FACULTAD DE INFORMÁTICA – UNIVERSIDAD NACIONAL DE LA PLATA

EXTRACCIÓN DE CONOCIMIENTO EN GRANDES BASES DE DATOS UTILIZANDO ESTRATEGIAS ADAPTATIVAS

Tesis presentada para obtener el grado de
Doctor en Ciencias Informáticas

Lic. Waldo Hasperué

Directores

Ing. Armando De Giusti
Lic. Laura Lanzarini

Febrero de 2012

Agradecimientos

A los dos amores de mi vida, Virginia y Sofía, quienes me brindaron su apoyo en todo momento, me dieron ánimo para terminar este trabajo y supieron, en este último tiempo, quedar en un segundo plano para que pudiera terminar de escribir esta tesis.

A mis directores, Tito y Laura, quienes con sus comentarios, sugerencias y valiosos consejos lograron que esta empresa llegara a su fin.

Tabla de contenido

Resumen

El objetivo general de esta tesis es el desarrollo de una técnica adaptativa para la extracción de conocimiento en grandes bases de datos.

Hoy en día, la tecnología posibilita el almacenamiento de enormes volúmenes de información. Por tal motivo, resulta de interés contar con técnicas que permitan, en una primera etapa, analizar tal información y obtener conocimiento que pueda ser expresado como reglas de clasificación. Sin embargo, es de esperar que la información disponible se modifique o incremente a lo largo del tiempo y por lo tanto, en una segunda etapa, sería relevante poder adaptar el conocimiento adquirido a los cambios o variaciones que ocurran en el conjunto de datos original.

El aporte de la tesis está centrado en la definición de una técnica adaptativa que permite extraer conocimiento de grandes bases de datos a partir de un modelo dinámico capaz de adaptarse a los cambios de la información, obteniendo así una técnica de minería de datos que sea capaz de generar conocimiento útil, produciendo resultados que sean de provecho al usuario final.

Los resultados de esta investigación pueden aplicarse en áreas tales como análisis de suelos, análisis genético, biología, robótica, economía, medicina, detección de fallas en plantas y comunicación de sistemas móviles. En estos casos es importante la obtención de un resultado óptimo, de modo de mejorar la calidad de las decisiones que se toman a partir del procesamiento. Desde el punto de vista informático estos problemas son un desafío interesante debido al volumen y distribución de los datos a analizar (incluso su complejidad) para obtener el conocimiento buscado.

1. Motivación

Diversas áreas de trabajo cuentan con grandes bases de datos, de las cuales resulta útil extraer conocimiento que sirva de ayuda al usuario final. En esta dirección, las técnicas adaptativas son muy importantes en estos sistemas ya que permiten crear un modelo de la información contenida en la base de datos, que represente el conocimiento actual, con capacidad suficiente para adaptarse automáticamente a los cambios de la información disponible, sin necesidad de realizar toda la tarea de extracción de conocimiento nuevamente.

En los últimos años la toma de decisiones ha ganado popularidad en distintos dominios, apareciendo distintas técnicas que ayudan a las personas encargadas de la toma de decisiones. Estas técnicas son llamadas sistemas de soporte de decisión. Una solución aceptable para estos sistemas debería poder ofrecer al usuario final distintas alternativas para realizar la toma de decisiones y de ser posible de manera online. Luego, a partir de la información obtenida debería actuar de acuerdo a experiencias pasadas, recordando cuales fueron respuestas efectivas y filtrando las indeseadas.

2. Desarrollos y aportes

Mi inserción al paradigma de la minería de datos ocurre con las redes neuronales artificiales y en particular con los mapas auto-organizativos dinámicos. La elección de este tipo de redes se debe a su capacidad para definir la arquitectura durante el proceso adaptativo. Además estas arquitecturas resultan robustas para lograr buenos modelos de los datos. Producto de esa investigación, en mi tesis de licenciatura, desarrollé una nueva estrategia que mejora la preservación de la topología de los datos de entrada respecto de las soluciones existentes (Hasperué & Lanzarini, 2005).

Debido al concepto de "caja negra" que tienen las redes neuronales artificiales y la gran dificultad que poseen para extraer información útil a partir de lo aprendido, comencé a estudiar las diferentes opciones para extraer

conocimiento a partir de un mapa auto-organizativo. Producto de esa investigación logré una técnica que es capaz de extraer un conjunto de reglas a partir de un mapa auto-organizativo (Hasperué & Lanzarini, 2006).

