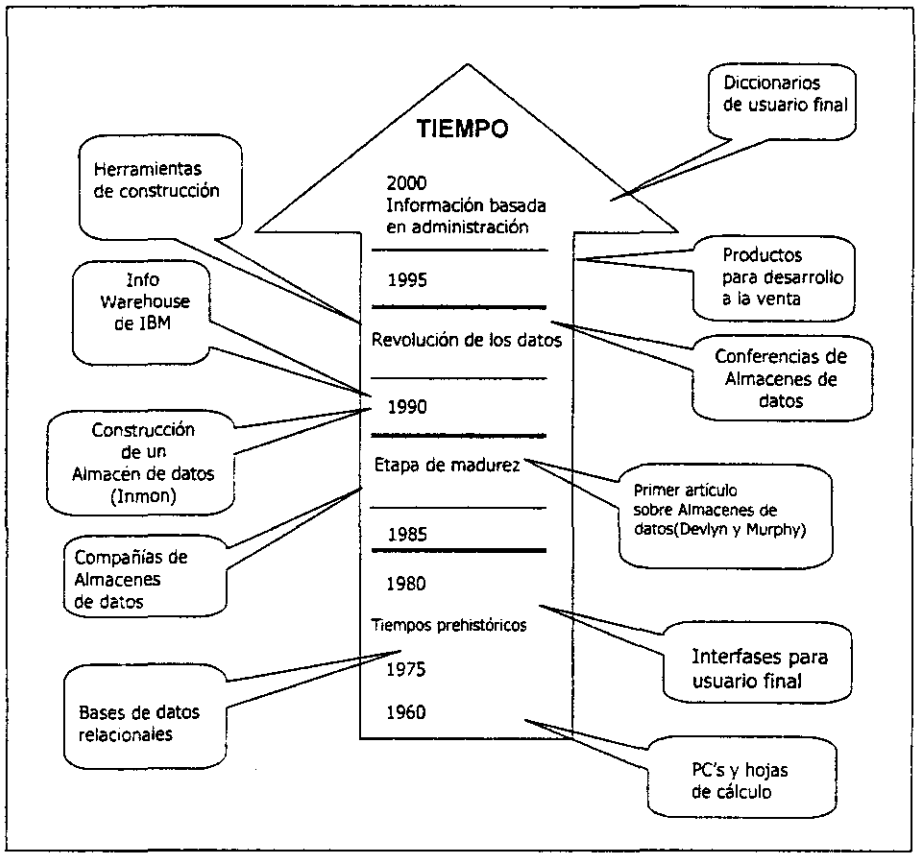

Fig. 1.5 La evolución del almacén de datos[12]

[12] Figura tomada de Derlin, Barry. Data Warehouse from architecture to implementation

1.7.2 DEFICINICIÓN Y CARACTERÍSTICAS

De acuerdo con W.H. Inmon, quien es considerado como el padre del Data Warehouse o Almacén de datos: " Un data almacén de datos es un conjunto de datos integrados orientados a una materia, que varían con el tiempo y que no son transitorios, los cuales soportan el proceso de toma de decisiones de una administración"[13].

Para algunas organizaciones el almacén de datos es una arquitectura, otros lo consideran un depósito de datos (separados y que no interfieren con los sistemas operativos y de producción existentes) que llenan por completo los diferentes requerimientos de acceso y reporte de datos. Otra connotación que recibe es la de una mezcla de datos de varias fuentes heterogéneas, incluyendo datos históricos adquiridos para soportar la constante necesidad de consultas estructuradas reportes analíticos y soporte de decisiones..

De la misma forma en que hay divergencia para establecer una definición precisa de un almacén de datos, hay un claro consenso de que la tecnología del almacén de datos es esencial en el conjunto de soluciones para el soporte de decisiones en una empresa.

UN ALMACÉN DE DATOS:

- **Está orientado a una materia**: organiza y orienta los datos desde la perspectiva del último usuario y no desde la perspectiva de la aplicación. Con frecuencia , la información que está organizada para que una aplicación del negocio la recupere y actualice con facilidad no es necesariamente adecuada para que un analista con herramientas gráficas inteligentes de consulta pueda formular las preguntas empresariales correctas.

- **Administra grandes cantidades de información**: La mayoría de los almacenes de datos contienen información histórica que se retira con frecuencia de los sistemas operativos porque ya no es necesaria para las aplicaciones operacionales y de producción . Por la necesidad de administrar toda la información histórica y además los datos actuales , un almacén de datos es mucho mayor que las bases de datos operacionales.

- **Comprende múltiples versiones de un esquema de base de datos**: Debido a que el almacén de datos tiene que guardar información histórica y administrarla y como ésta ha sido manejada en distintos momentos por diferentes versiones de esquemas de bases de datos, en ocasiones el almacén de datos tiene que controlar la información originada en organizaciones de base de datos diferentes.

- **Condensa y agrega información**: Con frecuencia , es muy alto el nivel de detalle de la información guardada por bases de datos operacionales para cualquier toma de decisiones sensata. Un almacén de datos, condensa y agrega la información para presentarla en forma comprensible a las personas indicadas. La condensación y adición es esencial para retroceder y entender la imagen global.

- **Integra y asocia información de diversas fuentes**: Debido a que las organizaciones han administrado históricamente sus operaciones utilizando numerosas aplicaciones de software y múltiples bases de datos, se requiere de un almacén de datos para recopilar y organizar en un solo lugar la información que estas aplicaciones han acumulado al paso de los años. Esta es un tarea desafiante por la diversidad de tecnologías de almacenamiento de técnicas de administración de bases de datos y de la semántica de los datos.

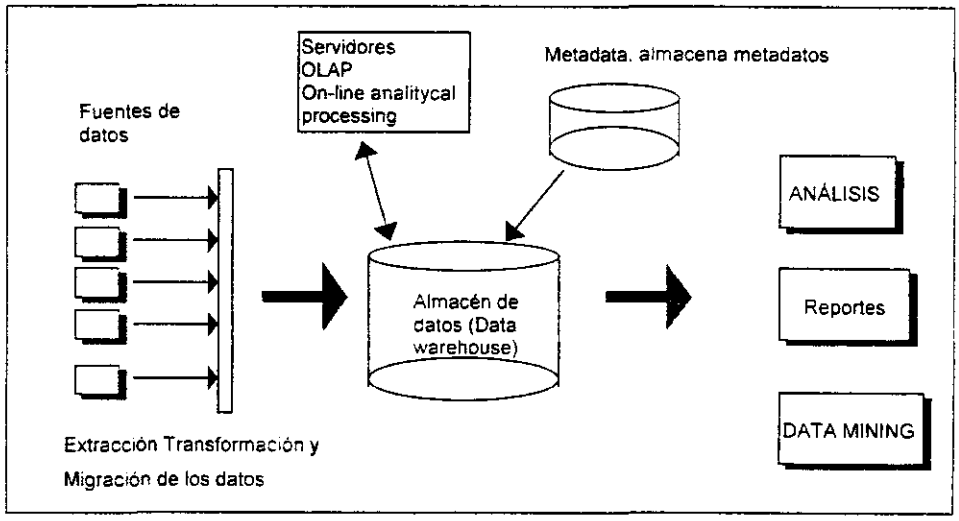

Fig. 1.6 Arquitectura Típica de un Almacén de datos[14]

[13] Harjinder S.Gill Et. al. Data Warehousing la Integración de información para la mejor toma de decisiones. Cap.1
[14] Ibid.

1.8 LA MINERÍA DE DATOS Y EL PROCESO DE DESCUBRIMIENTO DE CONOCIMIENTO EN LA BASES DE DATOS (KDD)

1.8.1 INTRODUCCIÓN

Uno de los grandes problemas históricos de los 90's ha sido el enorme flujo de información llamado también "glut" (superabundancia) al que se enfrentan las organizaciones día con día. con la Web y el empuje de las nuevas tecnologías siempre incrementando el cúmulo de datos en el escritorio, ha llegado a convertirse en uno de los principales retos a solucionar. El problema es especialmente agudo para quienes necesitan estar actualizados usando siempre información encadenada como base para toma de decisiones o innovaciones. A todo esto se agrega la rapidez de los cambios que constantemente desafían el status del conocimiento adquirido.

Existe una urgente necesidad de una nueva generación de Técnicas Computacionales y herramientas para asistir a los humanos en la extracción de la información (y de conocimiento) de un acelerado crecimiento del volumen de los datos. Estas técnicas y herramientas son el objeto del surgimiento del proceso de *Descubrimiento de Conocimiento en los Datos KDD (Knowledge Discovery in Databases)*. Actualmente se trabaja en la unificación del marco conceptual lo que permitirá entender la variedad de actividades en este multidisciplinario campo. Algunos autores ven el proceso de KDD como un conjunto de varias actividades que se realizan para dar un sentido a los datos. En el núcleo de este proceso esta la aplicación de métodos de *Minería de Datos* para descubrimiento de patrones, entendiendo como patrón un término usado para designar a los patrones o modelos extraídos de los datos, y el conjunto de datos visto como un objeto susceptible de contener conocimiento.

Históricamente la noción de encontrar patrones útiles en estos objetos, se ha presentado dando una variedad de nombres incluyendo minería de datos, extracción de conocimiento, descubrimiento de información y procesamiento de patrones en los datos. De acuerdo con el Gartner Group, muchas empresas invierten hasta 1.5 millones de dólares en consultoría para la obtención y administración del conocimiento, y esperan que dicha cifra se incremente a 5 millones de dólares para el 2001 y un mercado que crece a esa velocidad nos habla de la relevancia de estos nuevos conceptos.

2. DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS Y MINERÍA DE DATOS

2. DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS Y MINERÍA DE DATOS

2.1 FUNDAMENTOS FILOSÓFICOS

2.1.1. EL RAZONAMIENTO

En sentido general, el razonamiento es cualquier procedimiento de prueba, cualquier argumento, conclusión, inferencia, deducción, inducción, analogía, etc. En sentido estricto, el razonamiento es la sucesión lógica de juicios que desemboca en una conclusión. La forma más perfecta y compleja de razonamiento es la conceptual. Se distinguen diferentes tipos de razonamiento, de acuerdo con su forma y grado de rigor:

1) El deductivo: que consiste en derivar un caso particular de un principio general: todo hombre es mortal, por consiguiente Santiago (tal individuo) es mortal. Este tipo de razonamiento, puramente lógico, es riguroso, pero bastante estéril, consideran algunos.
2) El inductivo: que consiste en extraer de un caso particular (veo una mujer astuta) una ley general (todas las mujeres son astutas). Este razonamiento es bastante creativo, pero es muy poco riguroso. Tal tipo de razonamiento se practica en la ciencia física, con el mayor rigor posible, para pasar de la observación de los hechos (según las características consideradas) a la definición de una "hipótesis".

2.1.1.1 RAZONAMIENTO DEDUCTIVO

Se denomina "deductivo" al razonamiento que pasa de lo universal a lo menos universal, a lo particular, o, en el caso límite, de lo universal a lo igualmente universal. Ante todo, la deducción es un medio de demostración. Se parte de premisas que se suponen seguras, en las que basan su certeza las consecuencias deducidas. Aristóteles declara que es necesario que la ciencia demostrativa, la única verdadera a sus ojos, parta de premisas verdaderas, primarias, inmediatas y más conocidas que las conclusiones anteriores de las que son las causas. Las matemáticas han representado hasta una época muy reciente un modelo de este tipo de ciencia.

El vocablo deducción significa conclusión que resulta de una o más premisas. En la historia hay tres interpretaciones: la fundada en la esencia necesaria o sustancia; la fundada en la evidencia sensible la que le niega fundamento y la considera confiada a reglas.

Pero si la deducción demuestra, raras veces descubre. Para ello hay que alimentar la reflexión del sujeto por medio de la observación del objeto[15]. Esto afirman los empiristas, aunque sea distinto el proceso que propugnan. Kant llamó deducción a una demostración que, en oposición a la prueba por hechos, evidencia una exigencia de derecho. La deducción es trascendental (u objetiva) si explica cómo un concepto puro a priori se relaciona con los objetos, y empírica (o subjetiva) si muestra cómo se adquiere un concepto mediante la experiencia y la reflexión.

Muchas son las versiones que se han dado sobre la deducción; entre ellas tenemos: es un razonamiento de tipo mediato; es un proceso discursivo y descendente que pasa de lo general a lo particular; es un proceso discursivo que pasa de una proposición a otras proposiciones hasta llegar a una proposición que se considera la conclusión del proceso; es la derivación de lo concreto a partir de lo abstracto; es la operación inversa a la inducción; es un razonamiento equivalente al silogismo y, por lo tanto, una operación estrictamente distinta de la inductiva; es una operación discursiva donde se procede necesariamente de unas proposiciones a otras.

Ciertamente que cada una de las nociones anteriores adolece de varios inconvenientes, pero a la vez apunta a una o varias características iluminadoras de la deducción. La deducción es un razonamiento riguroso que consiste en aplicar un principio general a un caso particular: El silogismo constituye un ejemplo excelente, pues es la argumentación en que se comparan dos extremos con un tercero, para descubrir la relación que tienen entre sí. Toda virtud es laudable; así la prudencia es virtud; luego la prudencia es laudable. Los dos extremos, "prudencia" y "laudable", se comparan con el medio (el tercero), "virtud"; y de aquí se deduce que el atributo, laudable, conviene a la prudencia. Los extremos comparados se llaman términos: mayor, el más general; y menor, el otro. El punto de comparación se denomina término medio. En el ejemplo, prudencia es el menor, laudable el mayor, virtud el medio. La premisa en que se halla el término mayor, se llama mayor, y la otra menor: Es más frecuente que la mayor sea la primera del silogismo; pero aunque muden de lugar no varía su naturaleza.

La deducción se emplea en todas las ciencias (matemática, física, biología, ciencias sociales), pero es particularmente apropiada en las ciencias más formalizadas, tales como la lógica, la matemática y la física teórica. Los procedimientos deductivos parten de la ley general para llegar al caso particular.

[15] Cfr. Zubiri Xavier, Notas sobre la inteligencia humana. pp.341-353

2.1.1.2 RAZONAMIENTO INDUCTIVO

La inducción es una generalización, operación por la cual se hace extensivo a una clase de objetos lo que se ha observado en un individuo o en algunos casos particulares. La filosofía clásica distingue cierta inducción rigurosa, llamada aristotélica, que reconoce ciertas características a los fenómenos observados (en principio, la totalidad de los casos), generalizándolos o resumiéndolos en una ley, y la inducción amplificadora (erróneamente denominada baconiana) o experimental, que, partiendo de un número determinado de hechos observados, generaliza aplicándolos a un número infinito de hechos posibles. Quien proporcionó un concepto suficientemente preciso de la inducción y lo introdujo como vocablo técnico para designar un determinado proceso de razonamiento, fue Aristóteles, aunque no le dio un tratamiento único. Mientras la deducción concluye lo particular de lo universal, o de la esencia de un objeto sus propiedades necesarias, la inducción intenta obtener (de los casos particulares observados) una ley general válida también para los no observados.

La inducción es cierto razonamiento que nos hace pasar de lo particular a lo general. Por ejemplo, una persona que ve a un buen estudiante de determinado centro educativo, "induce" que todos los estudiantes de ese centro son buenos estudiantes. Al dominio de la inducción pertenecen las leyes de las ciencias naturales y de las ciencias particulares en general. Sexto Empírico partió de la distinción entre inducción incompleta e inducción completa. La inducción llamada "completa" consiste en la observación de todos los casos particulares, no es un raciocinio, sino una enumeración. En cambio, la inducción matemática, es decir, la conclusión de que una cierta fórmula, válida para n, vale asimismo para n + 1, se demuestra partiendo de la índole de la fórmula con el mismo rigor deductivo con que se prueba que vale para un número determinado; por tanto, en realidad es una deducción. La inducción rigurosa no permite pasar de los hechos a las leyes.

La novedad que introduce la experiencia no afecta a la lógica normal, pues las ciencias se desarrollarán con independencia de la misma. Pero sus métodos, al no fundamentarse en una lógica rigurosa, se separan de la lógica y de la filosofía. La verdadera inducción es la inducción incompleta, que, de un número relativamente corto de casos observados saca una conclusión respecto a todos los casos semejantes. Este raciocinio encuentra su justificación en el principio de razón suficiente, el cual, excluyendo la casual semejanza de los casos sometidos a observación metódica, exige, en las condiciones observadas, cierta necesidad por parte del proceso estudiado.

Ahora bien, si éste es necesario en las condiciones dadas, se verificará siempre que se den condiciones semejantes. La inducción engendra auténtica certeza, aunque, evidentemente, no es absoluta (certeza hipotética). Francis Bacon (1561-1625) planteó con insistencia la cuestión del tipo de enumeración que debía considerarse como propio del proceso inductivo científico. Afirmó que "el hombre, ministro e intérprete de la naturaleza, hace y entiende en la medida en que haya observado el orden de la naturaleza, mediante la observación de la cosa o con la actividad de la mente" (Novum Organon). Entiende por método no sólo el retorno a la experiencia sino también el progreso positivo del pensamiento y las precauciones para poseerlo. "La verdad, escribe, sobrevive más fácilmente al error que a la confusión". Afirma que la investigación no puede partir de la percepción de lo particular sino más bien "de las generalidades confusas del sentido común" para observar los casos particulares y volver en lo posible a otra generalidad, pero racional y ordenada. "Lejos de oponer inducción y deducción, que juegan un papel esencial en el método experimental", declara que "toda filosofía natural sólida y fructuosa emplea una doble escalera, a saber, la escalera ascendente y la escalera descendente; una que sube de la experiencia a los axiomas (principios o hipótesis), otra que desciende de los axiomas a los nuevos inventos" (Novum Organum).

Observando que en las ciencias se llega a la formulación de proposiciones de carácter universal luego de partir de enumeraciones incompletas, formuló, en sus tablas de presencia y ausencia, una serie de condiciones que permiten establecer inducciones legítimas. Denominó tablas a las coordinaciones de las instancias, esto es, a los conceptos particulares de un fenómeno y distinguió las tablas:

1) De presencia, casos en que se presenta un fenómeno particular
2) Las tablas de ausencia, casos en que dicho fenómeno no se presenta
3) Las tablas de grados o comparativas, las diferentes maneras en que el fenómeno puede manifestarse en contextos diferentes; y, por último
4) Las tablas exclusivas, las tablas ofrecen el esquema de toda investigación experimental (Novum Organum). Bacon no era un científico, sino un profeta que propagó la idea de una ciencia experimental y previó e inspiró la revolución industrial.

Al respecto se ha alegado que no es justo contraponer la inducción baconiana a la inducción aristotélica, pues Aristóteles y otros autores antiguos y medievales no excluyeron las inducciones basadas en enumeraciones incompletas; lo que hicieron fue distinguir entre enumeraciones completas y enumeraciones incompletas, agregando que si bien ambas son suficientes para producir inducciones legítimas, sólo las primeras exhiben claramente el

mecanismo lógico del proceso inductivo. La inducción no es un razonamiento riguroso (a diferencia de la deducción), pero es el principio de todos los descubrimientos[16]. En el esquema clásico del método científico, expuesto por Juan Stuart Mill (1806-1873) autor de "Principios de Economía Política", la inducción corresponde al segundo momento de la investigación, o sea que sigue a la "observación" y permite el tránsito de ésta a la enunciación de una "ley". El tercer momento de la investigación corresponde a la "verificación" experimental.

El sabio prueba el movimiento del pensamiento avanzando en el conocimiento; pero el filósofo se venga poniendo en entredicho el valor de la ciencia. El conflicto entre el rigor y la fecundidad se amplía y hace nacer el problema del conocimiento y del valor de la ciencia. Aún admitiendo la existencia de un punto débil en el descubrimiento científico, se pregunta Jacques Lefebvre (1455-1536), ¿puede considerarse absoluta la separación entre el rigor de la deducción y la fecundidad de la inducción? Según Aristóteles, el silogismo es un razonamiento en el que el término medio juega un papel esencial de mediación, de relación fundamental. Clasifica los objetos según las cualidades esenciales de la naturaleza. El juicio da un contenido de silogismo. Decir: Sócrates es un hombre, todos los hombres son mortales, luego Sócrates es mortal, es reconocer en Sócrates las cualidades de hombre y mortal.

Según Franz. Kutschera (Teoría de la ciencia, 1972), puede distinguirse tres fases en el proceso de investigación.

1) El punto de partida es la observación de los fenómenos empíricos, describimos y clasificamos esos fenómenos y reunimos material de observación.

2) Sobre la base del material de observación anticipamos unas generalizaciones; reunimos esas observaciones en hipótesis, que explican los fenómenos.

3) Se exponen varias hipótesis, sin que de momento estén vinculadas entre sí. Pero queremos ponerlas en una conexión sistemática. Así que buscamos una hipótesis de tipo superior. que está en condiciones de reunir y explicar las hipótesis de tipo inferior. Y así establecemos una teoría.

Los procedimientos inductivos van de lo particular hacia lo general para descubrir y establecer la ley científica. Además, debemos recordar los sub-procedimientos analíticos y sintéticos:

[16] Cfr. Loc. cit.

Los analíticos:

1) La división o separación de las partes constitutivas de un todo.
2) La clasificación u ordenamiento por clases.

Los sintéticos:

1) La conclusión o consecuencia lógica de una argumentación.
2) El resumen o condensación clara y ordenada de varios conceptos.
3) La definición o delimitación breve, clara y completa de un concepto.
4) La recapitulación o cierre total o parcial de aspectos fundamentales.

Los razonamientos inductivos no son válidos o inválidos en el sentido en que estos términos se aplican a los razonamientos deductivos. Claro está que pueden estimarse como mejores o peores los razonamientos inductivos, según el grado de verosimilitud o probabilidad que sus premisas confieran a sus conclusiones.

2.2. EL APRENDIZAJE

Posiblemente la característica más distintiva de la inteligencia humana es el aprendizaje. Podríamos decir que este proceso se divide a las siguientes fases:

- adquisición de conocimiento
- desarrollo de habilidades a través de instrucción y práctica
- organización de conocimiento
- descubrimiento de hechos

En este contexto, se dice que un sistema que aprende (o aprendiz) es un programa (o un conjunto de algoritmos) que para resolver problemas toma decisiones basadas en la experiencia acumulada, en los casos resueltos anteriormente, para mejorar su actuación. Estos sistemas deben de ser capaces de trabajar con un rango muy amplio de tipos de datos de entrada, que pueden incluir datos incompletos, inciertos, ruido inconsistencias etc. Una primera caracterización del aprendizaje puede ser la siguiente:

Aprendizaje = *selección + adaptación*

Visto así el aprendizaje es un proceso que tiene lugar en dos fases. Una en la que el sistema elige (selecciona) las características más relevantes de un objeto (o un evento), las compara con otras conocidas si existen, a través de algún proceso de cotejamiento (pattern matching) y

cuando las diferencias son significativas, adapta su modelo de aquel objeto (o evento) según el resultado del cotejamiento. La importancia del aprendizaje reside en que sus resultados, habitualmente se traducen en mejoras en la calidad de actuación del sistema.

2.2.1 APRENDIZAJE AUTOMÁTICO (MACHINE LEARNING)

El aprendizaje automático, también llamado aprendizaje artificial o aprendizaje con máquinas es un área de interés muy desarrollada en la Inteligencia Artificial. Aprendizaje es un término muy general que denota la forma, o formas, en la cual un ser vivo (o una máquina) aumenta su conocimiento y mejora sus capacidades de actuación (performance) en su entorno. De esta manera el proceso de aprendizaje puede ser visto como un generador de cambios en el sistema que aprende, que por otra parte ocurren lentamente, adaptativamente y que pueden ser revocados o ampliados.

Estos cambios se refieren no sólo a la mejora de las capacidades y habilidades para realizar tareas sino que también implican modificaciones en la representación de hechos conocidos. En este contexto, se dice que un sistema que aprende (o aprendiz) es un sistema que para resolver problemas toma decisiones basadas en la experiencia acumulada, en los casos resueltos anteriormente, para mejorar su actuación. Estos sistemas deben ser capaces de trabajar en un rango muy amplio de tipos de datos de entrada, que pueden incluir datos incompletos, inciertos, ruido ó inconsistencias.

Un agente autónomo tiene la capacidad de aprender cuando de forma autónoma es capaz de realizar nuevas tareas, de adaptarse a los cambios de su entorno, o mejorar su actuación en tareas ya conocidas.

2.2.1.1 PARADIGMAS DEL APRENDIZAJE AUTOMÁTICO

La pregunta no es saber si el aprendizaje automático (o artificial) es posible o no, sino cuáles son los métodos que efectivamente pueden conducir al aprendizaje. Según el tipo de selección y adaptación (transformación) que un sistema realiza sobre la información disponible es posible identificar varios paradigmas del aprendizaje automático[17].

[17] Cfr.Cortés García, Ulises. Et. al. Inteligencia Artificial. Cap.5. Passim.

❖ **Aprendizaje Deductivo**

Este tipo de aprendizaje se realiza mediante una secuencia de inferencias deductivas usando hechos, o reglas conocidas. A partir de los hechos conocidos nuevos hechos o nuevas relaciones son lógicamente derivadas. En este tipo de sistemas la monotonía de la teoría definida por la base de conocimientos es importante.

❖ **Aprendizaje Analítico**

Los métodos usados en este tipo de aprendizaje intentan formular generalizaciones después de analizar algunas instancias en términos del conocimiento del sistema. En contraste con las técnicas empíricas de aprendizaje, que normalmente son métodos basados en las similitudes, el aprendizaje analítico requiere que se le proporcione al sistema un amplio conocimiento del dominio. Este conocimiento es usado para guiar las cadenas deductivas que se utilizan para resolver nuevos problemas. Por lo tanto, estos métodos se centran en mejorar la eficiencia del sistema y no en obtener nuevas descripciones de conceptos como el aprendizaje inductivo.

❖ **Aprendizaje Analógico**

Este tipo de aprendizaje intenta emular algunas de las capacidades humanas más sorprendentes: poder entender una situación por su parecido con situaciones anteriores conocidas, poder crear y entender metáforas o poder resolver un problema notando su posible semejanza con problemas vistos anteriormente y adaptando de una forma conveniente la solución que se encontró para esos problemas. Este tipo de sistemas requieren de una gran cantidad de conocimiento.

❖ **Aprendizaje Inductivo**

Es el paradigma más estudiado dentro del aprendizaje automático. Normalmente estos sistemas carecen de una teoría del dominio, es decir, no conocen a priori los objetos con los que tratan o su cantidad. Trata problemas como los de inducir la descripción de un concepto a partir de una serie de ejemplos y contraejemplos del mismo o determinar una descripción jerárquica o clasificación de un grupo de objetos.

❖ **Aprendizaje mediante Descubrimiento**

El tipo de Descubrimiento es una forma restringida de aprendizaje en el cual un agente adquiere conocimientos sin la ayuda de un "profesor". Este proceso ocurre cuando no existe ninguna fuente disponible que posea el conocimiento que el agente busca. Un tipo particular de descubrimiento se lleva a cabo cuando un agente intenta agrupar objetos que supone del mismo conjunto.