Al trabajar y realizar distintos ensayos con diferentes modelos de redes neuronales surgió la necesidad de combinar los resultados obtenidos por los diferentes modelos logrando, mediante acumulación de evidencia, definir una caracterización más adecuada de los datos (Hasperué & Lanzarini, 2007a).

Las estrategias desarrolladas facilitaron la construcción de modelos a partir de la información disponible principalmente en forma de reglas de clasificación. Sin embargo, distintas situaciones en las cuales el modelo por si solo era incapaz de transmitir al usuario el conocimiento adquirido así como su importancia y relación con los datos subyacentes me llevaron a desarrollar estrategias para la toma de decisiones.

El gran obstáculo que presentaba la transferencia tecnológica en temas referidos a estrategias de minería de datos, era el desconocimiento por parte del usuario acerca de la forma de adaptar el modelo a sus necesidades. Por ese motivo comencé a estudiar estrategias que permitieran obtener a partir de la información del beneficio esperado, las acciones necesarias para modificar los datos y por lo tanto, lograr cambiar el modelo adecuadamente.

Como solución a estos problemas desarrollé una técnica que define, a partir del modelo basado en reglas de clasificación, las acciones a seguir para lograr el beneficio esperado. Esta técnica construye las reglas utilizando una matriz de co-asociación cuyos valores surgen de la combinación de distintos métodos de clustering aplicados a los datos de entrada. De esta manera, aunque la decisión final continua estableciéndola el usuario, puede disponerse de una línea de acción (Hasperué & Lanzarini, 2007b).

Otro aspecto importante que surgió durante mis investigaciones en minería de datos, es el dinamismo de la información. Como la tecnología posibilita el almacenamiento de enormes volúmenes de datos gran parte de las estrategias inteligentes existentes creaban modelos para un instante de tiempo dado. Sin embargo, comienzan a aparecer una amplia gama de problemas que requieren disponer de mecanismos capaces de adaptar el modelo existente a los cambios del entorno. En (Hasperué, y otros, 2008) generé una técnica de obtención de reglas difusas las cuales facilitan su adaptación ante nueva información.

Continuando con la aplicación de las redes neuronales en el área de clustering surgió la necesidad de construir modelos de datos utilizando hiper-rectángulos como mecanismo de representación de los datos. Con esta idea desarrollé una estrategia que utiliza hiper-rectángulos creados a partir de los vectores de pesos de las neuronas de un SOM entrenado para realizar tareas de clustering. Esta estrategia crea hiper-rectángulos agrupando los vectores de pesos de neuronas vecinas y luego mediante la expansión, contracción, unión y separación de hiper-rectángulos determina la cantidad de clusters del conjunto de datos inicial y como están conformados cada uno de ellos. La gran ventaja de utilizar hiper-rectángulos en problemas de clustering es que estos de manera sencilla dan como resultado las reglas que los describen con respecto al resto de hiper-rectángulos y de esa manera ofrecen al usuario final reglas para la comprensión de los datos y su uso ante nuevos datos obtenidos (Hasperué & Lanzarini, 2010).

Dejando de lado las redes neuronales y con la decisión de resolver problemas de clasificación, al mismo tiempo que se extrae conocimiento de un modelo de datos adaptativo, surge la técnica presentada en esta tesis doctoral.

Cuando el dominio de los datos a tratar es continuo una forma de representar a los datos que resulta práctica y que ha sido utilizada en varios trabajos de minería de datos es el hiper-rectángulo. Los hiper-rectángulos son una poderosa forma de representación de datos ya que es capaz de describir de manera casi natural el subconjunto de datos al cual representa, ya que los límites de cada hiper-rectángulo formado en el modelo de datos pueden ser utilizados como cláusulas en las reglas del tipo IF-THEN que resulten del proceso de extracción de conocimiento. Al mismo tiempo, y por sus características, pueden ser manejados fácilmente en el dominio de los datos de entrada permitiendo la contracción, unión y división con operaciones simples.

Durante el desarrollo de la tesis doctoral se logró la definición e implementación de una técnica que extrae conocimiento de grandes volúmenes de información en forma de reglas de clasificación, las cuales le ayudan al usuario final a comprender los datos con los cuales trabaja y al mismo tiempo le sirve para la toma de decisiones. Estas reglas, formadas a partir de los hiper-rectángulos que resultan de un modelo de datos creado son de carácter estricto representando la frontera entre dos hiper-rectángulos de manera totalmente rígida. Es decir, que un mismo dato sólo puede pertenecer a un único hiper-rectángulo.

El sistema genera hiper-rectángulos a partir de los datos iniciales y en base a un proceso iterativo de eliminación de superposiciones va tomando decisiones de cuál superposición debe ser eliminada basada en los cálculos de un conjunto de índices de superposición propuestos y desarrollados en esta tesis. Las decisiones provocan que los hiper-rectángulos modifiquen su tamaño o se dividan. Este proceso de optimización continúa hasta minimizar (o anular) el volumen de intersección entre los hiper-rectángulos de distintas clases. Finalmente se generan las reglas que resultan de los distintos hiper-rectángulos conseguidos.

En cuanto al comportamiento adaptativo, los nuevos datos que se ingresen al modelo de datos, causan la modificación de los hiper-rectángulos y eventualmente nuevas divisiones o uniones. Esto implica que el modelo no debe rehacerse nuevamente utilizando el conjunto de datos completo sino que solo se adapta modificando su estructura interna ante la llegada de nuevos datos. Una vez actualizada la estructura interna del modelo se actualiza el conjunto de reglas existentes.

La estrategia adaptativa propuesta presenta la particularidad que el usuario puede actuar supervisando y participando en la decisión del algoritmo al formar o ajustar los hiper-rectángulos. De esta manera un experto en el dominio del problema puede aportar valiosísima información durante la creación del modelo de datos.

Al mismo tiempo, el propio modelo, además de ofrecer las distintas alternativas y posibles soluciones con distintos grados de validez puede realizar sugerencias al usuario sobre la toma de acciones específicas basadas en el conocimiento adquirido y la tendencia del dinamismo de los datos, si es que se está llevando a cabo alguna actualización de los mismos.

Así, se ha establecido una estrategia adaptativa capaz de formar un primer modelo a partir de una base de datos y luego ir adaptándose gradualmente a medida que se adquiere nueva información. Esta estrategia si bien puede funcionar de manera completamente automática ofrece la posibilidad de la participación de un experto el cual decide en qué grado involucrase en el armado del modelo de datos.

Como trabajo futuro se puede adaptar esta estrategia a la lógica difusa lo cual le brinda al usuario final un abanico aún mayor de posibles soluciones como así también la posibilidad de agregar "memoria" al modelo para que sea capaz de recordar la toma de decisiones llevadas a cabo por el experto, de esa manera ante una próxima adaptación del modelo, este mismo puede sugerir como llevar a cabo dicha modificación en base a las mismas acciones tomadas con anterioridad.

3. Publicaciones derivadas de esta tesis doctoral

- Hasperué, W.; Lanzarini, L. 2010. "A new clustering strategy for continuous datasets using hypercubes". 36th Conferencia Latinoamericana de Informática (CLEI 2010). Asunción, Paraguay. Octubre 2010.

- Hasperué, W.; Osella Massa, G.; Lanzarini, L. 2008. "Obtaining a Fuzzy Classification Rule System from a non-supervised Clustering". 30th International Conference of Information Technology Interfaces (ITI). Cavtat, Croacia. June 2008.

- Hasperué, W.; Lanzarini, L. 2007b. "Extracting Actions from Classification Rules". Workshop de Inteligencia Artificial. Jornadas Chilenas de Computación 2007. Iquique, Chile. Noviembre de 2007

- Hasperué, W.; Lanzarini, L. 2007a. "Classification Rules Obtained from Evidence Accumulation". 29th International Conference on Information Technology Interfaces. ITI 2007. Dubrovnik, Croacia. 25 a 28 de junio de 2007. ISBN: 978-953-7138-09-7 / ISSN: 1330-1012 (CD Rom) – Publicado por IEEE Computer Society Press – Pág. 167-172.

- Hasperué, W.; Lanzarini, L. 2006. "Classification Rules obtained from Dynamic Self-organizing Maps". VII Workshop de Agentes y Sistemas Inteligentes, XII Congreso Argentino de Ciencias de la Computación. CACIC 2006. San Luis. Argentina. Octubre 2006. Pág. 1220-1230.

- Hasperué, W.; Lanzarini, L. 2005. "Dynamic Self-Organizing Maps. A new strategy to enhance topology preservation". XXXI Conferencia Latinoamericana de Informática (CLEI 2005). Cali. Colombia. Octubre de 2005. pp 1081-1087.