❖ **Algoritmos Genéticos y Conexionismo**

Los algoritmos genéticos están inspirados en las mutaciones y otros cambios que ocurren en los organismos durante la reproducción biológica de una generación a la siguiente y en el proceso de selección natural de Darwin. Los problemas principales que trata de resolver son el descubrimiento de reglas y la dificultad mayor con que se encuentra es la asignación de crédito a las mismas. Este último punto consiste en valorar positiva o negativamente la regla según lo útiles que sean al sistema. Esta valoración será la que determine qué regla aplicar para resolver un problema determinado.

Otra manera de concebir un sistema de aprendizaje automático es el denominado *enfoque conexionista*. En esta aproximación el sistema es una red de nodos interconectados que tiene asociada una regla de propagación de valores y cuyos arcos están etiquetados con pesos. Ante un conjunto de ejemplos el sistema reacciona modificando los pesos de los arcos. Se dice que el sistema aprende si adapta los pesos de las conexiones de tal manera que le lleven a dar la salida correcta ante todas (o la mayoría) de las entradas que se le ofrezcan. Un ejemplo de este tipo de enfoque son las redes neuronales artificiales.

2.2.1.2 EL APRENDIZAJE INDUCTIVO

El aprendizaje inductivo puede verse como el proceso de aprender una función. Por ejemplo, en aprendizaje supervisado, al elemento de aprendizaje se le da un valor correcto (o aproximadamente correcto) de una función a aprender para entradas particulares y cambia la representación de la función que está infiriendo, para tratar de aparear la información dada por la retroalimentación que ofrecen los ejemplos.

Un ejemplo es un par (x,f(x)), donde x es la entrada y f(x)la salida. El proceso de inferencia inductiva pura (o inducción) es: dada una colección de ejemplos de f, regresar una función h tal que se aproxime a f. A la función h se le llama la hipótesis.

En principio existen muchas posibilidades para escoger h, cualquier preferencia se llama bias o sesgo. Todos los algoritmos de aprendizaje exhiben algún tipo de sesgo. La selección de una representación para la función deseada es probablemente el factor más importante en el diseño de un sistema de aprendizaje. Desde un punto de vista más tradicional (hablando de representaciones simbólicas/reglas...), podemos decir que una buena parte del aprendizaje automático o machine learning está dedicada a inferir reglas a partir de ejemplos. Descripciones generales de clases de objetos, obtenidas a partir de un conjunto de ejemplos, pueden ser usadas para clasificar o predecir.

En general, el interés no está en aprender conceptos de la forma en que lo hacen los humanos, sino aprender representaciones simbólicas de ellos. Algunos elementos que deben de especificarse para caracterizar un problema de inferencia inductiva son:

- **El espacio de hipótesis:**

El espacio de hipótesis es el conjunto de descripciones tal que cada regla en la clase tiene por lo menos una descripción en el espacio de hipótesis. Diferentes espacios de hipótesis pueden usarse para la misma clase de reglas. El lenguaje de hipótesis debe de tener descripciones para todas las reglas en la clase, pero puede contener más. El lenguaje de la hipótesis depende del área de aplicación. Una vez definido, una buena parte del tiempo de desarrollo se dedica a seleccionar cuidadosamente las estructuras de conocimiento adecuadas para la tarea de aprendizaje.

El proceso de inducción puede verse como una búsqueda de hipótesis o reglas. El espacio puede buscarse sistemáticamente, hasta encontrar la regla adecuada. Dado un espacio de hipótesis particular, podemos tener una enumeración de descripciones, digamos, tal que cada regla en el espacio de hipótesis tiene una o más descripciones en esta enumeración.

- **Lenguaje de Hipótesis**

El lenguaje de hipótesis determina el espacio de hipótesis del cual el método de inferencia selecciona sus reglas. El lenguaje impone ciertas restricciones (o preferencias) en lo que puede ser aprendido y qué estrategias de razonamiento son permitidas.

Al escoger un lenguaje, debemos de considerar no sólo lo que queremos que el sistema realice, sino también qué información se le debe de proporcionar al sistema de entrada para permitirle resolver el problema, y si lo va a resolver a tiempo.

Al igual que en razonamiento, existe un balance fundamental entre la expresividad y la eficiencia. El lenguaje de hipótesis depende del área de aplicación. Una vez definido, una buena parte del tiempo de desarrollo se dedica a seleccionar cuidadosamente las estructuras de conocimiento adecuadas para la tarea de aprendizaje.

- **Conjunto de ejemplos y su presentación:**

Existen diferentes tipos de presentación de datos y sus efectos en la inferencia de lenguajes. Una presentación puede consistir en: (i) sólo ejemplos positivos y (ii) positivos y negativos. Casi todos los algoritmos requieren presentaciones admisibles, esto es, para cada regla falsa que es consistente con los ejemplos positivos, existe un ejemplo negativo que la refuta (se acerca a Popper, las teorías deben de ser refutables con hechos). Los ejemplos se usan para probar y formar hipótesis. En la práctica una selección de ejemplos se hace sobre el espacio de ejemplos. Esta selección puede ser dada por el medio ambiente, seleccionada en forma aleatoria, propuesta por el sistema. Una ``buena" selección de ejemplos puede mejorar el desempeño de un sistema. A veces esa selección puede mejorarse con conocimiento del dominio.

- **Métodos de inferencia:**

Intuitivamente un método de inferencia es un proceso computacional de algún tipo que lee ejemplos y produce hipótesis del espacio de hipótesis.

2.3 DESCUBRIMIENTO DE CONOCIMIENTO EN LOS DATOS (KDD)

2.3.1 INTRODUCCIÓN

El término KDD*(Knowledge Discovery in Databases)* es con el que se denomina al *DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS* fue creado en el primer congreso de KDD realizado en 1989, y dicho concepto se acuñó para enfatizar que el **conocimiento** es el producto final de una exploración conducida por los datos.

El KDD se ha desarrollado desde la intersección de campos de investigación aprendizaje automático (machine learning), reconocimiento de patrones, bases de datos, Inteligencia Artificial, adquisición del conocimiento para Sistemas Expertos y visualización de datos entre otros. La meta de todos estos campos es común y es la extracción de conocimiento en el contexto de un conjunto considerablemente grande de datos. El KDD se traslapa con el aprendizaje automático y el reconocimiento de patrones en el estudio de teorías y algoritmos particulares de minería de datos. **Uno de los objetivos del KDD es encontrar patrones comprensibles que puedan ser interpretados como útiles o como conocimiento interesante** y pone especial énfasis en trabajos con grandes conjuntos de datos del mundo real. El KDD también tiene mucho en común con estadística particularmente con métodos exploratorios o preliminares de análisis de datos. Los sistemas de software de KDD frecuentemente incrustan procedimientos particulares de estadísticas para representar y modelar datos, evaluando hipótesis y manejando "ruido" dentro de un marco general de descubrimiento de conocimiento. En contraste con las tradicionales aproximaciones en estadística, KDD emplea aproximación típica, más investigación del modelo, extracción y operación de un conjunto de objetos empleando útiles estructuras de datos.

Un área que comúnmente se relaciona con el KDD debido a la mercadotecnia que se ha aplicado, es el Data Warehousing o almacenaje de datos. El almacén de datos (data warehouse) es un sistema de administración de base de datos relacional (RDMS Relational Database Management System) diseñado específicamente para satisfacer las necesidades de transacción de procesamiento de sistemas que se refiere a la colección y depuración (limpieza) de datos para ponerlos disponibles para su análisis On-Line y para el soporte de decisiones. Un concepto conocido para el análisis del data warehouse ha sido llamado OLAP **(on-line analytical processing).** La función de las herramientas OLAP es proporcionar análisis de datos multidimensional, así como resumir información, a lo largo de muchas dimensiones o niveles de los datos. Dichas herramientas están destinadas a la simplificación y soporte interactivo del análisis de datos, mientras que **la meta de las herramientas de KDD es automatizar el proceso tanto como sea posible.**

2.3.2 DEFINICIÓN

"El Descubrimiento de Conocimiento en Bases de Datos (KDD) es el proceso no trivial de validación, e identificación, de conocimiento nuevo y potencialmente útil contenido en los patrones de los datos"[18].

[18] Fayyad,Usama. Et.al. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". Passim

Aquí dato es un conjunto de hechos llamado también objeto por la complejidad de su estructura y patrón es una expresión en algún lenguaje que describe un conjunto de datos o un modelo aplicable a aquel subconjunto.

El término proceso implica que KDD está comprendido de una serie de pasos que involucran preparación de datos, investigación de patrones evaluación del conocimiento y refinamiento de todos estos pasos repetidos en múltiples iteraciones. El hecho de ser no-trivial se refiere a que implica reconocimientos o inferencias lo que complica dicho proceso. El patrón descubierto puede ser aplicable o válido para un nuevo dato con algunos grados de certidumbre, este patrón debe ser comprensible, sí bien no inmediatamente, sí después de algunos post-procesamientos.

El proceso total del KDD incluye la evaluación y posible interpretación de los propios patrones para de esta forma determinar qué patrones pueden ser considerados como *conocimiento*. El KDD es un proceso interactivo e iterativo y abarca numerosos pasos con muchas decisiones lo que enfatiza la iteratividad natural del proceso.

Aquí se da un perfil general de los pasos básicos:

1) Desarrollar un dominio de la aplicación comprensible. establecer la prioridad relevante del conocimiento e identificar las metas del KDD desde el punto de vista del usuario.

2) Crear un conjunto de datos destino, enfocándose sobre un conjunto de variables o datos mostrados y sobre los cuales el descubrimiento pueda ser desarrollado.

3) Limpieza de datos y procesamiento: operaciones básicas tales como remover el "ruido" (información no útil) de los datos si así se requiere. reunir la información necesaria para el modelo y decidir sobre estrategias para manejo de campos perdidos en los objetos.

4) Reducción de datos y proyección: encontrar caracteristicas útiles para representar los datos dependiendo de la meta. Usar reducción de dimensionalidad o métodos de transformación para reducir un número efectivo de variables bajo consideración o para hallar representaciones invariables de los datos.

5) Establecer una correspondencia entre las metas del KDD (paso 1) y un método particular de Minería de datos (clasificación, regresión, agrupamiento, etc.).

6) Seleccionar los algoritmos de Minería de datos: elegir datos para usarse en el reconocimiento de patrones dentro de los datos. Esto incluye decidir que modelos y parámetros pueden ser apropiados (Por ejemplo los modelos para datos categóricos son diferentes a los modelos sobre vectores reales) y asociando un método particular con un criterio general o de conjunto particular del KDD.

7) **Minería de datos (Data mining):** busca patrones de interés en una forma de representación particular o en un conjunto de representaciones semejante tales como reglas de clasificación, árboles, regresión, agrupamiento, etc.

8) Interpretar los patrones extraídos. Este paso puede implicar visualización de los patrones o modelos extraídos o bien visualización de los datos a partir de los modelos obtenidos.

9) Consolidar el Conocimiento Descubierto: incorporando este conocimiento dentro de sistemas para nuevas acciones o simplemente documentándolo y reportando hacia grupos de interés. Esto también incluye revisión y resolución potencial de conflictos con conocimiento validado previamente.

La mayoría del trabajo inicial en KDD está enfocado sobre el paso 7 la *minería de datos*, sin embargo, los otros pasos son igual de importantes para la aplicación exitosa de este proceso KDD en la práctica.

El proceso KDD puede involucrar iteraciones significativas y puede contener ciclos entre dos pasos cualesquiera. El flujo básico de los pasos, aunque no la multitud potencial de iteraciones y ciclos, es ilustrada en la figura 2.1:

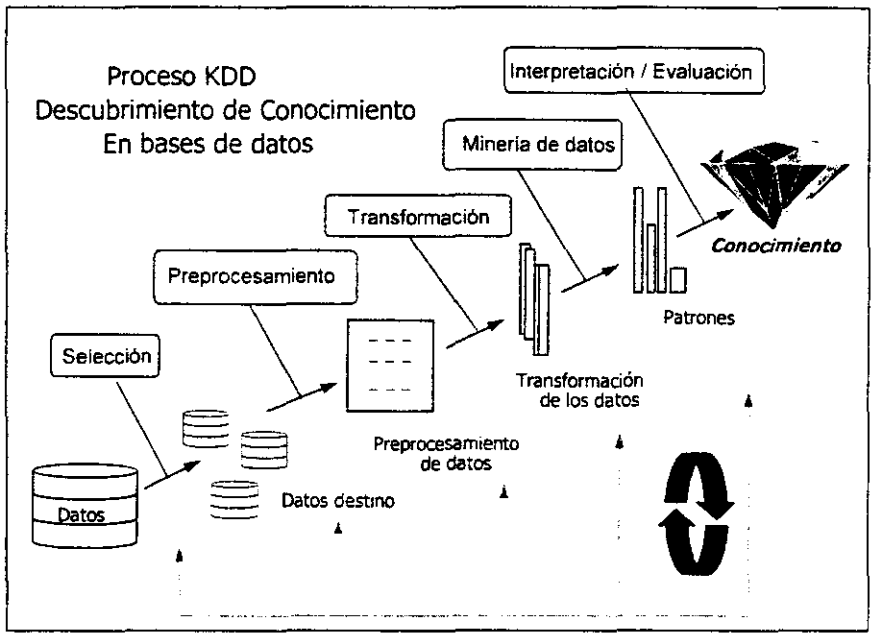

FIG. 2.1 El proceso de Descubrimiento en Bases de Datos (KDD).[19]

2.4 MINERÍA DE DATOS

2.4.1 MARCO CONTEXTUAL

Un conjunto de métodos matemáticos y técnicas de software usadas para encontrar patrones y regularidades en un conjunto de datos es comúnmente referido como *minería de datos*. Históricamente, el desarrollo de estadística ha llevado al descubrimiento de métodos para el análisis de datos, estos métodos estadísticos han sido aplicados con el objetivo de encontrar correlaciones y dependencias en conjuntos de datos. Entonces ¿qué es lo nuevo? Tres factores han cambiado recientemente el panorama del análisis de datos.

[19] Ibid

Primero, la disponibilidad del *poder computacional* a bajo costo. Cuando la mayoría de los métodos estadísticos ahora en uso fueron desarrollados, en los 60's y 70's, la capacidad de las mainframes era equivalente a la de una PC actual, a esto podemos aunar la *disminución en el costo de almacenamiento* de los datos.

Un segundo factor es la *acumulación de datos en proporciones exorbitantes*, este fenómeno es consecuencia de los bajos costos de almacenamiento combinado con el alto nivel de automatización en los procesos y la introducción de dispositivos electrónicos para almacenamiento o reunión de los datos, tales como puntos de venta o accesos remotos a diversas fuentes de información. El tercer factor es la *introducción de un nuevo conjunto de métodos lógicos desarrollados en el campo de la inteligencia artificial.* Se han tratado de desarrollar modelos generales de cognición y en particular de *aprendizaje*.

En particular las técnicas de representación del conocimiento y las de aprendizaje hacen posibles importantes aplicaciones para la representación y el análisis de los datos[20]. Es importante mencionar que el proceso KDD así como la minería de datos (data mining) son conceptos que han cobrado importancia con el surgimiento de la aplicación cliente-servidor Data Warehousing, ya que esta idea está a menudo más asociada con costo eficaz y flexibilidad y por consiguiente algo más práctico y comercial. Este contexto del desarrollo de la minería de datos, ayuda a explicar el gran interés que esto ha despertado actualmente.

2.4.2 DEFINICIÓN

La minería de datos es parte del proceso de descubrimiento de conocimiento en los datos (KDD) y consiste en aplicar *análisis de datos y algoritmos de descubrimiento de conocimiento* para producir un número de patrones particulares sobre los datos llamados *conocimiento*.

A continuación se presentan de manera breve las metas primarias de la Minería de datos, una descripción de estas metas y una visión somera de los algoritmos de minería de datos (data mining) que incorporan dichos métodos. Las metas del descubrimiento del conocimiento son definidas por la intención de uso del sistema. Se pueden distinguir dos tipos de metas:

A. **Verificación:** donde se limita a verificar las hipótesis del usuario.
B. **Descubrimiento**: donde el sistema autónomamente encuentra nuevos patrones

[20] Decker, Karsten. Et. al. "Technology Overview: A Report on Data Mining". p.1.

Adicionalmente la meta de Descubrimiento se subdivide en dos:

B1) Predicción: Donde el sistema encuentra patrones con el propósito de predecir el comportamiento futuro de algunas variables.

B2) Descripción: Donde el sistema encuentra patrones que se pretenden presentar al usuario en una forma humanamente entendible.

Si bien los límites entre predicción y descripción no están definidos (ya que algunos de los modelos predictivos pueden ser descriptivos hasta el grado en que estos sean comprensibles) la distinción es útil para comprender de manera general la meta del descubrimiento. La importancia relativa de predicción y descripción para aplicaciones particulares de Minería de datos puede variar considerablemente, sin embargo en el concepto de KDD la descripción tiende a ser más importante que la predicción. Esto contrasta con aplicaciones de reconocimiento de patrones y aprendizaje automático donde la predicción es frecuentemente la meta primaria. Las metas de predicción y descripción son alcanzadas a través de los principales métodos de minería de datos.

2.4.3 LAS TÉCNICAS DE MINERÍA DE DATOS

El descubrimiento de conocimientos tiene sus raíces en la Inteligencia Artificial y el Aprendizaje automático. Existe una amplia gama de técnicas[21] que incluyen *árboles de decisión, reglas, y clasificación, métodos basados en modelos con dependencia probabilística (incluyendo redes bayesianas) y modelos relacionados con el aprendizaje.* Cabe destacar que existe una estrecha relación entre *aprendizaje automático* (machine learning) y las metas de *la minería de datos*, ya que la esencia del campo del aprendizaje automático es desarrollar modelos computacionales para adquisición del conocimiento a partir de hechos y conocimiento previamente adquirido. Un punto importante es que cada técnica de minería de datos, adapta mejor cierto tipo de problemas.

De este modo nos damos cuenta de que no hay método universal de minería de datos y seleccionar una técnica específica para una aplicación particular conlleva algo de arte. A continuación se definen algunos conceptos relacionados con la forma de hacer minería de datos.

[21] Cfr. Berry, Michael. Op.cit. Cap.7. Passim.

❖ ANÁLISIS ESTADÍSTICO

Los sistemas de análisis estadístico, se usan para detectar patrones no usuales en los datos, dichos patrones de datos se explican después mediante modelos estadísticos y matemáticos. Algunas de las técnicas que se emplean son el análisis lineal y no lineal, el análisis de regresión, la univariación y la multivariación y el análisis de series históricas, incluyendo autocorrelación y pronósticos, las herramientas de análisis estadístico existen desde hace tiempo por lo que son las más desarrolladas que se tienen para minería de datos.

❖ HERRAMIENTAS PARA LA PREPARACIÓN DE LOS DATOS

Existen algunas herramientas auxiliares que nos permiten acondicionar los datos antes de aplicar alguna técnica de mineria de datos, con la finalidad de obtener mejores resultados, entre ellas están:

- **Sumarización**

  Este método encuentra una descripción compacta para un subconjunto de objetos. Una cuestión fundamental es ¿Cual es el nivel correcto de detalle en los datos? La respuesta depende del análisis que se realizará normalmente se desea obtener los datos en completo detalle, y en este caso se aplica la sumarización de diferentes formas para tener la presentación que deseamos de los datos, esto no quiere decir que el tener los datos a detalle es necesariamente mejor para los propósitos del análisis, pues existen varias razones para resumir o compactar datos antes de comenzar el mismo.

- **Geometría Fractal**

  Permite comprimir grandes cantidades de datos, y así poder analizarlos en un tiempo razonable. Esto no representa ningún tipo de automatización de razonamiento, simplemente es una ayuda técnica en la manipulación de los datos. Aún así, éstas son herramientas de mucho valor, y complementan la extracción de conocimientos propia de Mineria de datos

2.5 PROCESAMIENTO ANALÍTICO EN LÍNEA (OLAP).

No todo lo que es útil para el análisis de datos es necesariamente mineria de datos, *OLAP* es una forma de presentar datos relacionales de una manera apropiada para que el usuario entienda los datos y los patrones que existen en ellos. Un ejemplo de estos es la visualización ya que no se considera específicamente una técnica de mineria de datos, pero es una importante herramienta usada para extracción y presentación de información.

2.5.1 CARACTERÍSTICAS PRINCIPALES DE OLAP

- Comprende la consulta interactiva y el análisis de datos, por lo regular la interacción es de varias pasadas, lo cual incluye la profundización en niveles cada vez más detallados o el ascenso a niveles superiores de resumen.
- Ofrece opciones de modelado analítico, incluyendo un motor de cálculo para obtener proporciones, desviaciones etcétera, que comprenden mediciones de datos numéricos.
- Recupera y exhibe datos tabulares en dos o tres dimensiones cuadros y gráficas con un pivoteo fácil de los ejes.
- Crea resúmenes y adiciones de los datos

Los métodos *OLAP* están basados en bases de datos multidimensionales las cuales son representaciones de datos que se guardan de manera lógica en arreglos, y que permiten a los usuarios explorar en profundidad dicha base para entenderlos. Algunos la experiencia de algunos autores dice que en la mayoria de las áreas donde existe interés en la minería de datos pueden también obtenerse beneficios desde *OLAP*[22].

2.5.2 EL OLAP Y LA MINERÍA DATOS

En los últimos años las herramientas *OLAP* (on-line analytic processing) han surgido como respuesta al acceso y manejo de grandes bases de datos, ya sea que residan en un simple sistema operacional o en un almacén de datos (data warehouse), siendo éste último concepto con el que más se le ha vinculado. La publicidad generada alrededor del *OLAP* sugiere que es un sustituto de la mineria de datos, pero no es así, las herramientas *OLAP* son poderosas y rápidas para reportar datos a diferencia de la minería cuyo objetivo es encontrar patrones en

[22] Loc.cit.

los datos. *OLAP* y la minería de datos son complementarias; ambas son parte importante en la explotación de los datos, *OLAP* es una herramienta de presentación que puede permitir descubrimiento de conocimiento "manual", este sin embargo depende fundamentalmente de la inteligencia humana, pero ello no se considera formalmente parte de la minería de datos, sino como parte del manejo de los datos en el ambiente empresarial. El *OLAP* es una útil herramienta que puede ser usada de manera exitosa como complemento de la minería de datos.

2.6 LAS FUNCIONES DE LA MINERÍA DE DATOS

La mayoría de los métodos de la minería de datos están basados en *Lógica Inductiva*. A diferencia de la lógica deductiva, en la que todos los resultados (deducciones) son lógicamente correctos, la lógica Inductiva generaliza a partir de la base de datos y extrae información de alto nivel, o conocimiento. Como ya se dijo anteriormente la minería de datos utiliza la técnicas aprendizaje automático (machine learning), arboles de decisión, reconocimiento de patrones, estadística, agrupamiento (clustering), etc. Estos algoritmos se pueden ubicar en diversos grupos de acuerdo la función que realicen[23]. A continuación se explica en que consiste cada una de estas funciones:

2.6.1 CLASIFICACIÓN

La mayoría de las funciones de la minería de datos parecen ser un imperativo humano. En función de entender y comunicar acerca del mundo, nosotros constantemente clasificamos, categorizamos y jerarquizamos.

Un ejemplo de ello es la división que hacemos de los seres vivos en phyla, especie y genero, la división de la materia en elementos, de la gente en razas etc. La clasificación consiste en examinar las caracteristicas de un objeto nuevo (desconocido) y ubicarlo en una de las clases predefinidas. Para nuestros propósitos, los objetos a clasificar son generalmente representados por registros en una base de datos y la acción de clasificar consiste en actualizar cada registro colocando un campo con código que represente algún tipo de clasificación para dicho objeto. La tarea de clasificar se caracteriza por una adecuada definición de las clases en un conjunto de entrenamiento, el cual consiste en ejemplos preclasificados.

[23] Cfr. Berry, Michael . Data Mining Techniques For marketing, sales, and customer support. Cap.4. Passim

El objetivo es construir un modelo que pueda ser aplicado a un dato no clasificado y pueda darle una categorización. Algunos ejemplos de clasificación se dan en las siguientes áreas:

- Clasificación de créditos calificándolos en niveles bajo, medio y altamente riesgoso.
- Determinar que números telefónicos corresponden a máquinas de fax.
- Descubrir reclamos fraudulentos de seguros.

En todos estos ejemplos existe un número limitado de clases y la expectativa es poder asignar cualquier registro a una clase. *Los árboles de decisión y el razonamiento basado en ejemplos*, son técnicas que se adaptan muy bien a la tarea de clasificación.

2.6.2 ESTIMACIÓN

La clasificación nos da una salida discreta: si o no; aceptado o rechazado. La estimación trabaja con salidas de valores continuos. Dado algún dato de entrada, usamos la estimación para obtener un valor de salida para alguna variable desconocida como ingresos, talla o balance de tarjeta de crédito. En la práctica *la estimación frecuentemente se usa para desarrollar una tarea de clasificación.*

Un banco deseaba vender espacios de publicidad en los sobres de los estados de cuenta para un fabricante de patines, puede entonces construir un modelo de clasificación que ponga todos los archivos de clientes en dos clases. patinadores y no patinadores. Otra opción es asignar a cada cliente una puntuación que dependerá de su tendencia a patinar, esto puede ser evaluado del 0 al 1 indicando la estimación de probabilidad de que el cliente sea un patinador. Ahora la clasificación desciende en su ámbito de acción para establecer un umbral para dicha puntuación, por lo que cualquiera con una puntuación más grande o igual que el umbral, esta clasificado como un patinador y los que estén por debajo como no patinadores.

La estimación tiene la gran ventaja de que los registros individuales pueden ahora ser ordenados en categorías. Para resaltar la importancia de esto imaginemos que el fabricante de patines tiene presupuesto para enviar por correo 500,000 piezas. Si la clasificación es empleada y 1.5 millones de patinadores son identificados, entonces puede simplemente colocar el anuncio en el sobre de 500 000 personas seleccionadas al azar de esa lista. Si por otra parte cada cliente tiene una cierta tendencia a ser patinador, entonces podrá enviar la publicidad a los 500 000 más probables candidatos.

Algunos ejemplos de tareas de estimación son:

- Estimación de número de hijos en una familia
- Estimación del costo de un seguro de vida

Las redes neuronales son apropiadas para realizar funciones de estimación.

2.6.3 PREDICCIÓN

La predicción es similar a la clasificación o estimación excepto que los registros son clasificados de acuerdo a la predicción de un comportamiento futuro o un valor futuro estimado. En la predicción la única manera de verificar la exactitud de la clasificación es esperar y ver.

Cualquiera de las técnicas usadas para la clasificación y estimación pueden ser adaptadas para usar predicción usando ejemplos de entrenamiento donde el valor de la variable a predecir, es ya conocido, junto con datos históricos para esos ejemplos. Los datos históricos son usados para construir un modelo que explique el comportamiento actual observado. Cuando este modelo es aplicado para entradas actuales, el resultado es una predicción del comportamiento futuro.

El análisis de canasta de mercado (market basket analysis) usado para descubrir qué artículos son probablemente comprados juntos en una tienda de abarrotes, puede también adaptarse para modelar que compra futura o acción tiende a realizarse con los datos actuales.

El análisis de canasta, el razonamiento basado en modelos, los árboles de decisión, y las redes neuronales artificiales son métodos ideales para la predicción. La elección de la técnica depende de la naturaleza de los datos de entrada, del tipo de valores a pronosticar y de que tan importante sea la explicación adjunta de la predicción.

2.6.4 AFINIDAD DE GRUPOS

La tarea de agrupar por afinidad es determinar que cosas van juntas. El ejemplo prototipo es determinar que cosas van juntas en una lista de compras para el supermercado, de ahí el término análisis de canasta de mercado. Las cadenas de tiendas usan afinidad de grupos para planear los arreglos o acomodo de artículos en los estantes de las tiendas o en el catalogo de artículos, entonces los artículos que se compran frecuentemente juntos deberán de ser vistos juntos.

La afinidad de grupo es una aproximación para generar reglas de los datos. Si dos artículos como comida para gatos y camas para gato, se presentan juntos con bastante frecuencia, nosotros podemos generar dos *reglas de asociación*:

La gente que compra comida de gato también compra camas para gato con probabilidad P1
La gente que compra camas para gato también compra comidas para gato con probabilidad P2

Las reglas de asociación se describen siempre con un mínimo nivel de confianza. El nivel de confianza es una medida de la fuerza de la regla, estas reglas nos sirven para comprender el comportamiento de los clientes en este caso o de un fenómeno dado.

2.6.5 AGRUPAMIENTO O CLUSTERING

El agrupamiento es la tarea de segmentar una población heterogénea en subgrupos más homogéneos. Lo que distingue al agrupamiento de la clasificación es que el agrupamiento o clustering, no depende de clases predefinidas. En la clasificación, la población es subdividida asignando cada elemento o registro a una clase predefinida basada en un modelo desarrollado a través del entrenamiento con ejemplos clasificados previamente.

En el agrupamiento no hay clases predefinidas ni ejemplos. Los registros son agrupados juntos sobre las bases de su propia similitud. Grupos de síntomas pueden indicar diferentes enfermedades, grupos de atributos de hojas y frutas pueden indicar tipos de arboles. Este método suele ser un paso previo para algunas otras formas de minería de datos o modelado. Por ejemplo el agrupamiento puede ser el primer paso en un análisis de mercado. En lugar de intentar solucionarlo abarcando todo con un solo método, y plantando una regla como ¿a qué tipo de promoción los clientes responden mejor? Nosotros podemos primero dividir la base de clientes en grupos de gente con hábitos de compra similares y entonces preguntar que tipo de promoción resulta mejor para cada grupo. Cuando los datos que se van a analizar son relativamente pobres en cuanto a descripción o no se pueden integrar en ningún esquema de clasificación, se emplean las funciones de agrupamiento para descubrir clases en forma automática.

Diferentes algoritmos de agrupamiento generan distintos segmentos o clases en los mismos datos. Existen algunas técnicas para encontrar agrupamientos *incluyendo métodos geométricos, estadísticos y redes neuronales.*

El agrupamiento de datos es una forma muy conveniente de iniciar cualquier análisis sobre datos, grupos similares entre si pueden proporcionar un buen punto de partida.

2.6.6 DESCRIPCIÓN

En ocasiones los propósitos de la mineria de datos son simplemente describir lo que existe sobre una base de datos, pero de tal forma que incremente nuestro conocimiento de la gente, de los productos o de los procesos que producen los datos en primer lugar. Una buena descripción de un comportamiento frecuentemente sugerirá una explicación adecuada para el mismo. Al menos, una buena descripción aconseja dónde comenzar a buscar para hallar una explicación. La política norteamericana es un ejemplo de cómo una simple descripción, (" Las mujeres apoyan a los Demócratas en mayor número que los hombres"), puede provocar gran interés e incluso llegar ser estudiada por periodistas, sociólogos, y analistas políticos y ser considerada más allá de una simple mención de un candidato a un cargo público.

2.7 LA FUNCIÓN DE CLASIFICAR EN LA MINERÍA DE DATOS

2.7.1 INTRODUCCIÓN

En cuanto a los algoritmos de mineria de datos existen dos maneras básicas de desarrollarlos. La primera se refiere al aprendizaje supervisado, la otra es fundamentada en un aprendizaje no supervisado. En el aprendizaje supervisado los patrones son encontrados explorando un número de casos conocidos que muestran la definición de dichos patrones, y basándose en esto se produce una generalización. Con un aprendizaje no supervisado, los patrones en los datos se encuentran, haciendo en primera instancia una caracterización lógica de las regularidades contenidas, ya que inicialmente los patrones no son conocidos en su estructura.

En estos términos la minería de datos puede ser representada como un proceso dual que puede ser tanto sintético, generalizando a partir de casos conocidos, o analítico creando un alto nivel de descripción del modelo de los datos, lo anterior se ilustra en la figura 2.2. . Es importante resaltar que en el contexto de minería de datos, el aprendizaje automático es un conjunto de técnicas que desarrollan dos tareas fundamentales:

1) Generalizan a partir de un conjunto de ejemplos conocidos
2) Detallan una estructura a partir de su descripción

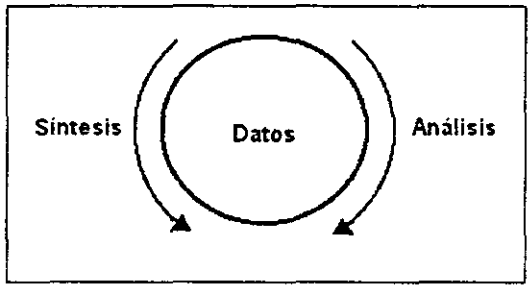

Fig. 2.2 La minería de datos como un proceso dual.

2.7.2 CONCEPTO DE CLASIFICACIÓN

Dentro de un proceso de generación de conocimiento *una de las primeras fases es la categorización o distribución del objeto a conocer en clases* que agrupan a objetos semejantes para su mejor estudio, por lo que la tarea de *clasificación* es de particular interés.

Ahora bien, *la noción básica de aprendizaje supervisado en minería de datos es la de* ***clasificador***[24]. Los clasificadores son sistemas que categorizan datos de entrada en un número de posibles clases. En ocasiones existe confusión alrededor de la definición de un sistema clasificador ya este término e usado en un contexto muy amplio. En este contexto un clasificador es un sistema que recibe alguna entrada y produce como salida una clasificación. En el aprendizaje supervisado los clasificadores son entrenados y probados sobre un conjunto de entrenamiento y uno de prueba o validación respectivamente.

En este sentido el clasificador se convierte en un generalizador, en la medida que generaliza a partir de un conjunto de ejemplos de entrenamiento, a un cierto grupo de datos de entrada externos y desconocidos. La definición correspondiente sería: un generalizador es entrenado para producir un conjunto de valores de clasificación, sobre un determinado número de ejemplos. Cuando generaliza puede ser restringido para usar el valor de clasificación determinado o bien puede producir valores adicionales.

Un sistema basado en reglas como el ID3 de Quinlan*, utiliza un número fijo de valores de generalización*, mientras que una red neuronal en principio puede producir interpolaciones de otros valores no contenidos en los ejemplos de entrenamiento.

[24] Cfr.Decker, Karsten. Et. al. "Technology Overview: A Report on Data Mining". passim.

El problema de la generalización se hace difícil por el enorme número de posibles funciones que existen para adaptar un modelo a partir de los datos. Si el conjunto de ejemplos de entrenamiento incluyera todos los posibles casos, entonces el problema podría reducirse a la construcción de una tabla de búsqueda. Para fines prácticos la minería de datos se basa en dos suposiciones:

1) Las funciones que desea generalizar pueden ser aproximadas a través de algún modelo computacional relativamente sencillo. con un cierto nivel de precisión.

2) El conjunto de datos de entrenamiento contiene información suficiente para desarrollar la generalización.

En otras palabras la minería de datos asume que las regularidades existen y que éstas pueden ser representadas con un procedimiento computacional manejable. Un punto de vista erróneo es pensar que un modelo complejo tiene mejor habilidad de generalización que uno sencillo. Esto no siempre es cierto, ya que existe una relación entre la dimesionalidad de un problema la complejidad del modelo y el número de ejemplos que son necesarios para un aprendizaje apropiado.

Una medida de complejidad de un modelo esta dada por el número de parámetros libres que necesitan especificarse para construir el modelo. La dimensionalidad de un problema es el número de parámetros independientes que son necesarios para describir la dinámica del fenómeno.

Si un modelo complejo es entrenado con un número de ejemplos demasiado pequeño, el normalmente el modelo ajustará perfectamente los ejemplos pero generalizará pobremente. es decir que tendrá poca exactitud en sus resultados. Este hecho es conocido como sobreajuste o sobreadaptación (overfitting).

De lo anterior podemos decir que los problemas complejos de minería de datos son factibles sólo cuando un gran número de datos de entrenamiento ha sido acumulado y esta disponible.

2.7.3 ALGORITMOS PARA CLASIFICACIÓN

En un modelo de clasificación, la conexión entra clases y propiedades puede ser definida en algo tan sencillo como una tabla o tan complejo y no estructurado como un procedimiento manual. Si esta discusión se restringe a modelos ejecutables (aquellos que pueden ser representados como programas de computadora) entonces tenemos dos diferentes formas en éstos pueden ser construidos.

Por un lado el modelo puede ser obtenido entrevistando a las personas que saben como se asignan las categorizaciones para cierto grupo de objetos y de esta manera se llega al modelo que refleje las relaciones que se establecen, a pesar de las conocidas dificultades que esto implica.

Alternativamente numerosas clasificaciones son aproximadas a través de la construcción de un modelo inductivo que se realiza generalizando a partir de un conjunto específico de ejemplos. No todas las tareas de clasificación se prestan para aproximaciones inductivas, por lo que es importante revisar los principales requerimientos de los métodos particulares de inducción.

2.7.3.1 LOS ÁRBOLES DE DECISIÓN

Un árbol de decisión es una posible representación de los procesos de decisión involucrados en tareas inductivas de clasificación. Los atributos son utilizados para crear particiones de conjuntos de ejemplos; los nodos del árbol corresponden a los nombres o identificadores de los atributos, mientras que las ramas de un nodo representan los posibles valores del atributo del nodo. Los hojas son conjuntos ya clasificados de ejemplos. Una de las principales ventajas de los árboles de decisión es que el modelo es completamente explicable ya que toma la forma de reglas explícitas, lo que permite a la gente evaluar los resultados, identificando atributos clave en el proceso.

La reglas por si mismas pueden ser expresadas fácilmente como sentencias lógicas en un lenguaje como SQL y entonces pueden ser aplicadas directamente a nuevos registros. **Un algoritmo básico dentro de los árboles de clasificación es el ID3**. Estos algoritmos se usan donde los elementos de datos tienen atributos o descriptores. El análisis de las tarjetas de crédito es un buen ejemplo de esta aplicación.

CARACTERÍSTICAS PARA USAR ÁRBOLES DE DECISIÓN (ID3)

- **Descripción atributo-valor**

Los datos para ser analizados deben estar contenidos en un archivo. Toda la información acerca de un objeto o caso debe ser expresable en términos de una colección ordenada de propiedades o atributos. Cada atributo puede tener valores tanto discretos como continuos. Pero los atributos usados para describir un caso no deben variar de un caso a otro.

- **Clases predefinidas**

Las categorías a las que estos casos serán asignados deben estar establecidas de antemano. Como ya vimos anteriormente en la terminología de aprendizaje automático esto es llamado aprendizaje supervisado, a diferencia del aprendizaje no supervisado, en el cual los grupos o categorías de los ejemplos son encontrados durante el análisis.

- **Clases discretas**

Este requerimiento se tiene con las categorías a las cuales los casos son asignados. las clases deben estar claramente definidas de tal manera que un caso pertenece o no pertenece a una clase en particular, y deben ser mucho más casos que clases.

Algunas tareas de aprendizaje no son de este tipo. Un grupo que no tiene clases discretas es relacionado con la predicción de valores como el precio del oro, o la temperatura a la que una mezcla se funde. En estos casos los valores continuos se dividen a su vez en categorías, para hacer una discretización.

- **Datos suficientes**

La generalización inductiva procede identificando los patrones en los datos. La aproximación es válida si se tiene patrones robustos, y éstos no pueden distinguirse a partir de coincidencias casuales. Como esta diferenciación generalmente depende de pruebas estadísticas de algún tipo, deben existir suficientes casos para permitir que estas pruebas sean efectivas. La cantidad de datos requerida es afectada por factores como los números de propiedades y clases y la complejidad del modelo de clasificación. A medida que esto incremente se necesitarán más casos de entrenamiento (cientos o incluso miles) para construir un modelo confiable.

- **Clasificación Lógica**

Estos algoritmos construyen solo clasificadores que puedan ser expresados como árboles de decisión o conjuntos de reglas de producción. Esto restringe la descripción de clases a una expresión lógica cuyas premisas o antecedente son declaraciones acerca de valores de atributos particulares. Una forma de común de modelos de clasificación, que no satisface este requerimiento es la discriminación lineal, en donde los pesos de los atributos son combinados aritméticamente y comparados con un umbral, de esta forma la correspondiente descripción de clases es más aritmética que lógica.

LA IMPORTANCIA DE LAS REGLAS DE PRODUCCIÓN

El propósito de construir modelos de clasificación no está limitado al desarrollo de predicciones acertadas, si bien ésta es principal intención, ***otro propósito fundamental es que dicho modelo sea inteligible (comprensible) para el ser humano***. En algunos casos el árbol de decisión es tan compacto y sencillo que puede ser fácilmente entendido. Sin embargo cuando la tarea de clasificación llega a ser más compleja los árboles crecen hasta alcanzar proporciones difíciles de manejar.

Algunas formas de superar esta barrera de comprensión han sido exploradas. un ejemplo de ello es la descomposición del árbol en pequeños árboles, ya que debido a su estructura inductiva permite que individuamente sean más fáciles de entender.

El ID3 alcanza esta meta de diferente manera, reexpresando un modelo de clasificación como reglas de producción cuya forma parece ser más comprensible que los árboles. Una forma simple de una regla de producción esta dada como sigue:

$$L \rightarrow R$$

en la cual el lado izquierdo (Left) es una conjunción de atributos-descripción con cierto valor, y el lado derecho (Right) es la clase. Para clasificar un caso usando un modelo de reglas de producción la lista de reglas es examinada para encontrar la primera cuyo lado izquierdo sea satisfecho por el caso a clasificar. Una clase es designada como un una categorización por defecto. La clase pronosticada es entonces la proporcionada por el lado derecho de dicha regla. Si el lado izquierdo no es satisfecho el caso asigna la clase default.

Hemos visto el funcionamientos de los arboles de decisión y el conjunto de reglas de producción, y la forma en que el ID3 expresa estos modelos de clasificación. Es conveniente ahora , conocer otros algoritmos que se tienen para hacer clasificación.

2.7.3.2 CLASIFICADORES BASADOS EN EJEMPLOS

Una forma de clasificar un caso es recordando un caso similar cuya clase sea conocida y predecir que clase tendrá el nuevo caso. Esta es la filosofía que fundamenta a los sistemas basados en ejemplos, que clasifican casos no conocidos remitiéndose a casos similares antes vistos. Algunas cuestiones centrales que deben ser consideradas en un clasificador basado en ejemplos son:

- ¿Qué casos de entrenamiento deben recordarse? Si todos los casos son incluidos el clasificador sería muy complicado de manejar y lento. Lo ideal es retener solo casos prototipo que juntos logren resumir toda la información importante.

- ¿Cómo puede medirse la similaridad de los casos? Si todos los atributos son continuos. podemos computar la distancia entre dos casos como la raíz cuadrada de la suma de los cuadrados de las diferencias de los atributos. Sin embargo esta medida puede complicarse dependiendo de la naturaleza del atributo.

- ¿Cómo debe ser relacionado el nuevo caso con el caso recordado? En este aspecto existen dos alternativas, usar simplemente el caso más similar recordado o bien usar algunos casos similares con pesos asociados de acuerdo a os distintos grados de similaridad que posean.

2.7.3.3 REDES NEURONALES

Las redes neuronales construyen modelos aprendiendo los patrones en los datos que se analizan. Las redes neuronales pueden usarse para predecir valores reales y no exactamente clases, sin embargo en un contexto de clasificación la predicción, es considerada, por lo que podríamos decir que son clasificadores de tipo implícito, consisten en "neuronas" o nodos interconectados que se organizan en capas. Las redes neuronales aprenden en forma supervisada o no supervisada. En la modalidad supervisada la red neuronal intenta predecir los resultados para ejemplos conocidos. Compara sus predicciones con la respuesta objetivo y aprende de sus errores.

Las redes neuronales supervisadas se emplean para predicción, clasificación y modelos de series históricas. El aprendizaje no supervisado es eficaz para la descripción de datos, pero no para la predicción de resultados. Las redes no supervisadas crean sus propias descripciones y validaciones de clase y trabajan exclusivamente a partir de los patrones de los datos. Una de las desventajas de este método es la dificultad para entender los modelos que produce.

En lo que se refiere a la minería de datos, las redes neuronales son funciones entrenables que aproximan los valores de la función proporcionada por el conjunto de entrenamiento, los árboles también resuelven el mismo problema desde una perspectiva diferente, ellos encuentran las regiones donde la función del conjunto de entrenamiento asume valores que coloca dentro de un cierto rango. Los árboles de inducción desarrollan esta tarea implementando una cascada de pruebas sobre la distribución de los datos en varias particiones. Como la región puede ser descrita a partir de reglas que definen dicha partición, ellos son empleados para inducir reglas.

Las redes y los árboles sirven básicamente para el mismo propósito, las primeras encuentran el valor de la función para una entrada arbitraria, y los árboles encuentran la región donde la función despeglará una salida arbitraria, sin embargo ambos muestran habilidades diferentes para manejar conjuntos específicos de datos, y esto nos da la pauta para elegir en un momento dado que algoritmo aplicar.

2.7.3.4 ALGORITMOS GENÉTICOS

Son métodos numéricos de optimización, en los que aquella variable o variables que se pretenden optimizar junto con las variables de estudio constituyen un segmento de información. Aquellas configuraciones de las variables de análisis que obtengan mejores valores para la variable de respuesta, corresponderán a segmentos con mayor capacidad reproductiva. A través de la reproducción, los mejores segmentos perduran y su proporción crece de generación en generación. Se puede además introducir elementos aleatorios para la modificación de las variables (mutaciones). Al cabo de cierto número de iteraciones, la población estará constituida por buenas soluciones al problema de optimización.

Los elementos requeridos para construir un clasificador genético son todos distribuidos, por lo que el desempeño del clasificador no depende críticamente de cualquier pequeña parte del modelo, es decir, algún caso puede ser olvidado, algún peso o liga puede dañarse o algún elemento descartado, sin destruir el desempeño del modelo.

Esta características le da cierta venta sobre modelos simbólicos como los árboles de decisión que son mucho más sensibles a pequeñas alteraciones.

2.7.3.5 LOS MÉTODOS ESTADÍSTICOS

En este contexto, la descripción del clasificador basado en ejemplos podría remitirnos al método k-nearest-neighbor para la clasificación de instancias no conocidas. De hecho, la estadística ha dado lugar a muchos métodos de clasificación, sin embargo como regla general la estadística tiende a enfocarse a tareas en las que todos los atributos tienen valores continuos muchos de ellos son también paramétricos, adaptan el modelo y encuentran valores apropiados.

Por ejemplo un clasificador lineal asume qué clases pueden ser expresadas como una combinación lineal que se ajuste mejor sobre el conjunto de entrenamiento. Los clasificadores de máxima probabilidad, frecuentemente asumen que los valores de los atributos están distribuidos normalmente, entonces usan el conjunto de entrenamiento para determinar la distribución, varianza, covarianza etc. (algunos autores han incluso agrupado a las redes neuronales con procedimientos de parámetros estadísticos, ya que desde el entrenamiento una red generalmente significa encontrar valores apropiados para determinar pesos y sesgos.

2.8 EJEMPLO DE APLICACIÓN DEL KDD EN EL SECTOR FARMACÉUTICO

En este campo el KDD es aplicado como parte de la solución del problema global que representa la administración de la información química[25]. La industria química y farmacéutica produce una gran cantidad de información ya sea como información de procesos internos o como resultado de la publicidad. Compañías como Sandoz y Pfizer tienen la necesidad de construir las más grandes bases de datos, introduciendo e integrando datos adicionales información biológica, la cuestión de cómo minar todos esos datos para ayudar a los investigadores a verificar sus hipótesis, es identificada como uno de los principales problemas de esta área.

Existen diversas fuentes de información, primero están las bases de datos comerciales de productos químicos y la información relacionada, estas bases de datos contienen información acerca de la estructura y propiedades de los compuestos químicos, en segundo lugar hay

[25] Decker,Karsten. Op.cit. pp.15-18.

resultados producidos por universidades y laboratorios de investigación que son publicados en el mundo científico y revistas científicas. Otra fuente de información son los datos generados por el desarrollo de experimentos, y comportamiento de productos comerciales, de la investigación de la propia compañía. Lo anterior podemos resumirlo en cuatro principales categorías: datos químico- físicos (sobre la estructura y propiedades de las moléculas), datos sobre el comportamiento químico-físico de los compuestos, datos de las pruebas de campo de los compuestos, y finalmente información adicional sobre patentes y otros factores de índole económico y legal. Los avances en esta industria han producido una explosión en la cantidad de datos que cualquier investigador necesita manejar. para el trabajo sobre nuevas moléculas. Usados de manera apropiada estos datos pueden ser muy útiles en el diseño racional de medicinas. materiales y compuestos industriales.

El manejo de la información química muestra básicamente dos tipos de necesidades: primero, ayudar al descubrimiento de nuevos compuestos de utilidad, segundo, analizar e interpretar datos de campo para evaluar y mejorar la utilidad de productos y su contribución para el beneficio de la compañía. Ahora se comenzará explorando el uso de la tecnología de la información y en particular la minería de datos, en el proceso de descubrimiento. Existen muchos retos en el manejo de la información química tales como la nomenclatura que se utiliza, para escribir los compuestos. ¿Cómo usan los farmacéuticos esta información y cómo encuentran información que pueda ayudarles en el proceso de descubrimiento?.

Las operaciones que realizan sobre las bases de datos están dentro de las siguientes categorías:

- Agrupamiento de moléculas por su similitud
- Buscar todas las moléculas que contienen patrones específicos
- Buscar las moléculas que atan en una forma específica con una molécula objetivo
- Predecir los resultados de experimentos sobre nuevas moléculas a partir de datos almacenados

La evaluación de similitud es un importante paso del proceso de descubrimiento, porque permite explorar diferentes compuestos que comparten importantes características. El agrupamiento o clustering es básicamente el método que debe utilizarse, la definición de similitud es de alguna manera arbitraria; consecuentemente existen diferentes medidas de similitud que son usadas en química (coeficiente de Tanimoto), y cada una aporta criterios diferentes para dicho agrupamiento.

La búsqueda de moléculas que contengan un patrón específico como una subestructura es otra importante herramienta conceptual, en el proceso de descubrimiento del área farmacéutica, esto es básicamente un problema de reconocimiento de patrones. En este caso un filtro para la base de datos genera específicamente una subestructura, ya que el sistema debe recuperar todas las moléculas que son similares a la que se estructuró de acuerdo a la elección métrica.

Las predicciones se requieren para obtener de la actividad farmacéutica de nuevos compuestos basados en la actividad de compuestos conocidos. Anteriormente estas predicciones se habian realizado con la ayuda de un sistema experto pero los esfuerzos de investigación de algunas compañías ahora están encaminados al uso de redes neuronales para esta aplicación. Glaxo esta experimentando con una red neuronal en determinación espectral. Akzo Pharma está usando las redes neuronales con algoritmos genéticos en relaciones de estructuras cuantitativas, para reducir los datos necesitados para encontrar relaciones significativas . La segunda área importante para el desarrollo de minería de datos es el análisis de datos de campo, en particular, pruebas clínicas y datos que proceden del uso de químicos industriales. Esta es un área de uso potencial de sofisticadas técnicas de mineria de datos, dado el tamaño de datos generados y la importancia de encontrar correlaciones entre estos.

Sin embargo, la investigación en el campo demuestra que la mayoría de las compañías usa métodos de estadística estándar disponible en paquetes comerciales. Esto es comprensible, mientras la necesidad primaria de la compañia farmacéutica sea conseguir la aprobación de los medicamentos; los datos deben presentarse en forma estándar. Sin embargo, las técnicas minería de datos pueden traer significativos beneficios, si integran apropiadamente el proceso de descubrimiento dentro de la corporación.

3. NOCIONES DE LA TEORÍA DE LA INFORMACIÓN Y FUNDAMENTOS DEL ALGORITMO ID3

3. NOCIONES DE LA TEORÍA DE LA INFORMACIÓN Y FUNDAMENTOS DEL ALGORITMO ID3.

3.1 LA TEORÍA DE LA INFORMACIÓN

3.1.1 DEFINICIÓN

La Teoria de la Información se refiere al análisis de una entidad llamada *Sistema de Comunicación*[26] que es representado por el diagrama de bloques de la fig. 3.1. La fuente del mensaje es la persona o máquina que produce la información que será comunicada. El codificador asocia a cada mensaje un "objeto" que es el adecuado para la transmisión del canal, dicho objeto puede ser una secuencia de dígitos binarios o una forma de onda continua. El canal es el medio sobre el cual el mensaje codificado es transmitido. El decodificador opera en la salida del canal y su función es extraer el mensaje original para enviarlo al destino. En general esto no se logra por completo, debido a la presencia de un efecto denominado "ruido" que tiende a producir errores en la transmisión.

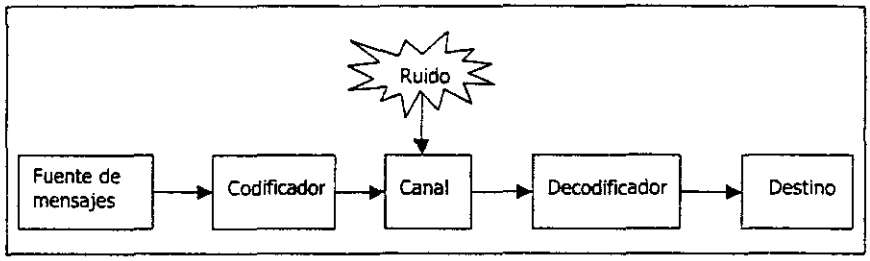

Fig. 3.1 Sistema de Comunicación[27]

La teoria de la Información responde a dos cuestiones fundamentales en la Teoria de la Comunicación: Cual es la última compresión de datos (respuesta: la entropía H), y que es la última velocidad de comunicación (respuesta: el canal de capacidad C). Por esta razón algunos consideran a la teoria de la Información como un subconjunto de la Teoría de la Comunicación. En este caso especial podemos circunscribir el estudio de la teoria de la información al estudio de un teorema llamado el Teorema fundamental de la teoria de la información, el cual dice que "Es posible transmitir información a través de un canal con "ruido" en cualquier velocidad menor que la capacidad del canal con una arbitrariamente pequeña probabilidad de error"[28].