Capítulo 1

Introducción a la minería de datos

Lo que sabemos es una gota de agua; lo que ignoramos es el océano.
(Isaac Newton)

La minería de datos nace por la aparición de las nuevas necesidades de utilizar la gran cantidad de datos almacenados por los sistemas de información de instituciones, empresas, gobiernos y particulares durante muchos años. Los datos pasan a ser la materia prima que hay que explotar para obtener un nuevo producto, el conocimiento. Este conocimiento se convierte en un elemento muy valioso para la ayuda en la toma de decisiones sobre el ámbito en el que se han recopilado o extraído los datos.

Si bien la estadística es la primera ciencia que consideró a los datos como su materia prima, ante las nuevas necesidades y las nuevas características de los datos (gran volumen y tipología), aparecen un importante número de disciplinas que comienzan a integrar lo que se conoce como minería de datos.

Dentro de la minería de datos existe una rama que se dedica a los procesos de extracción de conocimiento cuyo objetivo es el descubrimiento de conocimiento en bases de datos. Estos procesos constan de una secuencia iterativa de etapas o fases. Además presentan toda una tipología de tareas y técnicas para resolverlo y cuentan con medidas de evaluación mediante los conceptos de conjunto de entrenamiento y de prueba.

1. Minería de datos

El aumento del volumen y variedad de información que se encuentra almacenada en grandes bases de datos digitales y otras fuentes ha crecido enormemente en las últimas décadas. Gran parte de esta información es histórica, es decir, representa transacciones o situaciones que se han producido. Aparte de su función de "memoria de la organización", la información histórica es útil para explicar el pasado, entender el presente y predecir la información futura. La mayoría de las decisiones de empresas, organizaciones e instituciones se basan en gran medida, en información sobre experiencias pasadas extraídas de fuentes muy diversas, resultando necesario analizar los datos para la obtención de información útil.

En muchas situaciones, el método tradicional de convertir los datos en conocimiento consiste en un análisis e interpretación realizada de forma manual. El especialista en la materia analiza los datos y elabora un informe o hipótesis que refleja las tendencias o pautas de los mismos. Por ejemplo, un grupo de médicos puede analizar la evolución de enfermedades infecto-contagiosas entre la población para determinar el rango de edad más frecuentes de las personas afectadas. Este conocimiento puede ser usado por la autoridad sanitaria competente para establecer políticas de vacunación.

Esta forma de actuar es lenta, cara y altamente subjetiva. De hecho, el análisis manual es impracticable en dominios donde el volumen de los datos crece exponencialmente. Consecuentemente, muchas decisiones importantes se realizan, no sobre la base de los datos disponibles sino siguiendo la propia intuición del usuario al no disponer de las herramientas necesarias.

Existen herramientas analíticas que han sido empleadas para analizar los datos y tienen su origen en la estadística. Si bien estas herramientas son capaces de inferir patrones a partir de los datos, el problema es que resultan algo crípticos para los no estadísticos y generalmente no funcionan bien para la gran cantidad de volumen de información existente hoy en día, además, no se integran bien con los sistemas de información. En muchos contextos, como los negocios, medicina o la ciencia lo interesante es el conocimiento que puede inferirse a partir de los datos y, más aún, la capacidad de poder usar este conocimiento. Por ejemplo, es posible saber estadísticamente que el 10% de los ancianos padecen Alzheimer. Esto puede ser útil, pero seguramente es mucho más útil tener un conjunto de reglas que a partir de los antecedentes, los hábitos y otras características del individuo nos digan si un paciente tendrá o no Alzheimer.

Los problemas y limitaciones de la falta de herramientas han hecho surgir la necesidad de una nueva generación de técnicas para soportar la extracción de conocimiento útil desde la información disponible, y que se engloban bajo la denominación de minería de datos. La minería de datos se distingue de otras técnicas porque no obtiene información extensional (datos) sino intencional (conocimiento) y además el conocimiento no es una parametrización de ningún modelo pre-establecido o intuido por el usuario, sino que es un modelo novedoso y original extraído completamente por la herramienta.