[26] Ash, Robert. Information Theory. pp. 1-5.
[27] Ibid
[28] Cover, Thomas. Elements of Information Theory. pp. 1-8.

3.1.2 LA TEORÍA DE LA COMUNICACIÓN Y TEOREMAS DE SHANNON

Si la capacidad de un canal de comunicación es el límite máximo no superable por la cantidad real del flujo de información ¿Hasta qué punto se puede uno acercar a ese límite y con qué medios?. La respuesta la obtenemos de dos famosos teoremas de C.E. Shannon sobre codificación[29]. Hemos de recordar que un código es a un canal de comunicaciones lo que un transformador es a una transmisión eléctrica, es decir, un medio de mejorar su eficacia operativa. Así como generalmente usamos un transformador adecuado a la resistencia de carga del transmisor, para obtener un máximo de transferencia de potencia desde un generador hasta el receptor, también necesitamos un código adecuado a la estructura estadística del lenguaje usado en la transmisión. Por lo anterior no es una coincidencia que los teoremas de codificación estén relacionados con la utilización máxima de la capacidad del canal.

Las características del canal exigen muchas veces la codificación al ser ésta la única forma en que la señal pueda viajar por él. Un ejemplo de esto es la sucesión de puntos y rayas de la telegrafía, en que hay que transformar el mensaje antes de transmitirlo, y descifrarlo a su llegada a la terminal. En general, el transmisor o fuente del mensaje da el mensaje a un codificador antes de la transmisión y el decodificador lo devuelve a su forma original en cuanto llega al receptor. Evidentemente, dado un sistema de comunicaciones y el tipo de símbolos que puede manejar, puede escogerse arbitrariamente cualquier código construido con ellos para cifrar el mensaje aunque no todos los códigos permitirán la misma cantidad de flujo de información por el canal.

El primer teorema de Shannon *para canales "sin ruido"* tiene como núcleo la existencia de un código óptimo, bajo cualquier circunstancia, según el cual se nos garantiza un código óptimo para acoplar cualquier canal al lenguaje usado en la transmisión, a pesar de no poderlo crear en la realidad. No es un inconveniente insuperable el hecho de que esta primera aportación de Sahnnon no dé en la mayoría de los casos códigos óptimos, D.A. Huffman sugirió una ingeniosa corrección que lleva un método sencillo para construir códigos óptimos separables. Los códigos de Huffman son óptimos en el sentido de que la longitud media por mensaje, o el número de dígitos binarios usados en cada mensaje es menor que en cualquier otro código diseñable.

[29] Singh J. Ideas Fundamentales sobre la Teoría de la Información del lenguaje y de la Cibernética. Cap.4.*Passim*.

El segundo Teorema de codificación de Shannon, *para canales con "ruido"* (canales sujetos a alteraciones de las señales de transmisión), a grandes rasgos afirma que la capacidad del canal, como se definió anteriormente, es precisamente la cantidad máxima de información por unidad de tiempo que se puede recibir con una confianza bastante grande a pesar de todas las alteraciones producidas por el "ruido". El resultado es bastante sorprendente, sin lugar a dudas, ya que, a primera vista se espera que ocurra precisamente todo lo contrario. Por ejemplo, si nuestro canal de comunicaciones por teletipo, que transmite 0 y 1 con la misma probabilidad, fuese tan "ruidoso" que cualquier símbolo transmitido pudiese ser igualmente correcto que incorrecto, no sería difícil ver que no habría comunicación posible. El análisis matemático más riguroso lo confirma.

El método más evidente para lograr una transmisión fiable en presencia del "ruido" es repetir el mensaje suficientes veces para garantizar que la recepción sea fiable. Uno puede, por ejemplo repetir cada símbolo de una señal un número impar de veces y decidir en el receptor mediante un voto mayoritario que fue lo transmitido. Naturalmente, mientras más sean las repeticiones, mayor será la fiabilidad. Pero un gran numero de repeticiones traen consigo su propia desventaja ya que la confianza total conlleva la extinción , casi total, de la transmisión. Sin duda existe un gran número de alternativas intermedias que aseguran cierto grado de fiabilidad con una cantidad razonable de repeticiones o redundancias. Pero sigue permaneciendo el dilema de la fiabilidad por medio de la redundancia, es decir cuanto mayor sea esta última menor será la cantidad de flujo informativo asociado con ella.

Como resultado de lo anterior, las investigaciones en este campo, en vez de tratar de concentrarse en la creación de tales códigos, han centrado su atención en construir códigos que detecten errores y así proteger al mensaje transmitido de varios tipos de alteraciones producidas por el "ruido".

3.1.3 LA ENTROPÍA

3.1.3.1 DEFINICIÓN DE ENTROPÍA

La entropía de una variable aleatoria X con una probabilidad total función de p(x) es definida por:

$$H(x) = -\sum p(x) \log_2 p(x)$$

La entropía entonces es una medida de la incertidumbre promedio en la variable aleatoria, esto es el número de bits sobre el promedio requeridos para describir la variable aleatoria[30]. Ejemplo.

Considere una variable aleatoria que tiene una distribución uniforme con 32 resultados. Para identificar una salida, nosotros necesitamos una etiqueta que tome 32 diferentes valores. Así una cadena de 5 bits es suficiente como etiqueta. La entropía de esta variable aleatoria es :

$$H(x) = -\sum_{i=1}^{32} p(i)\log_2 p(i) = -\sum_{i=1}^{32} \frac{1}{32}\log_2 \frac{1}{32} = \log_2 32 = 5bits$$

El resultado obtenido coincide con el número de bits necesitados para describir X. En este caso todos los resultados tienen representación de igual longitud.

3.1.3.2 LA MEDIDA DE LA INFORMACIÓN

Considerando cualquier fuente de información analicemos si existe un fundamento seguro para una teoría métrica. Dicha fuente produce mensajes seleccionando sucesivamente símbolos discretos de un repertorio dado, como son letras de un alfabeto, palabras de un diccionario, las notas de la escala musical, los colores del espectro etc.

En otras palabras el mensaje que se transmite es una selección determinada de un conjunto de mensajes posibles formados por sucesiones de símbolos dados. El sistema de comunicación está ideado para que pueda transmitir toda posible selección. La técnica del proceso de comunicación no presta atención al contenido del mensaje en cuestión, por lo tanto la Teoría Métrica de la información no se ocupará del contenido semántico del conjunto de mensajes, del cual seleccionamos uno para transmitirlo. Como se requiere definir claramente, por lo menos para este caso específico, el significado de *"información"*, lo tomaremos como la medida de nuestra libertad de elección al escoger un mensaje del conjunto de mensajes disponibles, aunque muchos de ellos carezcan de significado.

Digamos que existen en total *n* mensajes entre los que podemos escoger, y que si cada mensaje tiene la misma probabilidad de ser escogido que cualquier otro, ese mismo número *n* se puede usar como una medida de la cantidad de "información" contenida en el conjunto de mensajes susceptibles de ser transmitidos.

[30] Ibid. p.5.

Es importante que exista una paridad entre los mensajes, expresada por la igual probabilidad de ser seleccionados, pues si decidimos escoger *n* como medida del contenido informativo del sistema de comunicación, no se puede admitir ninguna discriminación como las que surgirán con la ausencia de paridad. Cuando pasamos de sistemas simples a otros más complejos, esta medida de información, aunque sencilla, se hace incómoda. Supongamos por ejemplo, que se tiene la clave Morse con sus dos símbolos, un punto y una raya. Si unimos tres de esas claves para producir un nuevo conjunto, entonces constará de mensajes distintos entre los cuales se puede escoger.

$$\log(2x2x2) = \log 2 + \log 2 + \log 2$$

Si escogemos como medida del contenido informativo de un sistema el número de mensajes posibles, la medida de cada clave Morse tomada aisladamente será dos y la de la combinación de tres será ocho. Pero la medida no es apropiada debido a que se puede esperar que el contenido del sistema total sea la suma de los componentes individuales, es decir 2+ 2+ 2 = 6 y no su producto 2 X 2 X 2 = 8, como en realidad resulta ser, como se sabe los logaritmos transforman productos de números en sumas, es decir, el logaritmo de un producto de números de un conjunto cualquiera es la suma de sus logaritmos en este caso particular. Esta propiedad aditiva de los logaritmos es una gran ventaja ya que si se decide medir la información de la clave Morse por ejemplo, no con el 2 sino con el log 2, se tiene la certeza de que la medida de la información del conjunto de las tres claves juntas sea la suma de sus componentes individuales:

$$\log 2 + \log 2 + \log 2 = \log 2X2X2 = \log 8$$

Por esto los matemáticos prefieren medir el contenido de la información de un sistema de comunicaciones, que tenga n mensajes igualmente probables, por log n en vez de por n, dejando que la base logarítmica se escoja libre y arbitrariamente, según convenga el caso, El número 2 es el número más conveniente para escoger como base logarítmica por tratarse del número mínimo de mensajes que poseen incluso los sistemas de comunicación más rudimentarios, como la clave Morse.

Así la medida de la información de sistemas que tengan un repertorio de mensajes binario o doble, será : $\log_2 2$ es decir la unidad. Se ha llegado, ahora a una unidad natural de medida del contenido informativo de los sistemas de comunicación con las siguientes caracteristicas : un sistema con igual probabilidad de elección entre dos alternativas.

Esta unidad ha sido llamada *bit*, palabra formada por la contracción de "binary unit" (unidad binaria). Su mérito principal es que cumple con lo que vimos anteriormente, si el contenido informativo de un sistema de comunicaciones binario, como el timbre de una puerta o la clave Morse, es un "bit", entonces, el conjunto de *m* de estos sistemas será *m* bits, lo que demuestra la utilidad de haber introducido los logaritmos. Una limitación sería del sistema de medida es la necesidad de que existe una probabilidad semejante en la elección de todos los mensajes del sistema con lo que se garantiza la paridad matemática aunque simplifica excesivamente la situación real en la que se encuentran la mayoría de los sistemas de comunicación.

Una posible solución para la corrección de esta situación está en el hecho de que la falta de paridad entre los mensajes de un sistema de comunicaciones puede ser la probabilidad de que cada uno de ellos sea elegido. Así en el caso de la clave Morse, con su mayor probabilidad para los puntos que para las rayas, llamaremos paridad relativa de los dos mensajes a la probabilidad que cada uno de ellos tienen de ser seleccionado. Tomemos 9/10 ó 0.9, para un punto y 1/10 ó 0.1 para una raya. Ahora el sistema puede ser descrito mediante las probabilidades respectivas de un punto y una raya, en lugar de por el número 2 usado cuando ambas eran igualmente probables. Es obvio que utilizando el principio logaritmico, que ya habíamos encontrado, tendremos que la información aportada por un punto será: $\log_2(0.9)$ bits y la de una raya será $\log_2(0.1)$ bits. La información total del sistema será la suma de los componentes, en la que cada uno aporta su propia probabilidad de selección:

$$-\left(0.9\log_2 0.9 + 0.1\log_2 0.1\right) = 0.476\text{bits}$$

Como se aprecia en la ecuación el signo negativo antes del paréntesis se añade para hacer la cantidad positiva, ya que los logaritmos de todas las fracciones propias como 0.9 y 0.1 son números negativos. Hay que esperar que disminuya el contenido informativo del sistema, de un bit a casi medio, con la simple pérdida de paridad entre los dos mensajes, ya que al existir una marcada preferencia del sistema por los puntos, es más fácil predecir el resultado que anteriormente cuando ambas alternativas tenían la misma probabilidad. Si se generaliza el sistema para que incluya más de dos mensajes, por ejemplo n, cada uno con su propia probabilidad de selección, $p_1, p_2, p_3, \ldots p_n$. La medida de su información será análogamente la suma ponderada de cada contribución $\log_2 p$, es decir:

$$-\left(p_1\log_2 p_1 + p_2\log_2 p_2 + \ldots + p_n\log_2 p_n\right)\text{bits}$$

Es evidente que el contenido informativo de un repertorio de mensajes es una función de las probabilidades de ocurrencia de cada uno de los mensajes comprendidos en el sistema. Cambiando de mensajes dados, cambiamos totalmente el contenido informativo, es obvio por lo tanto, que el patrón probabilístico del conjunto de mensajes es la clave de su contenido informativo[31].

3.1.3.3 RELACION ENTRE LA INFORMACIÓN Y LA ENTROPÍA

Como ya se explicó el contenido informativo de un mensaje dentro de cualquier conjunto de mensajes esta dado por el logaritmo de su probabilidad de aparición. Esta forma de definir la información tiene un precedente en la mecánica estadística, donde la medida de la entropía es de forma idéntica a la de la información[32]. No es ninguna casualidad por lo tanto, que exista entre ellas dos una relación bastante estrecha. Para empezar ambas, la Teoría de la Información y la mecánica estadística son estadísticas. Ya hemos visto que la probabilidad de un mensaje, entre varios, es la frecuencia relativa de que aparezca en un número grande de casos. Las estadísticas sobre la incidencia de mensajes en casos repetidos son, por lo tanto, la misma de su contenido informativo. En la mecánica estadística también como implica el nombre, se trata de deducir el comportamiento de los cuerpos aplicando consideraciones estadísticas a los grupos de moléculas que los conforman.

Ahora bien, siempre que nos encontramos con grupos de entes (ya sean hombres, mensajes o moléculas) se nos presentan dos modos de estudiarlos. Ya sea especificando el (o los) atributo(s) estudiando a cada uno de los individuos del grupo, o bien especificando la(s) media(s) estadísticas totales de su(s) atributo (s) individual(es). La primera forma de estudio se dice que define la estructura interna o *microestado* del grupo. y la segunda su fachada exterior o *macroestado.* Consideremos el atributo edad en una clase de alumnos.

Su microestado se define con la edad de cada uno de los individuos. Su macroestado puede ser definido por la *edad media* de la clase completa. Obviamente, para cada macroestado dado de la clase (la edad media) existen una gama completa de posibilidades de microestados (varios patrones de edad que den la misma media).

---

[31] Singh J. Op.cit. Cap.2.

[32] Ibid. Cap. 7.

Al fijar la edad media de la clase, no se determina la distribución de las edades de los individuos. Sin embargo, cuando el número de individuos en una clase se hace muy grande, la especificación de su macroestado, por medio de la edad media de sus miembros, puede ser el único camino abierto, para saber algo del atributo edad.

Esta dificultad creada por la aglomeración de números en un grupo, se acentúa más en la mecánica estadística. Aquí los grupos de moléculas que forman los cuerpos en estudio son tan enormes, que cualquier intento de hacer una descripción detallada resulta prácticamente imposible. Por eso, la mecánica estadística supera esta dificultad, al tratar de captar el movimiento del conjunto completo de moléculas por medio de un estudio estadístico del grupo. Por ejemplo, calcula la energía media del conjunto de moléculas y la identifica con la medida de su temperatura.

De la misma forma que a cada macroestado de un grupo, definido por los valores medios de uno o más de sus atributos, le corresponden en general varios microestados del grupo, a cada macroestado del movimiento de las moléculas de un cuerpo, definido por la velocidad media de sus moléculas (o lo es lo mismo, por su temperatura), le corresponden muchos microestados de su movimiento molecular. No es sorprendente, pues, que en un cuerpo de billones de moléculas haya un número grande de microestados distintos de movimiento, cada uno de los cuales puede pertenecer a un macroestado con la misma velocidad media por molécula. Para nuestros inexactos sentidos, un cuerpo en cualquiera de estos microestados distintos parecerá estar a la misma temperatura, pero detrás de la aparente estabilidad de su velocidad la temperatura media se está produciendo un cambio continuo de transiciones de un microestado a otro, que no puede ser detectado por la medida de la temperatura del cuerpo.

El número de los distintos microestados que corresponden a un macroestado dado, definido por cualquier temperatura T, es conocido como la probabilidad termodinámica. La razón que existe para llamarla "probabilidad" se basa en que cuanto mayor sea el número de microestados correspondientes al macroestado definido por la temperatura T, mayor será la posibilidad de que cualquier microestado escogido al azar manifieste la característica externa de ese macroestado, es decir, la temperatura T. La probabilidad termodinámica de un cuerpo nos proporciona, por tanto, una medida de la información sobre el estado de los movimientos internos aunque sea de una forma negativa.

Un gran número de microestados, es decir, una probabilidad termodinámica alta, corresponde a un gran desorden y una escasa uniformidad en la composición interna, por lo que la probabilidad termodinámica (o mejor dicho, su logaritmo, que llamamos entropía) es realmente un índice del caos molecular existente en el interior. Se trata además de una medida útil, necesaria para la determinación cuantitativa de la tendencia de los procesos naturales, cuando los grupos de moléculas que componen un cuerpo son dejadas para que actúen por sí solas.

Así la entropía sirve para dos fines relacionados entre sí: primero, es lo que Eddington llamó la flecha del tiempo, es decir, un indicador de la tendencia de los procesos naturales. Segundo; nos revela cuantitativamente la estructura estadística de movimientos internos en forma muy parecida a como lo hace la Teoría de la Información con un conjunto de mensajes. Más aún lo hace de forma análoga, ya que tomamos la entropía como el logaritmo de la probabilidad termodinámica de un macroestado tal como medimos la información de un mensaje por el logaritmo de la probabilidad de su aparición.

Evidentemente, la información y la entropía son dos caras de la misma moneda, en el sentido de que el orden interno u organización, implicando un mayor conocimiento o información de la composición interna del sistema, va siempre acompañado de una probabilidad termodinámica baja, o mejor dicho, de su logaritmo o entropía. En cualquier sistema dado, cuanto mayor sea el número de estados microscópicos correspondientes a cualquier macroestado dado, mayor será su entropía,

De aquí se deduce que la entropía es una medida de nuestra ignorancia en el conocimiento de la estructura ultramicroscópica. En otras palabras, la entropía es el negativo de la información, por tal razón L. Brillouin creó el término "negentropía" acortando la frase "negativo de la entropía". Este extraordinario parecido entre la información y la entropía fue advertido por primera vez por L. Szilard al resolver la paradoja del demonio de Maxwell, enunciada por J.C. Maxwell en su Teoría del calor en 1871.

3.2 FUNDAMENTOS DEL ALGORITMO ID3

3.2.1 INTRODUCCIÓN

El aprendizaje inductivo es un caso particular entre las técnicas de aprendizaje a partir de ejemplos, siendo su cometido el inducir reglas a partir de los datos históricos disponibles para lo cual procederá a clasificar en la clase correspondiente diferentes objetos, basándose en el valor de las caracteristicas o atributos que los definen. Para la aplicación de estos sistemas de aprendizaje inductivo se parte de un conjunto de ejemplos (denominado conjunto de entrenamiento). Cada ejemplo debe tener la misma estructura consistente en una conclusión (o decisión) y un número de características o atributos que definen esa conclusión o decisión[33].

Un árbol de decisión representa la relación existente entre la conclusión-decisión y sus atributos. Es decir, se produce un proceso de generalización de forma que el árbol de decisión generado clasifica correctamente los ejemplos dados. Los árboles de decisión clasifican instancias o ejemplos ordenándolos bajo un árbol desde la raiz hasta algún nodo hoja que proveerá de clasificación de la instancia. Un árbol de decisión toma como entradas objetos o situaciones caracterizadas mediante un conjunto de propiedades; el árbol entrega a la salida una "decisión" sí, o No. La estrategia de construcción del árbol consiste en seleccionar (en cada momento) aquel atributo potencialmente más útil para la clasificación, entendiendo como tal aquel que prometa generar el mejor árbol a partir de ese momento.

Los árboles de decisión son poderosas herramientas para clasificación y predicción, lo más atractivo de usar estos métodos es que pueden también ser representados como conjuntos de reglas IF-THEN para perfeccionar el conocimiento humano. Lo anterior es muy importante, ya que es una alternativa para la adquisición del conocimiento, pudiendo convertir una base de datos existente en un conjunto de reglas.

3.2.2 PROBLEMAS APROPIADOS PARA APRENDIZAJE CON ÁRBOLES DE DECISIÓN

Si bien una variedad de métodos de aprendizaje con árboles decisión han sido desarrollados con algunas capacidades y requerimientos diferentes, el aprendizaje con árboles de decisión se adapta mejor a problemas con las siguientes caracteristicas:

[33] Cfr. Mitchell, Tom. Machine Learning. Cap.3. Passim.

❖ **Instancias representadas por pares "atributo-valor"**

Las instancias descritas por un arreglo fijo de atributos (temperatura y su valor (caliente). La situación más sencilla para árboles decisión es cuando cada atributo toma valores sobre un número pequeño de combinaciones posibles por ejemplo caliente, templado, frio. Sin embargo extensiones del algoritmo básico permiten también manejar valores reales de atributos por ejemplo representar temperatura numéricamente.

❖ **La función objetivo tiene valores discretos en la salida**

El árbol de decisión de la figura 3.1, asigna una clasificación booleana ("si" o "no") para cada ejemplo. Los árboles de decisión fácilmente extienden su aprendizaje a funciones con más de dos posibles valores, una modificación en el algoritmo permite aprender funciones objetivo con valores reales, sin embargo la aplicación de árboles de decisión en este conjunto es menos común.

❖ **Descripciones disyuntivas pueden ser requeridas:**

Como se observó arriba, los árboles de decisión naturalmente representan expresiones disyuntivas.

❖ **Los datos de entrenamiento pueden presentar atributos faltantes:**

Los árboles de decisión se pueden usar, incluso cuando algunos ejemplos de entrenamiento tienen valores desconocidos (por ejemplo si la humedad del día fuera conocida solamente para algunos de los ejemplos de entrenamiento ver tabla 3.1). En la práctica se han encontrado algunas técnicas que han permitido subsanar estas deficiencias.

3.2.3 EL ALGORITMO ID3

El algoritmo ID3 fue desarrollado por J. Ross Quinlan en 1979, y pertenece a la familia TDIDT (Top-down Induction of decision trees). El algoritmo construye el árbol seleccionando en cada momento el mejor atributo según una medida heurística, por lo que puede ser visto como una búsqueda hill-climbing sin vuelta a atrás (no backtracking) a través de todos los posibles árboles[34].

[34] Cortés Garcia, Ulises. Op.cit. p.126.

La mayor parte de los algoritmos que han sido desarrollados para árboles de decisión, son variaciones sobre éste algoritmo central o básico y su sucesor C4.5. El ID3 (Interactive Dichotomizer 3) es un algoritmo potente, cuya misión es la elaboración de un árbol de decisión bajo las siguientes premisas:

1. Cada objeto o instancia de la secuencia de entrada presentada al algoritmo toma la forma de una lista de pares atributo-valor. Cada nodo corresponde a un atributo y cada rama al valor posible de ese atributo.

2. Cada instancia va así mismo acompañada de la clase a la que pertenece. Una hoja del árbol especifica el valor esperado de la decisión o clasificación, de acuerdo con los ejemplos dados. La explicación de una determinada decisión viene dada por la trayectoria desde la raiz a la hoja representativa de esa clasificación.

3. A cada nodo es asociado aquel atributo más informativo que aún no haya sido considerado en la trayectoria desde la raíz.

4. Para medir cuánto de informativo es un atributo se emplea el concepto de entropia. Cuanto menor sea el valor de la entropia, menor será la incertidumbre y más útil será el atributo para la clasificación.

El ID3 es capaz de tratar con atributos cuyos valores sean discretos o continuos. En el primer caso, el árbol de decisión generado tendrá tantas ramas como valores posibles tome el atributo. Si los valores del atributo son continuos, el ID3 no clasifica correctamente los ejemplos dados. Por ello, Quinlan propuso el C4.5, como extensión del ID3[35], que permite:

- Construir árboles de decisión cuando algunos de los ejemplos presentan valores desconocidos para algunos de los atributos.

- Trabajar con atributos que presenten valores continuos.

- La poda de los árboles de decisión. El árbol de decisión ha sido construido a partir de un conjunto de ejemplos, por tanto, reflejará correctamente todo el grupo de casos. Sin embargo, como esos ejemplos pueden ser muy diferentes entre sí, el árbol resultante puede llegar a ser bastante complejo, con trayectorias largas y muy desiguales.

[35] Quinlan,J.R. Programs for Machaine Learning. pp. 1-60

Para facilitar la comprensión del árbol puede realizarse una poda del mismo, lo que significa la sustitución de una parte del árbol (sub-árbol) por una hoja. La poda tendrá lugar si el valor esperado de error en el sub-árbol es mayor que con la hoja que lo sustituya.

3.2.3.1 REPRESENTACIÓN

Este es un típico árbol de decisión basado en el algoritmo ID3. Este árbol de decisión clasifica que sábados por la mañana, de acuerdo al clima son apropiados para jugar tenis.

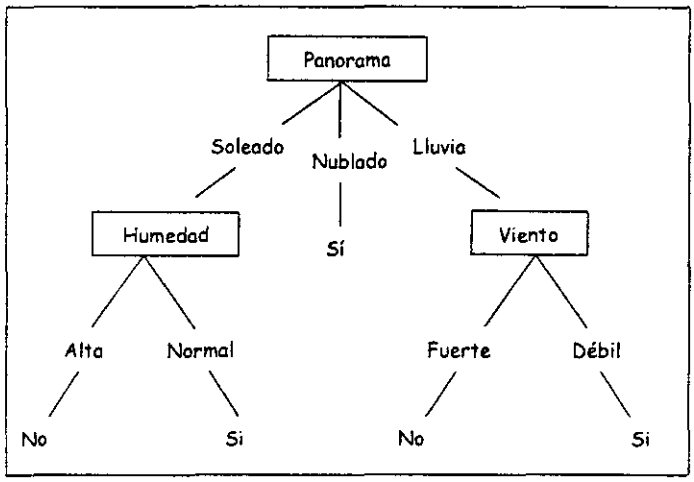

Fig. 3.2 Representación del algoritmo ID3

Como se muestra en la figura 3.2 los ejemplos se caracterizan mediante los valores de atributos y el valor de la función objetivo o clasificación. Si dicha clasificación es válida para cierto ejemplo, se dice que es un *ejemplo positivo*, en caso contrario se dice que es un *ejemplo negativo*. Lo anterior puede ilustrarse a través de la siguiente instancia con los valores siguientes:

Panorama = Soleado
Temperatura = Caliente
Humedad = Alta
Viento = Fuerte

El ejemplo o instancia puede ser clasificado bajo la rama más a la izquierda del árbol, y por lo tanto puede ser identificado como un ejemplo negativo (el árbol predice que jugar tenis = no). En general los árboles decisión representan una disyunción de conjunciones de restricciones, sobre los valores de las instancias. Cada ruta de la raíz del árbol a una hoja corresponde a una conjunción de pruebas de atributos. Por ejemplo, la decisión del árbol mostrado corresponde a la expresión.

(Panorama = soleado Humedad = normal)
(Panorama = nublado)
(Panorama = lluvia Viento = débil)

3.2.3.2 ¿ CÓMO TRABAJA EL ID3?