El resultado de la minería de datos son conjuntos de reglas, ecuaciones, árboles de decisión, redes neuronales, grafos probabilísticos, entre otros, los cuales pueden usarse para responder a diferentes cuestionamientos. Por ejemplo, ¿existe un grupo de clientes que se comporta de manera diferenciada?, ¿qué secuenciación de tratamientos puede ser más efectiva para este nuevo síndrome?, ¿existen asociaciones entre los factores de riesgo para realizar un seguro de automóvil?, ¿cómo califico automáticamente los mensajes de correo entre más o menos susceptibles de ser spam?

La tarea fundamental de la minería de datos es encontrar modelos inteligibles a partir de los datos. Para que este proceso sea efectivo debería ser automático o semi-automático (asistido) y el uso de los patrones descubiertos debería ayudar a tomar decisiones más seguras que reporten, por tanto, algún beneficio a la organización.

Por lo tanto, dos son los retos de la minería de datos. Por un lado, trabajar con grandes volúmenes de datos, procedentes mayoritariamente de sistemas de información, con los problemas que ello conlleva (ruido, datos ausentes, intratabilidad, volatilidad de los datos, confidencialidad, etc.) y por el otro, usar técnicas adecuadas para analizar los mismos y extraer conocimiento novedoso y útil. No hay que olvidar que el usuario final no tiene por qué ser un experto en las técnicas de minería de datos, ni tampoco puede perder mucho tiempo interpretando los resultados. Por esto, la información descubierta debe ser presentada de manera comprensible, por ejemplo usando gráficos, reglas en lenguaje natural o técnicas de visualización de datos) (Hernández Orallo, y otros, 2004).

Las diferentes técnicas de minería de datos, que serán vistas en las siguientes secciones, han sido utilizadas para resolver una amplia gama de problemas de distintas naturalezas. Entre los muchos trabajos existentes al respecto es posible citar algunos, como por ejemplo, al presentado en (Morik, y otros, 2012) donde los autores hacen una revisión de los aportes realizados en minería de datos en diferentes ciencias como climatología, recursos naturales, predicción de desastres naturales, biodiversidad, sustentabilidad, etc. En el trabajo presentado en (Tsai, y otros, 2011) los autores resuelven una tarea muy particular de minería de datos, ya que el objetivo es realizar clustering de trayectorias similares en objetos en movimiento. En (Gupta, y otros, 2010) presentan un método para realizar clustering de datos espaciales en ambientes de robótica. En (Bae, y otros, 2009) presentan un framework para diferentes tareas de minería de datos de GIS (Geographical Information Sistem). En (Lin, y otros, 2009) utilizan una novedosa métrica para medir similitudes entre los datos y así realizar minería de datos en bases de datos con información espacio-temporal. En (Khoshgoftaar, y otros, 2003) utilizan la programación genética para la construcción de un árbol de decisión en la aplicación de calidad de

software. En (Wang, y otros, 2008) utilizan una técnica híbrida entre árboles de decisión y SVM (Support Vector Machines) para la tarea de detección de fallas. En (Zhang, y otros, 2008) presentan una técnica donde combinan árboles de decisión, el algoritmo K-medias y algoritmos genéticos para el otorgamiento de créditos bancarios.

1.1. Tipos de datos

La minería de datos puede aplicarse a cualquier tipo de información, siendo las técnicas de minería diferentes para cada una de ellas. Existen muchos tipos de datos (enteros, reales, fechas, cadenas de texto, etc.) y desde el punto de vista de las técnicas de minería de datos más habituales solo interesa distinguir entre dos tipos: numéricos (enteros o reales) y categóricos o discretos (toman valores en un conjunto finito de categorías). Incluso considerando solo estos dos tipos de datos, se debe aclarar que no todas las técnicas son capaces de trabajar con ambos tipos.

Estos datos están contenidos en lo que se conoce como base de datos, las cuales pueden ser de diferente naturaleza dependiendo el tipo de información que almacenen. Algunos tipos de bases de datos son:

- Las bases de datos relacionales son las más utilizadas hoy en día como fuente para las técnicas de minería de datos. Una base de datos relacional es una colección de relaciones (tablas) donde cada tabla consta de un conjunto de atributos y puede tener un gran número de tuplas, registros o filas. Cada tupla representa un objeto, el cual se describe a través de los valores de sus atributos, y por lo general, se caracteriza por poseer una clave única que lo identifica unívocamente del resto. Una de las principales características de las bases de datos relacionales es la existencia de un esquema asociado, es decir, los datos deben seguir una estructura y son, por lo tanto, estructurados. Mediante una consulta (por ejemplo en SQL) podemos combinar en una sola tabla la información de varias tablas que requiramos para cada tarea concreta de minería de datos.