El algoritmo básico ID3 es un árbol de decisión constituido como un "top-down", comencemos con la pregunta ¿Qué atributo puede ser probado en la raiz del árbol?. Para responder está pregunta cada atributo de la instancia es evaluado usando una prueba estadística para determinar que también clasifica solamente ese ejemplo de entrenamiento. El mejor atributo es seleccionado y se usa como prueba en el nodo raíz del árbol[36].

Un descendente de la raíz es entonces creado por cada valor posible de este atributo y los ejemplos de entrenamiento son clasificados en un nodo descendiente apropiado (por ejemplo bajo la rama correspondiente a los valores del ejemplo para entrenamiento). El proceso completo es entonces repetido usando los ejemplos de entrenamiento asociados con cada nodo descendente. Esto forma una búsqueda exhaustiva para un árbol de decisión aceptable en la cual el algoritmo nunca regresa en su trayectoria para reconsiderar las primeras opciones.

3.2.3.3 QUE ATRIBUTO ES EL MEJOR CLASIFICADOR

La cuestión central en el algoritmo ID3, es elegir ¿cual atributo probar en cada nodo del árbol?. Una forma de empezar es seleccionando un atributo prueba que contenga el mejor desempeño al dividir la base de datos que contiene los ejemplos, en subconjuntos en los que muchas instancias tienen la misma clasificación.

[36] Cfr. Mitchell, Tom.Op.cit. Cap.3. Passim.

A continuación se define una propiedad estadística llamada *ganancia de información*, la cual nos indica que tan acertadamente, un atributo dado, separa los ejemplos de entrenamiento de acuerdo a la clasificación dada. El ID3 usa esta medida de ganancia de información para seleccionar de entre los candidatos que tiene, aquel atributo que utilizará en cada paso, mientras el árbol va creciendo.

3.2.3.4 ENTROPÍA: MEDIDA DE HOMOGENEIDAD

Para una base de datos de cualquier tamaño es poco probable que cualquier método aplicado, produzca incluso un subconjunto completamente homogéneo, en consecuencia para bases de datos reales se necesita un recurso poderoso para medir el desorden total, o falta de homogeneidad de los subconjuntos producidos. Para definir ganancia de información precisamente nosotros comenzamos por definir en el tema anterior, una medida usada comúnmente en teoría de la información, llamada *Entropía*, que caracteriza la impureza o no homegeneidad, de una colección arbitraria.

Dada una colección S, que contiene ejemplos positivos y negativos de algún objeto, el concepto de entropia de S relativo a esta clasificación de booleana es:

$$Entropia(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Donde $p_{\oplus}$ es la proporción de ejemplos positivos en S y $p_{\ominus}$ es la proporción de ejemplos negativos en S. En todos estos cálculos se define $0 \log 0 = 0$. Para ilustrar supongamos S como una colección de 14 ejemplos de algún concepto booleano, incluyendo 9 positivos y 5 negativos [9+, 5-] Entonces la entropía de S relativa a esta clasificación booleana es:

$$entropia(9+,5-) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

Como se observa la entropia es cero si todos los miembros de S pertenecen a la misma clase es decir son sumamente homogéneos.

Por ejemplo, si todos los miembros son positivos $(p_{\oplus} = 1)$ entonces, $(p_{\ominus} = 0)$ y la entropia resultaría como sigue:

$$entropia(S) = (-1)(\log_2(1)) - (0)(\log_2(0)) = 0$$

La entropía es 1 cuando la colección contiene un número igual de ejemplos de ejemplos positivos y negativos. Si la colección contiene desigual número de ejemplos positivos y negativos la entropia esta entre 0 y 1. *Una interpretación de entropia desde la teoria de la información es que especifica el mínimo número de bits de información necesitados para encontrar la clasificación de un miembro arbitrario de S.* De este modo se ha discutido el concepto de *ENTROPÍA* en el caso especial donde la clasificación es booleana. Generalizando tenemos que. si el atributo objetivo puede tomar uno o diferentes valores entonces la entropía de S respecto a la clasificación se define como:

$$Entropia(S) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i$$

Donde p_i es la proporción de S perteneciente a la clase de i, note que si el atributo objetivo puede tomar "c" valores posibles, la entropía puede ser tan grande como $\log_2 c$.

3.2.3.5 GANANCIA DE INFORMACIÓN

Dada la entropía como una medida de impureza en una colección de ejemplos de entrenamiento, nosotros podemos ahora definir una *medida de la efectividad de un atributo en la clasificación de un conjunto de entrenamiento.* La medida que usaremos la llamaremos *GANANCIA DE INFORMACIÓN*, es simplemente la reducción esperada en la entropia causada por la división de los ejemplos de acuerdo a este atributo. Más precisamente la ganancia de información de un atributo A es decir $Ganancia(S, A)$ relativa a una colección de ejemplos S, es definida como:

$$Ganancia(S, A) \equiv Entropia(S) - \sum_{v \in R \cdot \cdot \cdot (A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

Donde A es el conjunto de todos los posibles valores para el atributo A, y S_v es el subconjunto de S que para cada atributo A tiene un valor v.

El primer término en la ecuación es exactamente la entropía de la colección original S, y el segundo término es el valor esperado de la entropía después de que S es particionado usando el atributo A. La entropía esperada descrita por este segundo término, es simplemente la suma de las entropías de cada subconjunto, evaluado por la fracción de ejemplos $\frac{|S_v|}{|S|}$ que pertenece a S_v. La ganancia (S,A) es por lo tanto la reducción esperada en la entropía causada por el conocimiento del valor del atributo. De otra manera, la ganancia (S,A) es la información proporcionada acerca del valor de la función objetivo o clasificación, dado el valor de algún otro atributo A. El valor de la ganancia (S,A) es el número de bits ahorrados cuando codificamos. el valor objetivo de un miembro arbitrario de S, por conocimiento del valor del atributo A. Ejemplificando la Ganancia de Información

Día	Panorama	Temperatura	Humedad	Viento	Jugar tenis
D1	Soleado	Caliente	Alta	Débil	No
D2	Soleado	Caliente	Alta	Fuerte	No
D3	Nublado	Caliente	Alta	Débil	Sí
D4	Lluvia	Templada	Alta	Débil	Sí
D5	Lluvia	Fría	Normal	Débil	Sí
D6	Lluvia	Fría	Normal	Fuerte	No
D7	Nublado	Fría	Normal	Fuerte	Sí
D8	Soleado	Templada	Alta	Débil	No
D9	Soleado	Fría	Normal	Débil	Sí
D10	Lluvia	Templada	Normal	Débil	Sí
D11	Soleado	Templada	Normal	Fuerte	Sí
D12	Nublado	Templada	Alta	Fuerte	Sí
D13	Nublado	Caliente	Normal	Débil	Sí
D14	Lluvia	Templada	Alta	Fuerte	No

Tabla 3.1 Ejemplos de entrenamiento

Supongamos que S es una colección de ejemplos de entrenamiento, son días descritos por atributos incluyendo *viento*. que puede tener los valores débil o fuerte como vimos anteriormente. S es una colección que contiene 14 ejemplos [9+, 5-] (ver tabla 3.1) de estos 14 ejemplos supongamos que 6 de los positivos y 2 de los negativos, tienen viento = débil, y el resto tiene viento = fuerte.

La ganancia de información debida a la clasificación de los 14 ejemplos originales por el atributo *viento* puede entonces ser calculada como sigue:

Valores(viento) = débil, fuerte

$S = [9+,5-]$

$S_{débil} \leftarrow [6+,2-]$

$S_{fuerte} \leftarrow [3+,3-]$

$$Ganancia(S, viento) = Entropía(S) - \sum_{v \in \{débil, fuerte\}} \frac{|S_v|}{|S|} Entropía(S_v)$$

$$Ganancia(S, viento) = Entropia(S) - \left(\frac{8}{14}\right) Entropia(S_{débil}) - \frac{6}{14} Entropía(S_{fuerte})$$

$$Ganancia(S, viento) = 0.940 - \frac{8}{14}(0.811) - \frac{6}{14}(1.00)$$

$$Ganancia(S, viento) = 0.048$$

La ganancia de información es precisamente la medida usada por el ID3 para seleccionar el mejor atributo en cada paso del crecimiento del árbol. El uso de la ganancia para evaluar la relevancia de los atributos es resumido en la figura 3.3.

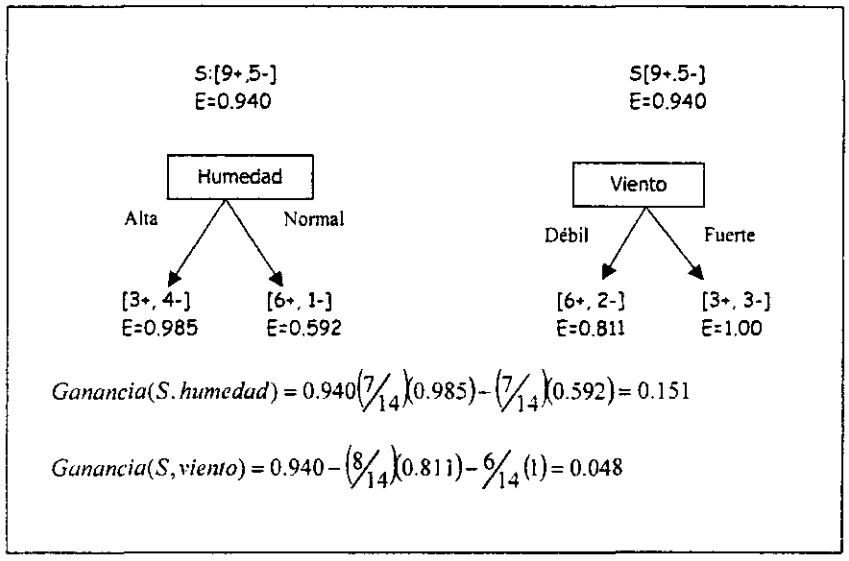

Fig. 3.3 La ganancia de información

Como se observa la ganancia de información de dos diferentes atributos, humedad y viento, es computada para determinar cual es el mejor atributo para clasificar los ejemplos de entrenamiento mostrados en la tabla 3.1.

Humedad proporciona una mayor ganancia de información que viento, en lo referente a la clasificación. Aquí E se encuentra para la entropia y S para la colección de ejemplos. Dada una colección inicial S [9+ 5-], clasificando estos por su humedad produce colecciones de [3+, 4 -] (humedad =alta) y [6+, 1-] (humedad normal). La información ganada por esta partición es 0.151 comparada con la ganancia de solo 0.048 para el atributo viento.

3.2.3.6 UN EJEMPLO ILUSTRATIVO DEL FUNCIONAMIENTO DEL ID3.

Para mostrar la operación del ID3, consideremos los ejemplos de entrenamiento de la tabla 3.1. Aquí el atributo-objetivo "jugar _tenis" que puede tener valores "si" o "no", para diferentes "sábados por la mañana", se puede predecir o pronosticar, basándose en otros atributos de las mañanas en cuestión[37]. Considere el primer paso a través del algoritmo, en que el árbol de decisión es creado. ¿Qué atributo puede ser probado primero en el árbol? ID3 determina la ganancia de información para cada atributo candidato (i.e., panorama, temperatura, humedad y viento) entonces selecciona, el que tiene más alta ganancia. El cálculo de la ganancia de información de todos estos atributos se muestra a continuación

La ganancia de información para los 4 atributos es:

Ganancia (S, Panorama) = 0.246
Ganancia (S,Humedad) = 0.151
Ganancia (S,Viento) = 0.048
Ganancia (S,Temperatura) = 0.029

Donde S denota la colección de ejemplos de entrenamiento de la tabla 3.1

De acuerdo a la medida de la ganancia de información el atributo *Panorama* proporciona la mejor predicción del atributo objetivo "jugar_tenis" sobre los ejemplos de entrenamiento. Por lo tanto *Panorama* es seleccionado como el atributo de decisión para el nodo raíz y las ramas son creadas debajo de la raíz por cada uno de sus posibles valores (i.e. Soleado, Nublado y lluvia) entonces selecciona el que tenga la mayor ganancia de información.

[37] Ibid. pp.59-60.

Note que cada ejemplo para que Panorama = Nublado es también un ejemplo positivo de "juego de tenis". Por lo tanto este nodo del árbol llega a ser un nodo hoja con la clasificación juega_ tenis = sí. En contraste, los descendientes correspondientes para panorama = soleado y panorama = lluvia todavía tienen entropía diferente de cero, y el árbol de decisión crecerá debajo de estos nodos. El proceso de selección de un nuevo atributo y partición de los ejemplos de entrenamiento se repite para cada nodo descendente no terminal, esta vez usando sólo los ejemplos de entrenamiento asociados con cada nodo.

Los atributos que han sido incorporados en la parte más alta del árbol son excluidos, entonces cualquier atributo dado puede aparecer más de una vez a lo largo de cualquier rama (o ruta) a través del árbol. Este proceso continúa para cada nueva hoja hasta que cualquiera de las dos condiciones siguientes se cumple(ver Figura 3.4) :

1. Cada atributo ha sido ya excluido a lo largo de una ruta a través del árbol, ó
2. Los ejemplos de entrenamiento asociados con este nodo hoja, tienen el mismo valor para el atributo objetivo (i.e., su entropía es cero).

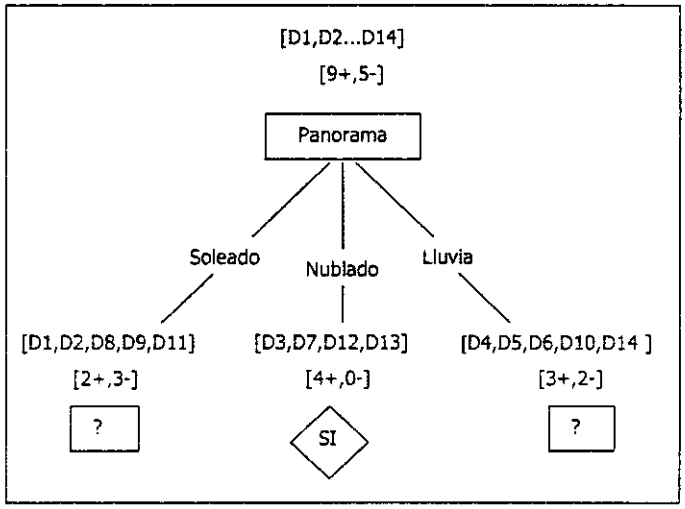

Fig. 3. 4 El funcionamiento del ID3

$$S_{soleado} = [D1, D2, D8, D9, D1]$$

$$Ganancia(S_{soleado}, Humedad) = 0.970 - \tfrac{3}{5}(0.0) - \tfrac{2}{5}(0.0) = 0.970$$

$$Ganancia(S_{soleado}, Temperatua) = 0.970 - \tfrac{2}{5}(0.0) - \tfrac{2}{5}(1.0) - \tfrac{1}{5}(0.0) = 0.570$$

$$Ganancia(S_{soleado}, Viento) = 0.970 - \tfrac{2}{5}(1.0) - \tfrac{3}{5}(0.918) = 0.019$$

Los ejemplos de entrenamiento son ordenados en los nodos descendientes correspondientes. Como se observa del atributo nublado solo descienden ejemplos positivos y por lo tanto se convierte en un nodo hoja con clasificación SI. Los otros dos nodos crecerán seleccionando el atributo con mayor ganancia de información relativa al nuevo subconjunto de ejemplos. El árbol de decisión final, obtenido por el ID3 a partir de los 14 ejemplos de entrenamiento de la tabla 3.1 es el mostrado en la figura 3.1.

3.2.4 ESPACIO DE BÚSQUEDA DE HIPÓTESIS EN APRENDIZAJE CON ÁRBOLES DE DECISIÓN.

Como con otros métodos de aprendizaje inductivo, el ID3 puede ser caracterizado como la búsqueda en un espacio de la hipótesis que ajuste los ejemplos de entrenamiento. ID3 desarrolla una búsqueda hill-climbing a través de éste espacio de hipótesis comenzando con el árbol vacío, entonces considera progresivamente hipótesis más elaboradas, tratando de encontrar un árbol de decisión que clasifique correctamente el conjunto de entrenamiento, la función de evaluación que guía esta búsqueda hill-climbing es la medida de la ganancia de información.

El espacio de hipótesis del ID3, de todos los árboles de decisión, es un espacio completo de funciones de valores discretos, relativo a los atributos disponibles. Dado que cada función finita de valores discretos puede ser representada por algún árbol de decisión, ID3 evita uno de los mayores riesgos de métodos que buscan hipótesis en espacios incompletos (tales como métodos que consideran sólo hipótesis conjuntivas); ya que dicho espacio puede no contener la función objetivo.

ID3 mantiene sólo una hipótesis activa mientras busca a través del espacio de árboles decisión. Esto contrasta por ejemplo con el método "eliminación del candidato", que mantiene el conjunto de todas las hipótesis consistentes con los ejemplos disponibles de entrenamiento(espacio de versiones).

Determinando una sola hipótesis, el ID3 pierde la capacidad que resulta de representar explícitamente todas las hipótesis consistentes. Por ejemplo, no tiene la habilidad para determinar si algunos árboles de decisión alternativos son consistentes con los datos de entrenamiento disponibles.

ID3 en su forma pura no desarrolla un backtracking (vuelta hacia atrás o regreso sobre su trayectoria) en su búsqueda. Una vez que selecciona un atributo para probar en un nivel particular en el árbol, nunca regresa para reconsiderar ésta elección. Por lo tanto, está susceptible al riesgo común de la búsqueda hill-climbing sin regreso sobre su trayectoria; convergiendo a una solución localmente óptima pero que no lo es de manera global.

ID3 usa todos los ejemplos de entrenamiento en cada paso en la búsqueda para tomar decisiones de manera estadística, considerando refinar su hipótesis actual. Una ventaja de usar las *propiedades estadísticas* de todos los ejemplos (i.e ganancia de información) es que la búsqueda resultante es mucho menos sensible a errores que en los ejemplos individuales de entrenamiento. ID3 puede ser modificado o ampliado para manejar datos de entrenamiento ruidosos modificando criterios de terminación para aceptar hipótesis que ajustan imperfectamente los datos de entrenamiento.

3.2.5 PREDISPOSICIÓN INDUCTIVA DEL ID3

¿Cuál es el criterio a partir del cual, el ID3 generaliza lo observado en ejemplos de entrenamiento para clasificar instancias o ejemplos no conocidas?, En otras palabras, ¿Qué es su *predisposición inductiva*?. La predisposición inductiva es el conjunto de suposiciones que junto con los datos de entrenamiento, deductivamente justifica la clasificación asignada por el aprendiz para instancias futuras.

Dada una colección de ejemplos de entrenamiento existen normalmente muchos árboles de decisión consistentes con los dichos ejemplos, ¿Cuál de estos árboles de decisión elige ID3? Elige el primer árbol aceptable que encuentra a través de su búsqueda, de manera burda podría decirse que la estrategia de este algoritmo es:

a) Inclinarse por el árbol más corto sobre los más grandes o largos.
b) Selecciona al que posiciona a los atributos con más alta ganancia lo más cerca de la raíz.

De lo anterior se puede definir esta predisposición como una preferencia por los árboles más sencillos o cortos sobre aquellos más grandes o complejos.

Predisposición a restricción y predisposición a preferencia

El ID3 busca en un espacio completo de hipótesis, (uno capaz de expresar cualquier función finita de valores discretos), busca de manera incompleta a través de este espacio desde las hipótesis, más simples hasta las más complejas, y termina hasta que su condición es satisfecha es decir hasta que encuentra una hipótesis consistente con los datos. La predisposición inductiva es una consecuencia del ordenamiento de las hipótesis por su estrategia de búsqueda, de este modo podemos clasificar dicha tendencia como una *preferencia* por ciertas hipótesis. Existen otras tendencias, con predisposición a la restricción categórica en un conjunto de hipótesis considerado. Aquí se busca en un espacio incompleto de hipótesis (i.e., uno que exprese solo un subconjunto de conceptos potencialmente susceptibles de ser aprendidos) busca en este espacio completamente, encontrando cada hipótesis consistente con los datos de entrenamiento.

Normalmente una tendencia a la preferencia es más deseable que una a la restricción, porque ello permite al *aprendiz* trabajar en un espacio completo de hipótesis que asegura contener la función objetivo, para la clasificación, en cambio una tendencia a restricción se limita estrictamente a un subconjunto de hipótesis introduce la posibilidad de excluir la función objetivo completamente.

3.2.6 CONSIDERACIONES SOBRE CUESTIONES QUE SE PRESENTAN EN LA PRÁCTICA

Cuestiones prácticas en el aprendizaje con árboles de decisión incluyen determinar el manejo de atributos continuos, conjuntos de entrenamiento con valores de atributos faltantes etc. Esto ha llevado a realizar algunas modificaciones o extensiones al algoritmo básico ID3 hasta convertirlo en un sistema denominado C4.5 . A continuación se detallan algunas de estas situaciones que se presentan.

3.2.6.1 LOS ATRIBUTOS CONTINUOS

La definición inicial del ID3 esta restringida atributos que toman un conjunto discreto de valores. En primer lugar porque el atributo objetivo cuyo valor es pronosticado por el árbol debe ser un valor discreto, y segundo porque los atributos probados en los nodos del árbol deben también ser valores discretos, esta segunda restricción puede ser fácilmente eliminada, ya el manejo de valores continuos lo obtenemos, definiendo dinámicamente nuevos valores discretos para dichos atributos, que dividan los valores continuos en un conjunto discreto de

intervalos. En particular, para un atributo "A" que es un valor continuo, el algoritmo puede dinámicamente crear un nuevo atributo booleano "A", que es verdadero si A<c y falso en cualquier otro caso. En estas situaciones el problema estriba en seleccionar el mejor valor para el umbral *c*. Existen diversos criterios que nos permiten hacer esta selección de forma adecuada, dependiendo del escenario que se tenga, pero debido a que la implementación que se presenta en el siguiente capítulo se circunscribe al caso discreto, no se abordarán los fundamentos de los mismos.

3.2.6.2 DATOS FALTANTES

En algunos casos los datos disponibles pueden carecer de algunos valores para ciertos atributos, puede suceder que los valores no hayan sido registrados o que su obtención resulte demasiado costosa, por ejemplo en el dominio médico, si se desea diagnosticar a algún paciente basándose en varias pruebas de laboratorio, puede suceder que el análisis de sangre sólo esté disponible para un subconjunto de pacientes. En esta situación es común estimar los valores faltantes de algún atributo apoyándose en otros ejemplos para los cuales dicho atributo tenga valores conocidos.

Considerando la situación en que $Ganancia(S,A)$ es calculada en el nodo n del árbol de decisión para evaluar si el atributo A es el mejor atributo para probar en este nodo. Suponemos que $x, c(x)$ es uno de los ejemplos de entrenamiento en S y que el valor de $A(x)$ es desconocido, una estrategia para convenir el valor faltante, es asignarle el valor que se presenta más comúnmente entre los ejemplos de entrenamiento en el nodo n, otra alternativa es asignar el valor más común en los ejemplos de entrenamiento en el nodo n que tenga la clasificación $c(x)$. El ejemplo de entrenamiento elaborado usando este valor estimado para $A(x)$ puede ser usado directamente por el árbol de decisión existente.

Otro procedimiento aunque un poco más complejo es asignar una probabilidad para cada uno de los posibles valores de A en vez de dar simplemente el valor mas común para $A(x)$. Estas probabilidades pueden ser estimadas basándose en las frecuencias observadas de varios valores para A entre los ejemplos del nodo n.

3.2.6.3 SOBREAJUSTE O SOBREADAPTACIÓN DE LOS DATOS (OVERFITTING)

Siempre que exista un conjunto grande de hipótesis posibles, hay que tener cuidado en no utilizar la libertad obtenida para encontrar una "regularidad" irrelevante en los datos. A este problema se le conoce como *sobreadaptación*. Se trata de un fenómeno muy generalizado y afecta a todo tipo de algoritmo de aprendizaje, no solo a árboles de decisión[38].

Estas dificultades pueden presentarse cuando hay ruido (errores aleatorios) en los datos, o cuando el número de ejemplos de entrenamiento es demasiado pequeño para producir una muestra representativa. En cualquiera de los dos casos, este algoritmo puede producir un árbol que sobreajuste a los datos de entrenamiento.

Se dice que una hipótesis sobreajusta o sobreadapta el conjunto de entrenamiento, si alguna otra hipótesis que se adecua a los ejemplos de entrenamiento en menor grado, actualmente se desempeña mejor sobre la distribución total de ejemplos (incluyendo las instancias que no pertenecen al conjunto de entrenamiento). A continuación se muestra el impacto del sobreajuste en una aplicación típica de aprendizaje con árboles de decisión (fig.3.5). En este caso el algoritmo ID3 es aplicado en la tarea de aprendizaje de qué pacientes tienen algún tipo de diabetes.

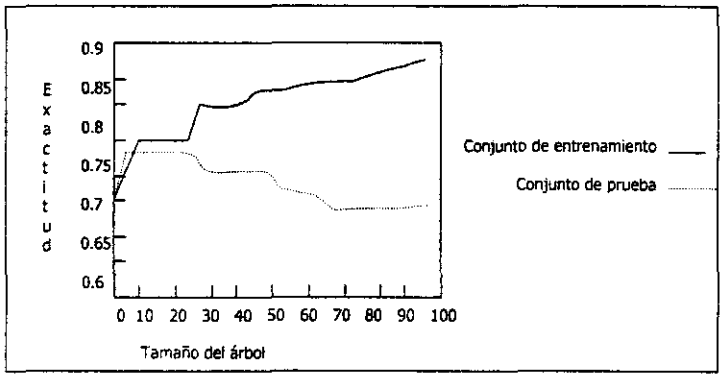

Fig. 3.5 Sobreajuste

El eje horizontal de esta gráfica indica el número total de nodos en el árbol de decisión, cuando el árbol se comienza a construir. La línea continua muestra la exactitud o precisión del árbol de decisión sobre los ejemplos de entrenamiento, mientras que la linea punteada

muestra la exactitud medida sobre un conjunto independiente de ejemplos de prueba (no incluidos en el conjunto de entrenamiento), se puede ver como la exactitud del árbol sobre los ejemplos de entrenamiento incrementa constantemente, mientras el árbol crece, sin embargo la exactitud medida sobre el conjunto de prueba primero incrementa y luego decrece, esto se observa una vez que el árbol excede aproximadamente 25 nodos.

Existen algunos métodos para evitar el sobreajuste y estos pueden agruparse como sigue:

- Los métodos que detienen el crecimiento del árbol, antes de que alcance el punto donde clasifica perfectamente los datos de entrenamiento. (pre-pruning)
- Los métodos que permiten al árbol sobreajustar los datos y posteriormente *podan el árbol* (post-pruning) Quinlan 1987.

3.2.6.4 REDUCCIÓN DEL ERROR PODANDO EL ÁRBOL (PRE-PRUNING)

Se consideran cada uno de los nodos de decisión del árbol como candidatos a ser podados. Podar (prune) un nodo de decisión consiste en remover el subárbol arraigado en ese nodo haciéndolo un nodo hoja y asignándole la clasificación más común de los ejemplos de entrenamiento asociados a ese nodo.

Los nodos son removidos sólo si el resultado del árbol podado desarrolla una mejor solución que el árbol original, sobre el conjunto de validación. Los nodos son podados de manera iterativa, eligiendo siempre el nodo de donde se remueven la mayoría de los incrementos en la exactitud del árbol sobre el conjunto de validación. Se podan los nodos siguientes hasta que el hecho de podar pueda resultar perjudicial para el árbol (es decir decremente la exactitud del árbol sobre el conjunto de validación).

Como se observa en la figura 3.5 la exactitud se mide sobre ambos conjuntos el de entrenamiento y el de prueba, la línea adicional muestra la exactitud sobre el conjunto de prueba a medida que el árbol se va podando. Aquí el conjunto de validación usado para el podado es distinto de los dos conjuntos usados anteriormente, por lo que los datos disponibles han sido divididos en 3 subconjuntos:

[36] Russell, Stuart. Et. al. Inteligencia artificial Un enfoque Moderno. pp.573-581.

- Los ejemplos de entrenamiento
- Los ejemplos de validación usados para podar el árbol
- Un conjunto de ejemplos de prueba, utilizados para proporcionar una estimación imparcial de exactitud sobre futuros ejemplos no vistos.

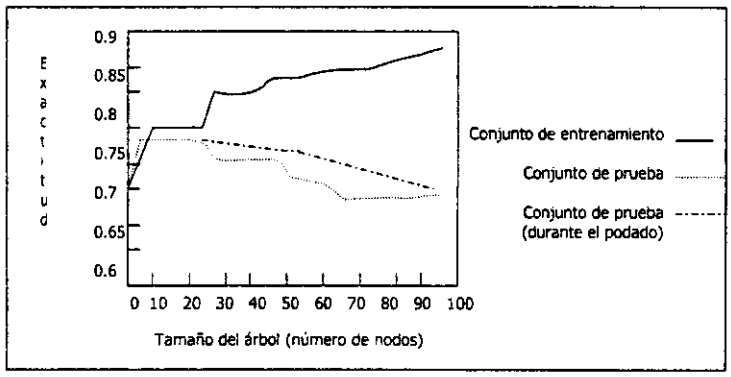

Fig. 3.6 Efecto de reducir el error podando el árbol de decisión

Usar un conjunto diferente de datos para guiar el podado es una aproximación efectiva, siempre y cuando se encuentre disponible una gran cantidad de datos. La principal desventaja de este método es que cuando la cantidad de datos es limitada, el hecho de utilizar algunos de ellos para el conjunto de validación nos lleva a reducir también el conjunto de entrenamiento lo que incide directamente en el desempeño del árbol (fig. 3.6).

A continuación se presenta una alternativa para aplicar el podado, que se ha encontrado útil en muchas situaciones prácticas donde los datos disponibles no son suficientes para obtener todos estos conjuntos.

3.2.6.5 PODAR LAS REGLAS (POST-PRUNING)

Una variante de esta aproximación es usada por C4.5, el post-podado de las reglas considera los siguientes pasos[39]:

1. Inferir el árbol de decisión a partir del conjunto de entrenamiento haciendo crecer el árbol hasta que los ejemplos de entrenamiento son adaptados tan bien como sea posible y permitiendo que el sobreajuste se presente.

[39] Cfr. Mitchell.Tom. Op.cit. pp.71-72.

2. Convertir el árbol obtenido en un conjunto equivalente de reglas, creando una regla para cada trayectoria desde el nodo raíz hasta un nodo hoja.

3. Podar (generalizar) cada regla suprimiendo alguna pre-condición o antecedente que implique mejorar su exactitud estimada.

4. Ordenar las reglas podadas por su exactitud estimada y considerarlas en esta secuencia cuando se clasifiquen los siguientes ejemplos.

Al aplicar este método una regla es generada para cada nodo hoja del árbol. Cada atributo se prueba a lo largo de la trayectoria desde la raiz hasta las hojas, convirtiéndose en el antecedente de la regla (pre-condición), y la clasificación dada en el nodo hoja se convierte en el consecuente (post-condición). Después cada regla es podada quitando algún antecedente o pre-condición cuya supresión no empeore su exactitud estimada. Cualquiera de estos pasos no se realizará si éste reduce la exactitud de las reglas. Un método de estimar la exactitud de las reglas es usar un conjunto de validación de ejemplos que no tenga conexión con el conjunto de entrenamiento.

Algunas ventajas de convertir el árbol a reglas antes de podarlo son las siguientes:

- Convertirlo a reglas nos permite poder distinguir los diferentes contextos en los que un nodo es usado, porque cada trayectoria produce una regla distinta. El podar decisiones considerando que atributo prueba, puede resultar diferente para cada trayectoria, de otra manera, si el árbol fuera podado sólo podria considerar dos opciones: quitar el nodo completamente o retenerlo en su forma original.

- Transformar el árbol en reglas elimina la diferencia entre atributos, ya que prueba que ocurre cerca de la raiz del árbol y lo que pasa en las hojas, de este modo se evita la contabilidad desordenada de descendencia, así como reorganizar el árbol si el nodo raiz es podado(aunque conserve parte del subárbol después de esta prueba).

- Las reglas son frecuentemente más fáciles de entender para las personas.

3.2.7 EVALUACIÓN DE LA EFICIENCIA DE UN ALGORITMO DE APRENDIZAJE.

Se considera que un algoritmo es bueno si produce hipótesis que permitan predecir satisfactoriamente las clasificaciones a las que pertenecen los ejemplos que no han sido vistos anteriormente. Para evaluar la calidad de una hipótesis se verifican sus predicciones en relación con la clasificación correcta una vez que la conocemos[40]. Lo anterior se realizará en un conjunto de ejemplos conocidos como conjunto de prueba. Si para lograr la capacitación se utilizan todos los ejemplos que tenemos, entonces tendremos que buscar otros para efectuar pruebas; debido a lo anterior, frecuentemente es preferible adoptar la siguiente metodología.

1. Reunir una gran cantidad de ejemplos
2. Dividirla en dos conjuntos disyuntos: el conunto de capacitación y el conjunto de prueba.
3. Emplear el arbol de decisión con el conjunto de capacitación como ejemplo de base para producir una hipótesis H.
4. Medir el porcentaje de ejemplos del conjunto de prueba correctamente clasificados.
5. Repetir los pasos del 1 al 4 en conjuntos de capacitación de tamaño diverso, y conjuntos de prueba por cada tamaño escogido aleatoriamente

[40] Cfr.Russell, Stuart. Op.cit. pp.568-569

4. IMPLEMENTACIÓN DEL ALGORITMO ID3

4. IMPLEMENTACIÓN DEL ALGORITMO ID3

4.1 INTRODUCCIÓN

Gran parte de las aplicaciones de inteligencia artificial para funciones prácticas importantes, están basadas en la construcción de modelos de conocimiento utilizados por expertos humanos. En la mayoría de los casos las tareas que un experto desarrolla pueden realizarse a través de la clasificación. Es decir asignando cosas u objetos de estudio a categorías o clases determinadas por sus propiedades.

En este contexto la mayoría de las funciones de la minería de datos, parecen ser un imperativo humano. En función de entender y comunicar acerca del mundo, nosotros constantemente clasificamos, categorizamos y jerarquizamos.

4.2 PLANTEAMIENTO DEL PROBLEMA

Ahora analicemos la siguiente información:

Una empresa planea una estrategia de mercado para lanzar un nuevo modelo de teléfono, y como parte de este proyecto, desea proponer a sus mejores clientes un plan de crédito para que puedan realizar la adquisición y se vean beneficiados con esta nueva tecnología. El problema radica en que una vez definido el criterio de discriminación, se encuentre un método que de manera automática pueda clasificar a los clientes.

En este caso los objetos a clasificar están representados por registros de una base de datos y la acción de clasificar consiste en actualizar cada registro colocando un campo con código que represente algún tipo de clasificación para dicho objeto. **El objetivo es construir un modelo que pueda ser aplicado a un dato no clasificado y pueda categorizarlo o asignarle una clase.**

En este caso la información se obtuvo de una base de datos real, la cual es administrada por el buró de crédito de una empresa líder de telefonía celular en México (cuyo nombre se omite por razones de confidencialidad), dicha base de datos contiene la información de la situación de los clientes de acuerdo al estado de sus cuentas.

❖ EL ACONONDICIONAMIENTO DE LOS DATOS

Como se ilustró en el capítulo 2, el proceso de KDD no sólo se refiere a los algoritmos de minería de datos sino que abarca desde el procesamiento para tener los datos de manera adecuada para minarlos hasta la validación y evaluación del conocimiento adquirido.

En este problema se debe realizar una etapa de acondiconamiento para posteriormente aplicar un algoritmo clasificador. Este procedimiento de selección, extracción y acondicionamiento de los datos fue realizado por personal de la empresa resultando de esta actividad un conjunto de ***1891 registros pre-clasificados***.

En primer lugar debemos de contar con una ***clasificación*** de los clientes en cuestión, cuyo resultado será *aceptado o rechazado*, para entrar en el plan de crédito anteriormente mencionado. Esta clasificación debe realizarse tomando en cuenta los atributos (o característica) que describen a los clientes que en este caso son el objeto a clasificar, y además un criterio que determinará cual será dicha clasificación.

La elección de los atributos que se tomarán en cuenta es muy importante ya que la clasificación correcta de los datos depende en gran medida de que la muestra seleccionada sea realmente representativa. Para lograrlo, el personal del buró de crédito de acuerdo al conocimiento experto que poseen del área, definió qué atributos eran los más significativos. La tabla 4.1 contiene la descripción de los atributos considerados.

El buró de crédito se encarga de establecer también los criterios a tomar en cuenta, para asignar una clasificación. Los datos son enviados a un departamento que de acuerdo a lo establecido discretiza los datos dejando a los atributos con sólo dos posibles y valores (1 y 0) el 1 lo asigna para los casos en que el atributo está dentro de los parámetros de "aceptar" y 0 en el caso contrario.

Una vez asignada la calificación para cada cliente de aceptado (A) o rechazado (R) que representa la clasificación para cada caso (esta clasificación se determina mediante la aplicación de un programa de cómputo que evalúa los atributos y de acuerdo a los rangos definidos establece dicha clase) los datos quedan como se aprecia en la tabla 4.2.

NOMBRE	DESCRIPCION	APROBAR	RECHAZAR
Pvencido	Número de cuentas a Plazos con saldo vencido> 2000	Si no tiene ninguna	Si tiene 1 o más
Hvencido	Número de cuentas de Hipotecario con saldo vencido >2000	Si no tiene ninguna	Si tiene 1 o más
Pmop	Peor MOP (indice de morosidad) histórico en cuentas a Plazos	Si tiene un indice de 00 03	Si tiene un indice de 04 a 99
Hmop	Peor MOP (indice de morosidad) histórico en cuentas de Hipotecario	Si tiene un indice de 00 05	Si tiene un indice de 06 a 99
Pnomoroso	Meses que han pasado desde que fue moroso en cuentas a Plazos	Número de meses >12	De 0 a 12 meses
Hnomoroso	Meses que han pasado desde que fue moroso en cuentas de Hipotecario	Número de meses > 6	De 0 a 6
Cuenta_i	Cuentas a Plazos o de Hipotecario investigadas.	Si son <=4	Si son >4
Cuenta_c	Cuentas a Plazos o de Hipotecario canceladas.	Si son 0	Si son >0

Tabla 4.1 Descripción de los atributos

Pvencidos	Hvencidos	Pmop	Hmop	Pmoroso	Hmoroso	Cuenta_i	Cuenta_c	Clase
1	1	1	1	1	1	1	1	A
1	1	1	1	1	1	1	1	A
1	1	1	1	0	1	1	1	R
1	1	0	1	1	1	1	1	R
1	1	1	1	1	1	1	1	A
1	1	0	1	0	1	1	0	R

Tabla 4.2 Clientes clasificados[41]

4.3 EVALUACIÓN DEL ESCENARIO DESCRITO

- Es importante recordar que la clasificación, es una función de la minería de datos que esta intrínsecamente ligada al aprendizaje supervisado.

- Debido a que se cuenta con un grupo de datos previamente clasificados podemos contar con un conjunto de entrenamiento.

[41] Existe además una columna al inicio donde se encuentra el número de identificación de cada cliente pero para este ejemplo no se requiere, y por cuestiones de confidencialidad ha sido eliminada.

- Los datos con los que se cuenta son discretos, situación que favorece la utilización de algoritmos clasificadores como los árboles de decisión. Ya que si bien los árboles pueden trabajar también con atributos con valores continuos, para esos casos se requiere incluir algún método apropiado para su tratamiento.

- En este caso la cantidad de datos con la que se cuenta es lo suficientemente grande como para dividirla en dos grupos y de esta manera contar con un conjunto de prueba que nos permita evaluar el desempeño del algoritmo.

Tomando en consideración las características del problema así como también las características de los diferentes algoritmos clasificadores que se describen en la segunda sección de este trabajo, se decidió construir un árbol de decisión basado en el algoritmo ID3 para obtener un modelo que ajuste al conjunto de entrenamiento, y lograr así una generalización para clasificar a las instancias desconocidas.

4.4 OBJETIVO DEL SISTEMA PROGRAMADO

El sistema construye un árbol de decisión que representa la relación existente entre la conclusión-decisión y sus atributos. Es decir, se produce un proceso de generalización de forma que el árbol de decisión obtenido clasifica correctamente los ejemplos dados. La finalidad de realizar esta implementación es ejemplificar la aplicación de un algoritmo de minería de datos, que es la parte medular del proceso de descubrimiento de conocimiento (KDD).

4.5 DESARROLLO

El ID3 es un algoritmo simple y, sin embargo, potente, cuya misión es la elaboración de un árbol de decisión, los fundamentos teóricos del mismo fueron explicados detalladamente en el capítulo 3 de este trabajo, se eligió este algoritmo por ser un prototipo básico de los árboles de decisión.

4.5.1 CONSTRUCCIÓN DEL PROGRAMA CLASIFICADOR

4.5.1.1 LA PROGRAMACIÓN

❖ **Programación Procedimental**

Con la programación procedimental se puede combinar las secuencias de instrucciones repetibles en un solo lugar. Una *llamada de procedimiento* se utiliza para invocar al procedimiento. Después de que la secuencia es procesada, el flujo de control procede exactamente después de la posición donde la llamada fue hecha. Al introducir *parámetros* así como procedimientos de procedimientos (*subprocedimientos*) los programas ahora pueden ser escritos en forma más estructurada y libres de errores. Por ejemplo, si un procedimiento ya es correcto, cada vez que es usado produce resultados correctos. Por consecuencia, en caso de errores, se puede reducir la búsqueda a aquellos lugares que todavía no han sido revisados. De este modo, un programa puede ser visto como una secuencia de llamadas a procedimientos.

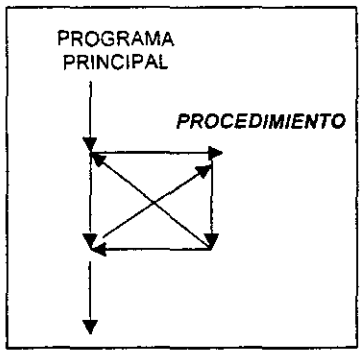

Figura 4.1 Flujo de control en la ejecución de procedimientos.

El programa principal es responsable de pasar los datos a las llamadas individuales, los datos son procesados por los procedimientos y, una vez que el programa ha terminado, los datos resultantes son presentados. Así, el *flujo de datos* puede ser ilustrado como se muestra en la figura 4.1. Para resumir: tenemos ahora un programa único que se divide en pequeñas piezas llamadas procedimientos(ver fig. 4.2). Para posibilitar el uso de procedimientos generales o grupos de procedimientos también en otros programas, aquéllos deben estar disponibles en forma separada. Por esa razón, la programación modular permite el agrupamiento de procedimientos dentro de módulos.

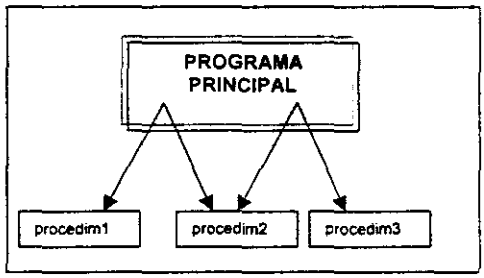

Figure 4.2 Programación Procedimental. El programa principal coordina las llamadas a procedimientos y pasa los datos apropiados en forma de parámetros.

❖ **Programación Modular**

En la programación modular, los procedimientos con una funcionalidad común son agrupados en *módulos* separados. Un programa por consiguiente, ya no consiste solamente de una sección. Ahora está dividido en varias secciones más pequeñas que interactúan a través de llamadas a procedimientos y que integran el programa en su totalidad.

Cada módulo puede contener sus propios datos. Esto permite que cada módulo maneje un *estado* interno que es modificado por las llamadas a procedimientos de ese módulo. Sin embargo, solamente hay un estado por módulo y cada módulo existe cuando más una vez en todo el programa. Lo anterior se ilustra en la figura 4.3 :

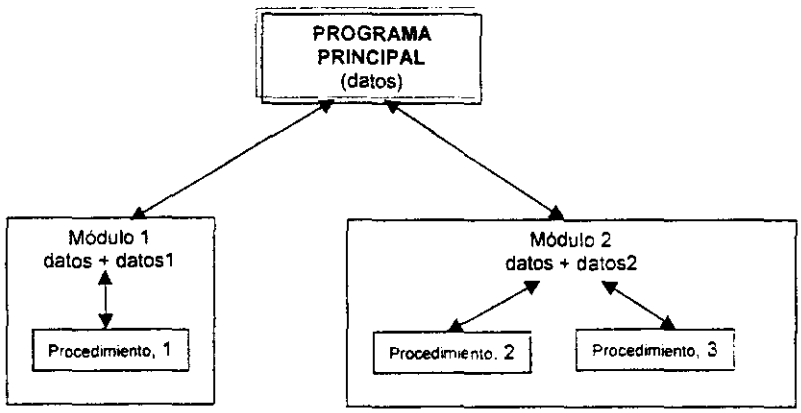

Figura 4.3 Programación Modular. El programa principal coordina las llamadas a procedimientos en módulos separados y pasa los datos apropiados en forma de parámetros.

4.5.1.2 EL LENGUAJE C

Desarrollado en la última parte de los 1970s, C se constituyó en un enorme éxito debido al desarrollo de UNIX que fue escrito casi por completo en este lenguaje. En contraste con otros lenguajes de alto nivel, C fue escrito por programadores para programadores.

Debido a que C es un lenguaje procedimental, permite la definición de *funciones*. Los procedimientos son "simulados" por funciones que no regresan "ningún valor". Este valor es un tipo especial llamado void. Las funciones se declaran en forma similar a las variables, pero aquéllas encierran sus argumentos entre paréntesis (aún si no llevan argumentos. los paréntesis deben ser especificados). Otra característica importante es que permite crear y mantener una biblioteca de funciones propia, que puede ser usada en muchos programas diferentes. Además dado que permite la compilación separada se puede trabajar fácilmente en grandes proyectos y minimizar la duplicidad de esfuerzos.

4.6 CARACTERÍSTICAS DEL PROGRAMA REALIZADO

Fue desarrollado en lenguaje C utilizando una programación modular y presenta la siguiente estructura:

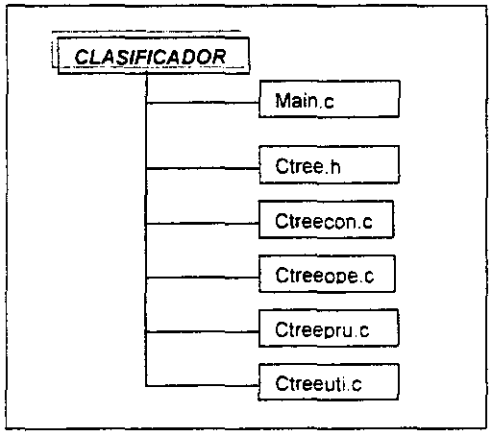

Figura 4.4. Estructura del programa *Clasificador*

Main contiene el programa principal y en él se realizan las llamadas a los diferentes módulos, esta función recibe como parámetros 3 argumentos que son los archivos de datos que procesará, esto se explica más a detalle en la parte de los datos.

Ctree.h Contiene las definiciones de los datos y estructuras empleadas en la implementación del árbol. A lo largo de los diferentes módulos es requerida como librería. (#include "ctree.h").

Ctreecon.c Contiene las definiciones de las funciones para la construcción del árbol.

Ctreeope.c Contiene la definición de las operaciones requeridas.

Ctreepru.c Aquí se encuentran contenidas las funciones para el podado del árbol.

Ctreeuti.c Se refiere a las funciones y definiciones útiles para la lectura de datos.

Ver código en el apéndice A.

El programa el ejecutarse, recibe tres parámetros desde la linea de comando, estos parámetros son los archivo de datos (formato *.txt) que requiere para hacer la clasificación:

a) Un archivo que contenga los parámetros del conjunto de entrenamiento, es decir, número de ejemplos, número de atributos y número de términos que toma cada atributo.

b) Un archivo que contenga al conjunto de entrenamiento. Para esta aplicación se eligieron varias muestras de diferentes proporciones para realizar el entrenamiento, la elección fue hecha al azar, para observar los resultados que se generan en cada caso.

c) Un archivo que contenga al conjunto de prueba. Este archivo contiene registros cuyo número varia para cada caso y dicho conjunto es independiente de los datos de entrenamiento.

Una vez que el se obtiene el árbol, *el programa lo traduce a reglas* como se describe en el capitulo dos. de tal manera que la salida del programa es el conjunto de reglas correspondiente al árbol de decisión obtenido.

El sistema clasifica el archivo de prueba (para ello no toma en cuenta la última columna que es la contiene la clasificación asignada previamente) y posteriormente compara la clasificación que obtuvo con la de la columna de clasificación del archivo, para tener un punto de referencia y finalmente calcular el número de ejemplos bien clasificados.

Estos resultados son alojados en un archivo modo texto que el sistema genera (ver apéndice A)

- ***Clasificadores.txt*** : este archivo contiene las reglas obtenidas.

4.7 RESULTADOS OBTENIDOS

En la tabla 4.3 se reportan los resultados obtenidos por el programa implementado.

Base de datos de Cía. Telefónica	No. de ejemplos de entrenamiento	No. de registros de prueba	No. de registros bien clasificados	Desempeño del programa
Primera Muestra	42	27	24	88.88%
Segunda Muestra	100	891	855	95.95%
Tercera Muestra	1000	891	873	97.97%

Tabla 4.3 Reporte de resultados

4.7.1. DISCUSIÓN DE RESULTADOS

Para analizar el comportamiento del algoritmo en diferentes condiciones se seleccionaron 3 muestras con diferentes proporciones para el conjunto de entrenamiento y el conjunto de prueba respectivamente (ver tabla 4.3).

En el primer caso tenemos un conjunto de entrenamiento demasiado pequeño, por lo que la generalización no se realizó adecuadamente. Para la segunda muestra los datos de entrenamiento fueron 100 y el conjunto de prueba aumentó a 891, aquí podemos observar como un ligero aumento en los datos de entrenamiento repercutió notablemente en el número de resultados bien clasificados, a pesar de que el conjunto de prueba fue considerablemente más grande que en ejemplo anterior.

El último caso ejemplifica muy cercanamente lo que podemos tener en la realidad, ya que si bien nuestra muestra de entrenamiento es grande, también lo es la cantidad de registros que deseamos clasificar automáticamente.

Se puede observar como el número de ejemplos bien clasificados aumentó en relación al segundo caso. Aunque el conjunto de prueba fue el mismo, la muestra de entrenamiento aumentó, lo que nos permite observar cómo el número de ejemplos de entrenamiento incide directamente en el número de instancias bien clasificadas.

Comparando los dos últimos escenarios, se puede advertir que si bien el desempeño mejora con un conjunto de entrenamiento más grande, el modelo que se obtiene tomando sólo 100 datos nos permite tener una buena generalización y este hecho también puede relacionarse con los factores que influyen para tener un buen ajuste de los datos, ya que para esta aplicación los datos con que se cuentan no contienen ruido, y además los atributos considerados son 9 y éstos a su vez sólo presentan dos valores en la salida.

4.7.2 EL PODADO

En este punto es importante tener en cuenta que existe una relación entre la dimesionalidad de un problema la complejidad del modelo y el número de ejemplos que son necesarios para un aprendizaje apropiado, ya que puede presentarse el sobrajuste (overfitting).

El podado (prunning) es una solución para el problema del sobreajuste de lo cual se habla en capítulos anteriores (2 y 3), existen diversos factores que pueden provocar este fenómeno, *aunque no siempre se presenta*.

La cantidad de datos requerida es afectada por factores como los números de propiedades y clases y la complejidad del modelo de clasificación. A medida que esto incremente se necesitarán más casos de entrenamiento (cientos o incluso miles) para construir un modelo confiable.

Debido a que se implementó el módulo de podado, antes de ver como se comportaba el algoritmo sobre los datos, una vez que el algoritmo se probó en las diferentes muestras se pudo observar que dadas las características del problema que ya se mencionaron, no hubo sobreajuste, por lo cual no se podó ninguna regla. Esto pudo observarse porque al comparar las reglas obtenidas antes de podar eran las mismas que las del conjunto de reglas post-podado.

5. CONCLUSIONES

5. CONCLUSIONES

A lo largo de este trabajo se ha visto como el concepto de dato ha sido ampliado al de conocimiento (objetos, relaciones, hechos, reglas, etc.) y los sistemas de simple procesamiento de datos se han convertido en una búsqueda inteligente para encontrar vías de solución en el conocimiento (figura 5.1). En este sentido el descubrimiento de conocimiento surge, por lo tanto, del desarrollo necesario del procesamiento de datos.

"Se dice que el hombre será tan inteligente como conocimientos posea y como capacidad tenga de superar situaciones nuevas aplicando dicho conocimiento con experiencia y sagacidad" [42].

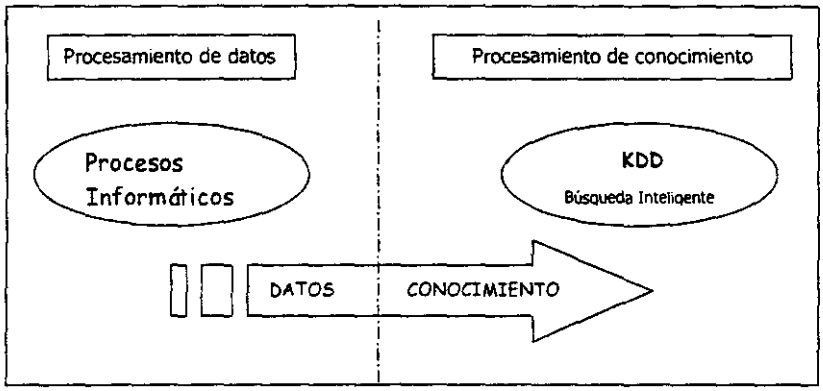

Figura 5.1 Transición de los datos al conocimiento.

El hecho de qué se requiera contar con información de calidad para *extraer conocimientos valiosos y potencialmente útiles* que incidan directamente en eventos futuros que beneficien a las organizaciones, ha permitido que *la tecnología vaya pasando de las universidades y centros de investigación, a los clientes de diversos sectores empresariales e industriales*, ya que la base de gran parte de la tecnologia de descubrimiento de conocimiento se deriva de la investigación en la Inteligencia Artificial y del aprendizaje automático. Pienso que uno de los factores que más ha influido para que se dé esta coyuntura es la mercadotecnia que rodea al descubrimiento del conocimiento en las bases de datos, que si bien no se ha dado conocer como tal, si se ha introducido al mundo de los negocios a través de conceptos como *minería de datos* (data mining) y almacenamiento de datos (data warehousing).

[42] Nebendahl, Dieter. Sistemas Expertos. p.18.

Debido a que el descubrimiento de conocimiento es un proceso conducido por los datos pueden presentarse problemas debido a la **calidad de los datos**, pues en las situaciones reales existen datos erróneos o incompletos, redundantes, y desde luego bases de datos exorbitantemente grandes. Una forma de garantizar su calidad es usar las técnicas de minería adecuadas analizando cuidadosamente las anomalías, las investigaciones en este aspecto están encaminadas a los siguientes problemas:

Alta dimensionalidad de los datos: No solo existen bases de datos enormes con un gran número de registros, sino que también tienen demasiados campos (atributos, variables etc) por lo que el problema de la dimensionalidad es considerable. Una alta dimensionalidad genera problemas en los conjuntos de datos en términos del tamaño del espacio de búsqueda para un modelo de inducción, lo cual incrementa la posibilidad de que el patrón generado por un algoritmo no sea válido en lo general.

El sobreajuste: aunque en la implementación del ID3 se presentó una solución para este problema, existen aún situaciones que deben resolverse a este respecto.

La constante variación en los datos: La rápida variación de los datos (no son estacionarios), puede hacer que un patrón se invalide en poco tiempo, posibles soluciones pueden ser métodos de actualización para los patrones, tratando este cambio constante cono una oportunidad para el descubrimiento de conocimiento.

Datos faltantes o ruidosos: este problema es especialmente agudo en bases de datos de negocios. Posibles soluciones se investigan en sofisticadas técnicas estadísticas.

Patrones más entendibles: En muchas aplicaciones es importante hacer el descubrimiento más comprensible para los humanos, algunas soluciones incluyen representación gráfica, estructuración de reglas, generación de lenguaje natural y técnicas para visualización de datos y estrategias de refinamiento de reglas.

Integración con otros sistemas: Un sistema de descubrimiento stand-alone puede no ser muy útil, generalmente esta ligado a herramientas como DBMS, hojas de cálculo, visualización y además puede integrarse a sistemas basados en conocimiento tales como Sistemas expertos.

- Una conclusión importante que podemos derivar de este trabajo es que gracias ala mercadotecnia que se ha generado a su alrededor, el KDD (Descubrimiento de conocimiento en bases de datos) nos brinda la oportunidad de establecer ese difícil y en ocasiones imposible vínculo de la investigación científica con el sector empresarial, lo cual debe aprovecharse al máximo.

- Una de las principales ventajas de algoritmo ID3 es que nos permite obtener el conocimiento en forma de reglas lo cual nos facilita su integración con otros sistemas, por ejemplo este clasificador podria alimentar a una base de conocimientos.

- También es importante considerar las limitaciones del ID3 ya que su éxito depende en gran parte del tipo de datos que se tengan, y de las características del problema a resolver, por ejemplo si las clases no esta bien definidas, y los atributos toman valores continuos, entonces tal vez convendria utilizar una red neuronal en lugar de un árbol de decisión. Por lo anterior es conveniente que se realice un análisis cuidadoso del problema antes de elegir el algoritmo de mineria de datos que utilizaremos.

- Los algoritmos de la mineria de datos son de gran utilidad cuando *no se cuenta con el conocimiento suficiente* de alguna área y si el conocimiento experto de un área especifica esta disponible, el KDD nos permite *enriquecer ó validar dicho conocimiento* y de esta manera contar con *bases de conocimiento más completas.*

- Es importante el estudio del Descubrimiento de Conocimiento en Bases de Datos ya que el KDD está creciendo rápidamente con la expectativa de ser un gran campo de aplicación. El descubrimiento de conocimiento pretende ser la nueva tecnología en bases de datos en los próximos años. La necesidad de herramientas para el descubrimiento automatizado, ha causado un enorme crecimiento en el número y tipo de herramientas disponible comercialmente y en el dominio público y debido a la aplicación potencial del descubrimiento del conocimiento, en diversas áreas hay un crecimiento de oportunidades en la investigación sobre este campo.

BIBLIOGRAFÍA

Agrawal, Rakesh, et.al. "An Interval Classifier for Database Mining Applications". San José, CA. IBM Almaden Research Center. 1992.

Ash, Robert. Information Theory, US, Interscience Publisher John Wiley & Sons Inc, 1965.

Berry, Michael. y Linoff Gordon. Data Mining Techniques, for marketing, sales and customer support. USA. John Wiley. 1997.

Carbonell, Jaime. Machine Learning: paradigms and methods.The MIT Press.

Cortés Garcia Ulises et. al. Inteligencia Artificial. Barcelona, España, Ediciones UPC,1993.

Cover, Thomas et. al. Elements of Information Theory. US,John Wiley & Sons Inc, 1991.

Decker, Karsten et. al.Technology Overview: A Report on Data Mining.CSCS-ETH,Swiss Scientific Computing Center,1995.

Derlin, Barry. Data Warehouse From Architectura to Implementation. USA, Addison Wesley, 1997.

Fayyad, U.M., et. al. "In Knowledge Discovery and Data Mining Towards a Unifying Framework." (KDD-96), AAAI Press,1996.

Gil Pechuán, Ignacio. Sistemas y Tecnologías de la Información para la Gestión. España, McGraw-Hill,1997.

Harjinder, Gill.et. al. Data warehousing, la integración de la información para la mejor toma de decisiones. México, Prentice Hall Hispanoamericana,1996.

Hutchinson, Alan. Algorithmic Learning. Oxford University Press, 1994.

Korth, Henry. et. al. Fundamentos de Bases de datos. Segunda Edición, McGraw-Hill, 1993.

Kroenke,David. Procesamiento de Bases de Datos, Fundamentos Diseño e Instrumentación, México, Prentice Hall Hispanoamericana, 1996.

Mitchell, Tom. Machine Learning. USA, McGraw-Hill, 1997.

Nebendahl, Dieter. Sistemas Expertos Introducción a la técnica y aplicación. Barcelona España.Editorial Marcombo,1988.

Quinlan, J. Ross. C4.5: Programs for machine learning. California, Morgan Kaufmann Publishers,1993.

Rusell,Stuart y Norvig, Peter. Inteligencia Artificial. Un enfoque Moderno. México, Prentice Hall Hispanoamericana,1996.

Star,Ralph. Principles of Information Systems a Managerial Approach. USA,Florida State University, CTI, 1996.

Singh J. Ideas Fundamentales sobre la Teoría de la Información del lenguaje y de la Cibernética. España,Alianza Editorial, 1982.

Scott, George. y Cohen, Daniel. Sistemas de Información. México, McGraw-Hill, 1996.

CASO 1: 42 ejemplos de entrenamiento y 27 en el conjunto de prueba.
Bien clasificados: 24

(Pmop = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 1) && (Pvencidos = 0) && ===> Clase = A)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 1) && (Pvencidos = 1) && (Hvencidos = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hmop = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hmop = 1) && (Hnomoroso = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hmop = 1) && (Hnomoroso = 1) && (Cuenta_i = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 0) && (Cuenta_c = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hmop = 1) && (Hnomoroso = 1) && (Cuenta_i = 1) && ===> Clase = A)

(Pmop = 1) && (Pnomoroso = 1) && (Hnomoroso = 0) && ===> Clase = R)

(Pmop = 1) && (Pnomoroso = 1) && (Hnomoroso = 1) && ===> Clase = A)

CASO 2: 100 ejemplos de entrenamiento y 891 en el conjunto de prueba. Bien clasificados: 855

(Pnomoroso = 0) && (Pmop = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 1) && (Pvencidos = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 1) && (Pvencidos = 1) && (Cuenta_c = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 1) && (Pvencidos = 1) && (Cuenta_c = 1) && (Hvencidos = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 1) && (Pvencidos = 1) && (Cuenta_c = 1) && (Hvencidos = 1) && (Hnomoroso = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 1) && (Pvencidos = 1) && (Cuenta_c = 1) && (Hvencidos = 1) && (Hnomoroso = 1) && (Cuenta_i = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Hmop = 1) && (Pvencidos = 1) && (Cuenta_c = 1) && (Hvencidos = 1) && (Hnomoroso = 1) && (Cuenta_i = 1) && ===> Clase = A)

(Pnomoroso = 1) && (Pmop = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 1) && ===> Clase = A)

CASO 3: 1000 ejemplos de entrenamiento y 891 en el conjunto de prueba.
Bien clasificados: 873

(Pnomoroso = 0) && (Pmop = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 1) && (Hvencidos = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hnomoroso = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 1) && (Hmop = 0) && ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 1) && (Hmop = 1) && (Cuenta_i = 0) ===> Clase = R)

(Pnomoroso = 0) && (Pmop = 1) && (Pvencidos = 1) && (Hvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 1) && (Hmop = 1) && (Cuenta_i = 1) ===> Clase = A)

(Pnomoroso = 1) && (Pmop = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 1) && (Hnomoroso = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 1) && (Hmop = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 1) && (Hmop = 1) && (Cuenta_i = 0) && ===> Clase = R)

(Pnomoroso = 1) && (Pmop = 1) && (Hvencidos = 1) && (Pvencidos = 1) && (Hnomoroso = 1) && (Cuenta_c = 1) && (Hmop = 1) && (Cuenta_i = 1) && ===> Clase = A)

/* Archivo: main.c */

```
#include <stdio.h>
#include "ctree.h"

 extern void lee_datos(char*);
 extern void lee_datos_prueba(char*);
 extern void lee_parametros(char*);
 extern int crisp_crea_arbol(struct Nodo**,struct Nodo*,int);
 extern void imprime_arbol(struct Nodo*,FILE*,int);
 extern FILE* abre_archivo(char*);
 extern void imprime_ejemplos(void);
 extern void imprime_terminos(void);
 extern void genera_reglas(void);
 extern int imprime_reglas(struct Nodo *,FILE*);
 extern int imprime_lista_reglas(char*,struct condicion**);
 extern int imprime_clasificador(char*,struct condicion**);
 extern void post_pruning();
 extern void imprime_exactitud(struct regla_ex*);
 extern float desempeño_reglas(int);
 extern int cuenta_bien_clasificados(int);


void main(int argc, char *argv[])
{
 int i,j;
 FILE *parchivo;
 FILE *reglas;

 if(argc < 4)
 {
  printf("\n\t Uso: arbold archivo_parametros archivo_datos arch prueba\n");
  exit(1);
 }
 ARCHIVO = fopen("ejemplos_regla.txt","w");
 lee_parametros(argv[1]);
 lee_datos(argv[2]);
 lee_datos_prueba(argv[3]);
 parchivo = abre_archivo("resulta.dos");
 reglas = abre_archivo("reglas.txt");
 crisp_crea_arbol(&P_raiz,P_raiz,0);
 imprime_reglas(P_raiz,reglas);
 imprime_arbol(P_raiz,parchivo,0);
 printf("\n Num_hojas: %d\n",Num_hojas);
 genera_reglas();
 imprime_lista_reglas("listareglas.txt",Reglas);
 imprime_clasificador("clasificadores.txt",Clasificadores);
 post_pruning();
 imprime_clasificador("listapodado.txt",Clasificadores);
 printf("\nBien clasificados  : %d",cuenta_bien_clasificados(PODADO));
}
```

```
/* Archivo: ftreeuti.c
 Finalidad: Funciones útiles para la lectura de datos de los archivos de datos y de definición.*/

#include "ctree.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

  extern float crisp_ganancia(int*,int);
  int tipo_dato(int);

float* arreglo_float(int entradas)    /* Crea arreglo de flotantes.*/
{
      float *pf;
      pf = (float*)malloc(entradas * sizeof(float));


if(pf == NULL)
          {
          printf("\n arreglo_float: Memoria insuficiente.");
           exit(1);
          }
     return pf;
}

int* arreglo_int(int entradas) /* Crea arreglo de enteros.*/
{
       int *pi;
       pi = (int*)malloc(entradas * sizeof(int));
       if(pi == NULL)
         {
         printf("\n arreglo_int: Memoria insuficiente.");
         exit(1);
         }
       return pi;
}


char* arreglo_char(int tamaño) /* Crea arreglo de caracteres.*/
{
         char *pc;
          pc = (char*)malloc(tamaño * sizeof(char));
         if(pc == NULL)
          {
          printf("\n arreglo_char: Memoria insuficiente.");
          exit(1);
          }
         return pc;
}

/* Crea arreglo para guardar los datos, es decir, el conjunto de entrenamiento.*/

cval * crea_arreglo_datos(int natributos, int nejemplos)
{
    cval  *pa;
    int i, j, k;
    int tamaño;
    tamaño = nejemplos * natributos;
    pa = (union valor*)malloc(tamaño * sizeof(union valor));

         if(pa == NULL)
         {
         printf("\n crea_arreglo_datos: No hay suficiente memoria.");
```

```
        exit(0);
        }
    return pa;
}

/* Devuelve el valor de un atributo de un elemento del conjunto de entrenamiento */

cval  dev_val_atributo(int ejemplo,int atributo)
        {
        if(ejemplo > Num_ejemplos || atributo > Num_atributos)
                {
         printf("\n\tdev_val_atributo: Atributo o Ejemplo inexistente.");
        exit(1);
        }
 return Arreglo_datos[Num_atributos*ejemplo-Num_atributos+atributo-1];
}


cval  dev_val_atributo_T(int ejemplo,int atributo)
        {
        if(ejemplo > Num_ejemplos_T || atributo > Num_atributos_T)
        {
        printf("\n\tdev_val_atributo_T: Atributo o Ejemplo inexistente.");
        exit(1);
        }
 return Arreglo_datos_T[Num_atributos_T*ejemplo-Num_atributos_T+atributo-1];
}

/* Devuelve el valor de la clasificación de un ejemplo con respecto al termino de la variable de  clasificacion. */

cval  dev_val_clasificacion(int ejemplo)
{
 return dev_val_atributo(ejemplo,Num_atributos);
}

/*  Crea el arreglo que contiene los nombres de los atributos que describen a los elementos bajo estudio. */

 struct C_atributo* crea_arreglo_atributos(int natributos)
        {
         C_atributo *pa;
        int i;
        pa = (struct C_atributo*)malloc(natributos * sizeof(struct C_atributo));
        if(pa == NULL)
                {
                printf("\n crea_arreglo_atributos: Memoria insuficiente.Aqui");
                exit(0);
                }
        return pa;
        }

 /* Devuelve el nombre del atributo */

 char* dev_nombre_atrib(int atributo)
        {
        if(atributo > Num_atributos)
                {
                printf("\n\t dev_nombre_atrib: Atributo solicitado es mayor que el numero de ellos");
                exit(1);
                }
        return Arreglo_atributos[atributo - 1].nombre;
        }

/* Obtiene el numero de términos de un atributo.*/
```

```
int terminos_atributo(int atributo)
        {
         return Arreglo_atributos[atributo - 1].num_terminos;
        }


/* Crea el arreglo de struct  terminos */

struct Termino** crea_arreglo_terminos(int numatributos)
        {
        struct Termino** pt;
        int nterminos;
        int i;
        int j;
         pt = (struct Termino**)malloc(numatributos * sizeof(struct Termino*));
                  if(pt == NULL)
                           {
                            printf("\n crea_arreglo_terminos: Memoria insuficiente.");
                           exit(1);
                           }
 for(i = 1; i <= numatributos; i++)
        {
         nterminos=terminos_atributo(i);
        pt[i - 1] = (struct Termino*)malloc(nterminos * sizeof(struct Termino));
                  if(pt[i - 1] == NULL)
                  {
                   printf("\n crea_arreglo_terminos: Memoria insuficiente.");
                   exit(1);
                  }
        }
        return pt;
        }

int tipo;
        pa=fopen(arch,"r");
                  if(pa == NULL)
                           {
                            printf("\n lee_parametros: No fue posible abrir archivo de parametros.");
                           exit(1);
                            }
        fscanf(pa,"%d",&Num_ejemplos);
         getc(pa);                          /* lee salto de linea */

        fscanf(pa,"%d",&Num_atributos);
        getc(pa);                           /* lee salto de linea */

          fscanf(pa,"%d",&Num_terminos);
        getc(pa);

 Arreglo_atributos = crea_arreglo_atributos(Num_atributos);
         for(i = 0; i < Num_atributos; i++)
                  {
                   fscanf(pa,"%s",&Arreglo_atributos[i].nombre);
                   Arreglo_atributos[i].nombre[MAX_LON_NAT]='\0';
                   getc(pa);
                   fscanf(pa,"%d",&Arreglo_atributos[i].num_terminos);
                   getc(pa);
                   fscanf(pa,"%d",&Arreglo_atributos[i].tipo);
                   getc(pa);
                    Arreglo_atributos[i].utilizado = 0;
                    }

 Terminos = crea_arreglo_terminos(Num_atributos);
 for(i = 0; i < Num_atributos; i++)
```

```
        {
        fscanf(pa,"%d",&atributo);
        getc(pa);
        fscanf(pa,"%d",&nterminos);
        getc(pa);
          tipo = tipo_dato(atributo);
          for(j = 0; j < nterminos; j++)
                {
                Terminos[i][j].atributo = atributo;
                 fscanf(pa,"%c",&Terminos[i][j].val.cval);
                 getc(pa);
                }
        }
```

/* Funcion que lee el archivo de datos. El formato del archivo plano es como sigue:

numero_de_atributos
numero_de_ejemplos
vat11,vat12,...,vat1n,y1
vat21,vat22,...,vat2n,y2
.Donde atributoi es el nombre del atributo i, vatij es el valor del atributo j para el ejemplo i y yi es la clasificacion del ejemplo i. */

```
void lee_datos(char* arch)
{
 FILE *pa;
 int i,j;
 int atributo =0;
 pa=fopen(arch,"r");

 if(pa == NULL)
        {
         printf("\n lee_elementos: No fue posible abrir archivo de datos.");
        exit(0);
        }
 printf("\n\t Leyendo archivo de datos %s...",arch);
 fscanf(pa,"%d",&Num_atributos);
 getc(pa);                    /* lee salto de linea */
 fscanf(pa,"%d",&Num_ejemplos);
 getc(pa);

 Arreglo_datos = crea_arreglo_datos(Num_atributos,Num_ejemplos);
        if(Arreglo_datos == NULL)
        {
          printf("\n lee_datos: Arreglo de datos NULO.");
        exit(0);
        }
 printf("\n\t Arreglo de datos creado...");
        for(i = 0; i < Num_atributos * Num_ejemplos; i++)
        {
         atributo = (i % Num_atributos) + 1;
        fscanf(pa,"%c",&Arreglo_datos[i].cval);
         getc(pa);
        }
}
 }


 void lee_datos_prueba(char* arch)
 {
 FILE *pa;
 int i,j;
 int atributo =0;
 pa=fopen(arch,"r");
```

```
            if(pa == NULL)
            {
            printf("\n lee_elementos: No fue posible abrir archivo de datos.");
            exit(0);
            }
  fscanf(pa,"%d",&Num_atributos_T);
  getc(pa);                    /* lee salto de linea */
  fscanf(pa,"%d",&Num_ejemplos_T);
  getc(pa);

  Arreglo_datos_T = crea_arreglo_datos(Num_atributos_T,Num_ejemplos_T);
            if(Arreglo_datos_T == NULL)
            {
            printf("\n lee_datos: Arreglo de datos NULO.");
             exit(0);
            }
  printf("\n\t Arreglo de datos creado...");
            for(i = 0; i < Num_atributos_T * Num_ejemplos_T; i++)
            {
            atributo = (i % Num_atributos_T) + 1;
            fscanf(pa,"%c",&Arreglo_datos_T[i].cval);
            getc(pa);
            }
}
 }

 float log2(float f)
{
 float lf;
            if(f == 0.0)
             return 0.0;
           else
            lf= log10(f)/log10(2.0);
           return lf;
}



 int atributo_probado(int atributo,struct Nodo *pn)
 {
  struct Nodo *paux;
  int i = 0;
           do
           {
            pn = pn->padre;
             paux = pn;
            if(paux->atributo == atributo)
             return 1;
            }while(paux != P_raiz);
   return 0;
 }


int num_hijos(int atributo,struct C_atributo* atributos)
{
 return atributos[atributo - 1].num_terminos;
}

/* Marca a un atributo como ya utilizado.*/

void marca_atributo(struct C_atributo* pc,int atributo)
{
 pc[atributo - 1].utilizado = 1;
```

```
}
void destruye_arbol(struct Nodo *raiz)
{
 int i;
 for(i = 0; i < raiz->num_hijos; i++)
 destruye_arbol(raiz->arreglo_hijos[i]);
 raiz->padre = NULL;
 if(raiz->tipo == 1)
  free(raiz->arreglo_hijos);
 free(raiz->pNk);
 free(raiz->N_pertenencia);
 free(raiz);
}

/* Abre archivo de salida*/

 FILE* abre_archivo(char *archivo)
{
 FILE *farch;
 farch = fopen(archivo,"w");
          if(farch == 0)
           {
          printf("No se pudo abrir archivo de salida");
           exit(1);
           }
  }

/* Imprime el arbol generado. */

 int imprime_arbol(struct Nodo *raiz,FILE *parchivo,int depth)
{
 int i;
 int j;

 if(raiz->arreglo_hijos == NULL)
 {
           for(j=0; j <depth; j++)
           fprintf(parchivo,"\t");
           fprintf(parchivo,"N %x T %d P %x ",raiz,raiz->tipo.raiz->padre);
           fprintf(parchivo,"te %d, At %d, Nh %d ",raiz->termino,raiz->atributo,raiz->num_hijos);
           fprintf(parchivo,"E %d Clase %d ".raiz->Ejemplos[0],raiz->clase);
           fprintf(parchivo,"\n");
 }
          else
          {
           for(j=0; j < depth; j++)
          fprintf(parchivo,"\t");
          fprintf(parchivo,"N %x T %d P %x ",raiz,raiz->tipo.raiz->padre);
          fprintf(parchivo,"te %d. At %d, Nh %d ",raiz->termino,raiz->atributo,raiz->num_hijos);
          fprintf(parchivo,"E %d Clase %d ".raiz->Ejemplos[0].raiz->clase);
           fprintf(parchivo,"\n");
 for(i = 0; i < raiz->num_hijos; i++)
          {
          imprime_arbol(raiz->arreglo_hijos[i],parchivo,depth+1);
          }
 }
  return 1;
}

void inicializa_raiz(struct Nodo *pn)
{
 int i;
```

```
 if(pn == NULL || pn->N_pertenencia == NULL)
          {
          printf("\n\tInicialize_raiz: Apuntador nulo. ");
           exit(1);
          }
          for(i = 0; i < Num_ejemplos; i++)
          pn->N_pertenencia[i] = 1.0;
}

 void crisp_crea_arreglo_terminos()
 {
 }

 int tipo_dato(int atributo)
 {
  return Arreglo_atributos[atributo - 1].tipo;
 }


int crisp_cuenta_ejemplos(int atributo,int termino,cval  val,int *ejemplos)
{
 int num_ejemplos = 0;
 int i;
 cval  lval;
 int tipo = 0;
 tipo = tipo_dato(atributo);
          for(i = 1; i <= ejemplos[0]; i++)
          {
          lval = dev_val_atributo(ejemplos[i],atributo);
          if(lval.cval == val.cval)
          num_ejemplos++;
          }
  return num_ejemplos;
}

cval  obten_termino(int atributo,int termino)
{
 return Terminos[atributo-1][termino-1].val;
}


int* crisp_obten_ejemplos(int *ejemplos,int atributo,int termino)
{
 int i = 0;
 int j = 0;
 int n = 0;
 int tipo = 0;
 cval  term,lval;
 int *subconjunto;

 term = obten_termino(atributo,termino);
 n = crisp_cuenta_ejemplos(atributo,termino,term,ejemplos);
 subconjunto = arreglo_int(n+1);
 subconjunto[0] = n; /* ver nota 5 del 30/07/98 */
          if(n == 0)
          return subconjunto;

          j++;
tipo = tipo_dato(atributo);
for(i = 1; i <= ejemplos[0]; i++)
          {
           lval = dev_val_atributo(ejemplos[i],atributo);
          if(lval.cval == term.cval)
```

```
                    {
                     subconjunto[j] = ejemplos[i];
                     j++;
                    }
            }
}
 return subconjunto;
}

 struct ejem_valor* crea_arreglo_ejem_valor(int tamaño)
 {
  struct ejem_valor *temp;
  temp = (struct ejem_valor*)malloc(tamaño * sizeof(struct ejem_valor));
  if(temp == NULL)
          {
           printf("\n crea_arreglo_ejem_valor: Memoria agotada.");
           exit(1);
          }
  return temp;
 }


 struct regla_ex* crea_arreglo_regla_ex(int num_reglas)
 {
  struct regla_ex* aux;
  aux = (struct regla_ex*)malloc(num_reglas * sizeof(struct regla_ex));
          if(aux == NULL)
           {
           printf("\n\tcrea_arreglo_regla_ex: Memoria insuficiente.");
           exit(1);
          }
  return aux;
 }
```

/* Archivo: treeope.c
Finalidad: Funciones para la definición de operaciones.*/

```
#include "ctree.h"
#include <stdlib.h>

 extern float log2(float);
 extern int atributo_probado(int,struct Nodo*);
 extern void fija_arreglo_hijos(struct Nodo*,int);
 extern struct Nodo* crea_nodo(void);
 extern int crisp_cuenta_ejemplos(int,int,union valor,int*);
 extern int* crisp_obten_ejemplos(int*,int,int);
 extern cval obten_termino(int,int);

/* Entropia.caso crisp*/

 float crisp_entropia(int *ejemplos)
 {
  int card_S = ejemplos[0];
  int i = 0;
  int en_clase_i = 0;
  float entropia = 0.0;
  float pi = 0.0;
  cval termino;

          if(card_S == 0)
          return 0;

          for(i = 1; i <= Num_terminos; i++)
```

```
        {
        termino = obten_termino(Num_atributos,i);
        en_clase_i = crisp_cuenta_ejemplos(Num_atributos,i,termino,ejemplos);
        pi = (float)en_clase_i / (float)card_S;
        entropia = entropia + ( -1.0) * (pi * log2(pi));
        }

 return entropia;
}

/* Ganancia para el caso crisp. */
float crisp_ganancia(int *ejemplos ,int atributo)
{
 float aganancia = 0.0;
 float entropiaS = 0.0;
 float entropiaSv = 0.0;
 float Sumatoria = 0.0;
 int *EjemplosSv;
 int Sv = 0;
 int S = ejemplos[0];
 int num_terminos = 0;
 int v = 0;
 cval termino;
 int i;
 num_terminos = terminos_atributo(atributo);
 entropiaS = crisp_entropia(ejemplos);
        for(v = 1; v <= num_terminos; v++)
        {
        EjemplosSv = crisp_obten_ejemplos(ejemplos,atributo,v);
        Sv = EjemplosSv[0];
        entropiaSv = crisp_entropia(EjemplosSv);
        Sumatoria = Sumatoria + ((float)Sv/(float)S) * entropiaSv;
        }
 aganancia = entropiaS - Sumatoria;
 return aganancia;
}

/* Elige el atributo con mayor ganacia caso crisp.*/
 int crisp_elige_atributo(struct Nodo *pn,struct C_atributo* atributos,int natributos)
 {
 float a_ganancia;
 float mayor_ganacia = -100.0;
 int mejor_atributo = -1;  /* atributo con mayor ganacia.        */
 int i;
 int ocupados = 0;
 int atributo_libre = 0;
 natributos = natributos - 1;

        for(i = 1; i <= natributos; i++)
        if(atributo_probado(i,pn) == 0)
         atributo_libre += 1;

         if(atributo_libre > 1)
        {
         for(i = 1; i <= natributos; i++)
         if(atributo_probado(i,pn) == 0) /* si no se ha utilizado. */
        {
         a_ganancia = crisp_ganancia(pn->Ejemplos,i);
         if(a_ganancia > mayor_ganacia)
         {
          mayor_ganacia = a_ganancia;
         mejor_atributo = i;
        }
         }
```

```
        }
 if(atributo_libre > 1)
         return mejor_atributo;
 if(atributo_libre == 1)
         {
         for(i = 1; i <= natributos; i++)
         if(atributo_probado(i,pn) == 0)
         atributo_libre = i;
         return atributo_libre; /* el unico atributo libre*/
         }
 if(atributo_libre == 0)
  return 0; /* retorna cero si ya todos los atributos estan usados. */
}
```

/* Archivo: ctree.h Contiene las definiciones de la implementaron de un árbol de decisión. */

```
#include <stdio.h>
#define MAX_LON_VAT 15 /* Longitud máxima del valor de un atributo */
#define MAX_LON_NAT 15 /* Longitud máxima del nombre de un atributo */
#define NOMINAL 0
#define PODADO 1
#define NOPODADO 0
typedef char* Atributo;
char   cval;
int Num_atributos; /* Numero de atributos. La clasificación se toma como atributo */
int Num_atributos_T; /* Numero de atributos entren. La clasificación se toma como atributo */
int Num_ejemplos; /* Numero de ejemplos de entrenamiento. */
int Num_ejemplos_T; /* Numero de ejemplos de prueba*/
int Num_reglas;   /* Numero de reglas.   */
int Num_hojas;    /* Numero de hojas del arbol.  */

struct C_atributo /* C_atributo guarda las caracteristicas de los atributos.    */
    {
     char nombre[MAX_LON_NAT];
     int utilizado; /* 0 = no utilizado, 1 = utilizado */
     int tipo;
    };

struct Nodo
    {
     int termino; /*  Indica el termino o rama de donde baja el nodo.
                      Este termino corresponde al atributo del nodo padre.*/
     int atributo; /* indica la localidad del atributo en el arreglo de atributos.
                      Este atributo es el que le corresponde a este nodo.*/
     int tipo;    /* 1 - interno u 0 - hoja */
     int num_hijos;
     int clase;   /* si es hoja asocialo a una clase  */

     struct Nodo *padre; /* Apuntador al nodo padre */
     struct Nodo **arreglo_hijos;   /* Apuntador al vector de apuntadores a nodos */
     float* N_pertenencia; /* Arreglo de pertenecias al nodo.  */
     int *Ejemplos; /* Guarda los indices de los ejemplos que corresponden al subconjunto de  entrenamiento
                       para el calculo de  la ganancia de información en este nodo. En la primera localidad se
                       coloca la cardinalidad de este conjunto. */
    }; /* Estructura de los nodos. */

struct condicion
        {
         int atributo;
        int operador; /* consecuente : 0 antecedente: 1 */
        int tipo;
        int podado; /* 0 indica que fue podado, 1 lo contrario. */
         struct condicion *siguiente;
```

```
        struct condicion *anterior;
       };

struct regla_ex   /* Guarda el numero de regla con su exactitud. */
       {
        int regla;
        float exactitud;
       };
struct condicion **Reglas;
struct condicion **Clasificadores;
struct Termino **Terminos; /* Apuntador a arreglo de terminos */
struct Nodo *P_raiz; /* Apuntador al nodo raiz */
cval *Arreglo_datos;  /* Arreglo de ejemplos de entrenamiento */
cval *Arreglo_datos_T; /* Arreglo de ejemplos de prueba */
char **CTerminos; /* Apuntador a arreglo de terminos caso crisp */
struct regla_ex *Exactitud_pre; /* Contiene la exactidut de las reglas antes del podado. */
struct regla_ex *Exactitud_pos; /* Contiene la exactidut de las reglas despues del podado. */
FILE* ARCHIVO;
```

/*Archivo : ftreecon.c
Contiene las definiciones de las funciones necesarias para la construcción del árbol */

```
#include "ctree.h"
#include <stdlib.h>
#include <stdio.h>

extern float informacion_e_IN(struct Nodo*);
extern void destruye_arbol(struct Nodo*);
extern void inicializa_raiz(struct Nodo*);
extern float* arreglo_float(int);
extern int* arreglo_int(int);
extern int determina_clase(int*);
extern int* crisp_obten_ejemplos(int*,int,int);
extern float crisp_entropia(int*);
extern void ejemplos_en_nodos(int*,int,int,FILE*);



void conecta_nodos(struct Nodo *padre, int localidad, struct Nodo *hijo)
{
       if(localidad >= padre->num_hijos)
        {
        printf("\n conecta_nodos: Numero de hijos menor a localidad\n");
       exit(0);
       }
        if(padre == NULL || hijo == NULL)
       {
        printf("\n conecta_nodos: Error apuntador Nulo\n");
        exit(0);
       }
 padre->arreglo_hijos[localidad]=hijo;
}

/* arreglo de apuntadores a Nodos. Los apuntadores apuntaran a los nodos hijo.*/

struct Nodo** crea_arreglo_h(int tamaño)
{
 struct Nodo **pnod;
 pnod = (struct Nodo**)malloc(tamaño * sizeof(struct Nodo*));
        if(pnod == NULL)
       {
        printf("\n\ncrea_arreglo: Espacio insuficiente para crear arreglo");
```

```
            exit(0);
           }
 return pnod;
}


void enlaza_arreglo_h(struct Nodo *pnod, struct Nodo** arreglo)
{
           if(pnod == NULL || arreglo == NULL)
           {
            printf("\nn enlaza_arreglo: Apuntador nulo");
           exit(0);
          }
 pnod->arreglo_hijos = arreglo;
}

/* Fija la clase que le corresponde a un nodo hoja.*/

 void crisp_fija_clase(struct Nodo *pn,int termino)
{
           if(pn == NULL)
           {
           printf("\n\t crisp_fija_clase: Apuntador NULO");
          exit(1);
          }
 pn->clase = termino;
}

/* Fija el valor del atributo tipo de un nodo.*/

 void fija_tipo(struct Nodo *pn,int valor)
{
          if(pn == NULL)
          {
           printf("\n\t fija_tipo: Apuntador NULO");
           exit(1);
          }
 pn->tipo = valor;
}

 void fija_padre(struct Nodo *padre,struct Nodo *hijo)
{
           if(padre == NULL || hijo == NULL)
           {
           printf("\n\t fija_padre: Apuntador NULO");
          exit(1);
          }
 hijo->padre = padre;
}

/*  Actualiza el atributo I de un nodo.*/

 void actualiza_al(struct Nodo* pn,float valor)
{
           if(pn == NULL)
           {
           printf("\n\t actualiza_al: Apuntador NULO");
           exit(1);
           }
 pn->IN = valor;
}

 void fija_arreglo_hijos(struct Nodo *pn,int nhijos)
{
```

```
        if(pn == NULL)
         {
        printf("\nn fija_arreglo_hijos: Apuntador nulo");
         exit(0);
        }
 enlaza_arreglo_h(pn,crea_arreglo_h(nhijos));
}

 struct Nodo* crea_nodo(void)
 {
  struct Nodo *pn;
  pn = (struct Nodo*)malloc(sizeof(struct Nodo));
          if(pn == NULL)
          {
          printf("\n\t crea_nodo: Memoria insuficiente para crear el arbol\n");
          destruye_arbol(P_raiz);
          exit(1);
         }
return pn;
 }

 /*   Crea el arreglo Clasificacion. */

  float** crea_clasificacion(int nejemplos,int nterminos)
  {
   float **pf;
   int i;
   pf = (float**)malloc(nejemplos * sizeof(float*));
          if(pf == NULL)
         {
          printf("\n crea_clasificacion: Memoria insuficiente.");
          exit(1);
          }

          for(i = 0; i < nejemplos; i++)
         pf[i] = (float*)malloc(nterminos * sizeof(float));
          if(pf[i] == NULL)
         {
          printf("\n crea_clasificacion: Memoria insuficiente.");
          exit(1);
          }
  return pf;
 }

/* Fija el atributo del nodo.*/

  void fija_atributo(struct Nodo* pn,int ag)
  {
   pn->atributo = ag;
  }

/*  Fija el termino que le corresponde a un nodo.*/

  void fija_termino(struct Nodo *pn,int termino)
  {
   pn->termino = termino;
  }

/* Fija el numero de hijos del nodo*/

  void fija_num_hijos(struct Nodo* pn,int nh)
  {
   pn->num_hijos = nh;
  }
```

```
/* Construcción del árbol caso crisp. */

int crisp_crea_arbol(struct Nodo **pn,struct Nodo *padre,int termino)
{
  int ag;       /* atributo con maxima ganancia. */
  int nh;       /* numero de hijos. */
  int i;
  int clase;
  float valor;
  int raiz = 0;
  float entropia = 0.0;
  struct Nodo *auxiliar;


  *pn = crea_nodo();
        if(*pn == P_raiz)
        padre = *pn;
          fija_padre(padre,*pn);
         fija_termino(*pn,termino);
         if(*pn == P_raiz)
         {
         (*pn)->Ejemplos = arreglo_int(Num_ejemplos + 1);
         (*pn)->Ejemplos[0] = Num_ejemplos;
         for(i = 1; i <= Num_ejemplos; i++)
         (*pn)->Ejemplos[i] = i;
         }
          else
          (*pn)->Ejemplos = crisp_obten_ejemplos((*pn)->padre->Ejemplos,(*padre).atributo,termino);

entropia = crisp_entropia((*pn)->Ejemplos);
        if(entropia == 0) /* si solo hay ejemplos de una clase */
        {
         clase = determina_clase((*pn)->Ejemplos);
         crisp_fija_clase(*pn,clase);
         fija_tipo(*pn,0);
          Num_hojas++;
           fija_num_hijos(*pn,0);
           return 0;
         }
   ag = crisp_elige_atributo(*pn,Arreglo_atributos,Num_atributos);
        if(ag == 0) /* atributos agotados */
        {
        clase = determina_clase((*pn)->Ejemplos);
        crisp_fija_clase(*pn,clase);
         fija_tipo(*pn,0);
        Num_hojas++;
        fija_num_hijos(*pn,0);
        return 0;
         }

  fija_tipo(*pn,1);
  fija_atributo(*pn,ag);
  nh = num_hijos(ag,Arreglo_atributos);
  fija_num_hijos(*pn,nh);
  fija_arreglo_hijos(*pn,nh);
  marca_atributo(Arreglo_atributos,ag);
  padre = *pn;
  for(i = 1; i <= nh; i++)
   crisp_crea_arbol(&(*pn)->arreglo_hijos[i - 1],padre,i);
 }
```

/* Archivo : ctreepru.c Finalidad: Tiene funciones para el podado del árbol de decisión. */

```
#include "ctree.h"
#include <stdio.h>
#include <stdlib.h>
extern cval obten_termino(int,int);
extern char* dev_nombre_atrib(int);
extern int tipo_dato(int);
extern cval obten_termino(int,int);
extern cval dev_val_atributo_T(int,int);
extern struct regla_ex* crea_arreglo_regla_ex(int);

struct condicion** crea_arreglo_reglas(int num_reglas)
{
 struct condicion **a_reglas;
 a_reglas =(struct condicion**)malloc(num_reglas * sizeof(struct condicion*));
 return a_reglas;
} /* crea_arreglo_reglas */


struct condicion* crea_condicion(void)
{
 struct condicion *cond;

 cond = (struct condicion *)malloc(sizeof(struct condicion));
        if(cond == NULL)
        {
        printf("\n crea_condicion: Espacio insuficiente.");
        exit(1);
        }
 return cond;
}

int imprime_reglas(struct Nodo *raiz,FILE *parchivo)
{
 struct Nodo *nodo;
 int i = 0;
 int tipo;
 char nombre[MAX_LON_NAT];
 cval  termino;

        if(raiz->tipo == 0)
         {
         if(raiz == P_raiz)
        {
         printf("\n imprime_reglas: Raiz Unico nodo.");
        return 1;
         }

 fprintf(parchivo,"\n");
 nodo = raiz;
 while(1)
 {
  /* viaje de la hoja a la raiz para obtener la regla. */
         if(nodo->tipo == 0)
        {
        fprintf(parchivo,"Clase = %d <=== ",nodo->clase);
         tipo = tipo_dato(nodo->padre->atributo);
                 termino = obten_termino(nodo->padre->atributo,nodo->termino);
        fprintf(parchivo,"%s = %c && ",dev_nombre_atrib(nodo->padre->atributo),termino.cval);
         tipo = tipo_dato(nodo->padre->atributo);
         termino = obten_termino(nodo->padre->atributo,nodo->termino);
        fprintf(parchivo,"%s = %c && ",dev_nombre_atrib(nodo->padre->atributo),termino.cval);
```

```
        }
  nodo = nodo->padre;
  if(nodo == P_raiz)
   break;
 } /* while */


 nodo = NULL;
 return 1;
 }
        else
        for(i = 0; i < raiz->num_hijos; i++)
        imprime_reglas(raiz->arreglo_hijos[i],parchivo);
         return 1;
} /*  genera_reglas(struct Nodo *) */

int cuenta_hojas(struct Nodo *raiz)
{
 int i;
        if(raiz->tipo == 0)
        Num_hojas++;
        else
        for(i = 0; i < raiz->num_hijos; i++)
        cuenta_hojas(raiz->arreglo_hijos[i]);
        return 1;
}

int genera_lista_reglas(struct Nodo *raiz)
{
 struct Nodo *nodo;
 struct condicion *cond;
 struct condicion *auxiliar;
 cval !val;
 int i = 0;
          if(Reglas == NULL)
        {
        printf("\n genera_lista_reglas: Arreglo de reglas NULO.");
         exit(1);
        }
        if(P_raiz->tipo == 0)
        {
        printf("\n genera_lista_reglas: !Raiz: Unico nodo.!");
        return 1;
        }
        if(raiz->tipo == 0)
         {
        nodo = raiz;
        Num_reglas++;
        cond = crea_condicion();
          auxiliar = cond;
        cond->atributo = nodo->atributo;
        cond->termino = obten_termino(Num_atributos,nodo->clase);
        cond->operador = 0; /* es el consecuente */
        cond->tipo = 0; /* tipo_dato(nodo->atributo); */
        cond->podado = 1;
        cond->siguiente = NULL;
        cond->anterior  = NULL;
         Reglas[Num_reglas - 1] = cond;
        /* antecedentes */
        cond = crea_condicion();
        cond->anterior = auxiliar;
        auxiliar->siguiente = cond;
        auxiliar = cond;
         cond->atributo = nodo->padre->atributo;
```

```
          cond->tipo = tipo_dato(cond->atributo);
          cond->termino = obten_termino(nodo->padre->atributo,nodo->termino);
          cond->operador = nodo->termino;
          cond->podado = 1;
          cond->siguiente = NULL;
          while(1)
          {    /* viaje de la hoja a la raiz para obtener la regla. */
          nodo = nodo->padre;
          if(nodo == P_raiz)
           {
           break;
  }

  cond = crea_condicion();
  auxiliar->siguiente = cond;
  cond->anterior = auxiliar;
  auxiliar = cond;
  cond->atributo = nodo->padre->atributo;
  cond->tipo = tipo_dato(cond->atributo);
  cond->termino = obten_termino(nodo->padre->atributo,nodo->termino);
  cond->operador = nodo->termino;
  cond->podado = 1;
  cond->siguiente = NULL;

 }/* while */
 auxiliar = NULL;
 cond = NULL;
}
 else
          for(i = 0; i < raiz->num_hijos; i++)
          genera_lista_reglas(raiz->arreglo_hijos[i]);
           return 1;
} /* genera_reglas */


int imprime_lista_reglas(char *archivo,struct condicion** reglas)
{
 int i;
 int tipo;
 FILE *pa;
 struct condicion *auxiliar;

 pa = fopen(archivo,"w");
 if(pa == NULL)
          {
          printf("Imposible abrir archivo %s ",archivo);
          exit(1);
          }
           if(reglas == NULL)
          fprintf(pa,"imprime_lista_reglas: Arreglo de reglas NULO\n");
          for(i = 0; i < Num_hojas; i++)
          {
          auxiliar = reglas[i];
          fprintf(pa,"\n");
           fprintf(pa,"Clase = %c) <=== ",auxiliar->termino.cval); /* este nodo corresponde a una hoja */
           if(auxiliar->siguiente == NULL)
          {
          printf("Solo el nodo hoja. Revisar.");
          return 1;
          }
  auxiliar = auxiliar->siguiente;
          do{
           tipo = tipo_dato(auxiliar->atributo);
          fprintf(pa,"(%s = %c) && ",dev_nombre_atrib(auxiliar->atributo),auxiliar->termino.cval);
```

```
        auxiliar = auxiliar->siguiente;
        } while(auxiliar != NULL);
 }
}

int genera_clasificador(void)
{
 int i;
 struct condicion* auxiliar;
 struct condicion* nodo;
 Clasificadores = crea_arreglo_reglas(Num_hojas);
        for(i = 0; i < Num_hojas; i++)
         {
         auxiliar = Reglas[i];
        do
        {
         auxiliar = auxiliar->siguiente;
        }while(auxiliar->siguiente != NULL);

 Clasificadores[i] = auxiliar;
    }
  } /* genera_clasificador(void) */

void genera_reglas(void)
{
 Reglas = crea_arreglo_reglas(Num_hojas);
 genera_lista_reglas(P_raiz);
 genera_clasificador();
}

int imprime_clasificador(char *archivo,struct condicion** reglas)
{
 int i;
 int tipo;
 FILE *pa;
 struct condicion *auxiliar;

 pa = fopen(archivo,"w");
         if(pa == NULL)
        {
        printf("Imposible abrir archivo %s ",archivo);
        exit(1);
        }
        if(reglas == NULL)
         fprintf(pa,"imprime_lista_reglas: Arreglo de reglas NULO\n");

        for(i = 0; i < Num_hojas; i++)
        {
        auxiliar = reglas[i];
         fprintf(pa,"\n");

        do{
         if(auxiliar->operador == 0)
        {
        fprintf(pa," ===> Clase = %c)",auxiliar->termino.cval); /* este nodo corresponde a una hoja */
        break;
        }
         if(auxiliar->podado == 1)
        {
         tipo = tipo_dato(auxiliar->atributo);
        fprintf(pa,"(%s = %c) && ",dev_nombre_atrib(auxiliar->atributo),auxiliar->termino.cval);
        }
  auxiliar = auxiliar->anterior;
  }while(1);
```

```
  }
 }

 int cumple_condicion(struct condicion *cond, cval  val_ejemplo)
 {
  /* Retorna 0 si no cumple, 1 si cumple. */
  if(!cond->tipo)
  {
   if(cond->termino.cval == val_ejemplo.cval)
    return 1;
   else
    return 0;
  }
 } /* cumple_condicion  */

 char clasificador(int ejemplo,int regla,int podado)
 {
  /* devuelve la clasificacion del ejemplo según la regla.  */
  cval val_ejemplo;
  struct condicion *auxiliar;
  int valor_verdad = -1;
  char clasificacion = '&';
  auxiliar = Clasificadores[regla - 1];

 if(podado == PODADO)
          do
          {
          {
           if(auxiliar->podado == PODADO)
          {
          val_ejemplo = dev_val_atributo_T(ejemplo,auxiliar->atributo);
           valor_verdad = cumple_condicion(auxiliar,val_ejemplo);
           if(valor_verdad == 0)
          {
           clasificacion = '@'; /* esto es devuelto si no cumple alguna precondicion */
          break;
           }
           else
    clasificacion = '&';
   }
  }/* if */
  auxiliar = auxiliar->anterior;
 }while(auxiliar->operador != 0);
 else
  do
  {
          {
    val_ejemplo = dev_val_atributo_T(ejemplo,auxiliar->atributo);
    valor_verdad = cumple_condicion(auxiliar,val_ejemplo);
    if(valor_verdad == 0)
          {
          clasificacion = '@';
           break;
           }
           else
           clasificacion = '&';
          }/* if */
  auxiliar = auxiliar->anterior;
 }while(auxiliar->operador != 0);
           if(clasificacion == '&')
           return auxiliar->termino.cval;
          else
          return clasificacion;
          }/* clasificador */
```

```
/*  Prueba si la regla clasifica bien al ejemplo. */
int clasifica_bien(int ejemplo,int regla,int podado)
{
 char clas_regla;
 char clas_ejem;
 cval  val_ejemplo;
 int tipo;
 clas_regla = clasificador(ejemplo,regla,podado);
 val_ejemplo = dev_val_atributo_T(ejemplo,Num_atributos);
         if(clas_regla == '@')
          return -1;
          if(clas_regla == val_ejemplo.cval)
         return 1;
         else
         return 0;
}

int post_pruning_regla(int regla)
{
 /*  El podado sera regla por regla  */
 struct condicion *auxiliar;
 int ejemplo;
 int i;
 int bien_clasificados = 0;
 int mal_clasificados = 0;
 int clasificacion = 0;
 float exactitud = 0.0;
 if(Reglas == NULL)
         {
         printf("\n\tpost_pruning_regla: Reglas es NULO.\n");
         exit(1);
         }
         for(i = 0; i < Num_reglas; i++)
         Exactitud_pos[i].exactitud = Exactitud_pre[i].exactitud;
         auxiliar = Reglas[regla - 1];

 do
 {
          if(auxiliar->tipo)
          {
         auxiliar->podado = 0;
         for(ejemplo = 1; ejemplo <= Num_ejemplos_T; ejemplo++)
         {
         clasificacion = clasifica_bien(ejemplo,regla,PODADO);
         if(clasificacion == 1)
          bien_clasificados++;
          if(clasificacion == 0)
         mal_clasificados++;
  }
 exactitud = (float)bien_clasificados /(mal_clasificados + Num_ejemplos_T);

  /* Si mejoro la clasificacion dejarlo como podado. en otro caso dejarlo no podado */
          if(Exactitud_pre[regla - 1].exactitud <= exactitud)
         Exactitud_pos[regla - 1].exactitud = exactitud;
         else
         auxiliar->podado = 1;
 }
 auxiliar = auxiliar->siguiente;
}while(auxiliar != NULL);

}/* post_pruning_regla */

void exactitud_reglas(int podado)
```

```
{
 struct condicion *auxiliar;
 int ejemplo;
 int clasificacion = 0;
 int bien_clasificados = 0;
 int mal_clasificados = 0;
 float exactitud = 0.0;
 int regla;

        if(Reglas == NULL)
        {
         printf("\n\texactitud_reglas: Reglas es NULO.\n");
         exit(1);
        }
        for(regla = 1; regla <= Num_reglas; regla++)
        Exactitud_pre[regla -1].exactitud = 0.0;
        for(regla = 1; regla <= Num_reglas; regla++)
        {
        bien_clasificados = 0;
        mal_clasificados = 0;
        auxiliar = Reglas[regla - 1];
        do
        {
         if(auxiliar->tipo)
         {
        for(ejemplo = 1; ejemplo <= Num_ejemplos_T; ejemplo++)
        {
        clasificacion = clasifica_bien(ejemplo,regla,podado);
                if(clasificacion == 1)
                bien_clasificados++;
                if(clasificacion == 0)
                mal_clasificados++;
        }
  }
  auxiliar = auxiliar->siguiente;
 }while(auxiliar != NULL);
exactitud = (float)bien_clasificados /(mal_clasificados + bien_clasificados +1);
Exactitud_pre[regla - 1].exactitud = exactitud;
 }
}/* exactitud_reglas */

void post_pruning(void)
{
 int i;
 Exactitud_pre = crea_arreglo_regla_ex(Num_reglas);
 Exactitud_pos = crea_arreglo_regla_ex(Num_reglas);
 exactitud_reglas(NOPODADO);
 imprime_exactitud(Exactitud_pre);
 for(i = 1; i <= Num_reglas; i++)
 post_pruning_regla(i);
}

float desempeño_reglas(int podado)
{
 struct condicion *auxiliar;
 int ejemplo;
 int clasificacion = 0;
 int bien_clasificados = 0;
 int mal_clasificados = 0;
 float exactitud = 0.0;
 int regla;
        if(Reglas == NULL)
        {
         printf("\n\tdesepeño_reglas: Reglas es NULO.\n");
```

```
            exit(1);
           }

for(regla = 1; regla <= Num_reglas; regla++)
           {
           auxiliar = Reglas[regla - 1];
           do
           {
            if(auxiliar->tipo)
           for(ejemplo = 1; ejemplo <= Num_ejemplos_T; ejemplo++)
           {
           clasificacion = clasifica_bien(ejemplo,regla,podado);
                    if(clasificacion == 1)
                    bien_clasificados++;
                     if(clasificacion == 0)
                    mal_clasificados++;
           }
  auxiliar = auxiliar->siguiente;
 }while(auxiliar != NULL);
}
exactitud = (float)bien_clasificados / (mal_clasificados + Num_ejemplos_T);
 return exactitud;
}/* desempeño_reglas */

int cuenta_bien_clasificados(int podado)
{
 int regla;
 int ejemplo;
 int aciertos = 0;
 int clasificacion = 0;

 for(ejemplo = 1; ejemplo <= Num_ejemplos_T; ejemplo++)
           for(regla = 1; regla <= Num_reglas; regla++)
           {
           clasificacion = clasifica_bien(ejemplo,regla,podado);
           if(clasificacion == -1)
           continue;
            if(clasificacion == 1)
           {
   aciertos++;
   fprintf(ARCHIVO,"ejemplo :%d, regla:%d\n",ejemplo,regla);
   break;
  }
 }
 return aciertos;
} /* cuenta_bien_clasificados() */
libera_memoria_reglas(void)
{
}
```