- Las bases de datos espaciales contienen información relacionada con el espacio físico en un sentido amplio (una ciudad, una región montañosa, un atlas cerebral, etc.). Estas bases de datos incluyen datos geográficos, imágenes médicas, redes de transporte o información de tráfico, etc. donde las relaciones espaciales son muy relevantes. La minería de datos sobre estas bases de datos permite encontrar patrones entre los datos, como por ejemplo las características de las casas en una zona montañosa, la planificación de nuevas líneas de transporte público en función de la distancia de las distintas áreas a las líneas existentes, etc.

- Las bases de datos temporales almacenan datos que incluyen muchos atributos relacionados con el tiempo o en el que éste sea muy relevante. Estos atributos pueden referirse a distintos instantes o intervalos temporales. En este tipo de bases de datos las técnicas de minería de datos pueden utilizarse para encontrar las características de la evolución o las tendencias del cambio de distintas medidas o valores de la base de datos.

- Las bases de datos documentales contienen descripciones para los objetos (documentos de texto) que pueden ir desde las simples palabras clave a los resúmenes. Estas bases de datos pueden contener documentos no estructurados (biblioteca digital de novelas), semi-estructurados (documentos con índices o tablas de contenido) o estructurados (fichas bibliográficas). Las técnicas de minería de datos pueden utilizarse para obtener asociaciones entre los contenidos y, agrupar o clasificar objetos textuales. Para ello, los métodos de minería se integran con otras técnicas de recuperación de información y con la construcción o uso de jerarquías específicas para datos textuales, como los diccionarios y los tesauros.

- Las bases de datos multimedia almacenan imágenes, audio y video. Soportan objetos de gran tamaño ya que, por ejemplo, los videos pueden necesitar varios gigabytes de capacidad para su almacenamiento. Para la minería de estas bases de datos también es necesario integrar los métodos de minería con técnicas de búsqueda y almacenamiento.

- La World Wide Web es el repositorio de información más grande y diverso de los existentes en la actualidad. Por ello, hay gran cantidad de datos en la web de los que se puede extraer conocimiento relevante y útil. Éste es precisamente el reto al que se enfrenta la minería web. Minar la web no es un problema sencillo, debido a que muchos de los datos son no estructurados o semi-estructurados, a que muchas páginas web contienen datos multimedia y a que estos datos pueden residir en diversos servidores. Otros aspectos que dificultan la minería web son cómo determinar a qué páginas debemos acceder y cómo seleccionar la información que va a ser útil para extraer conocimiento. Toda esta diversidad hace que la minería web se organice en torno a tres categorías, minería del contenido, minería de la estructura y minería del uso que hace el usuario durante la navegación.

1.2. Tipos de modelos

La minería de datos tiene como objetivo analizar los datos para extraer conocimiento. Este conocimiento puede ser en forma de relaciones, patrones o reglas inferidos de los datos, o bien en forma de una descripción más concisa (resumen). Estas relaciones o resúmenes constituyen el modelo de los datos analizados. Existen muchas formas diferentes de representar los modelos y cada una de ellas determina el tipo de técnica que puede usarse para inferirlos.

En la práctica, los modelos pueden ser de dos tipos: predictivos y descriptivos. Los modelos predictivos pretenden estimar valores futuros o desconocidos de variables de interés, que se denominan variables objetivo, usando otras variables de la base de datos, a las que se conocen como variables independientes. Los modelos descriptivos, en cambio, identifican patrones que explican o resumen los datos, es decir, sirven para explorar las propiedades de los datos examinados, no para predecir nuevos datos.

Cada tarea puede ser realizada usando distintas técnicas. Por ejemplo, los modelos inferidos por los árboles de decisión y las redes neuronales pueden inferir modelos predictivos. Igualmente, para una misma técnica se han desarrollado diferentes algoritmos que difieren en la forma y criterios concretos con los que se construye el modelo.

2. Extracción de conocimiento

La minería de datos no es más que un paso esencial de un proceso más amplio cuyo objetivo es el descubrimiento de conocimiento en bases de datos. Este proceso consta de una secuencia iterativa de etapas o fases: preparación de datos, minería de datos, evaluación, difusión y uso de modelos.

La extracción de conocimiento es un proceso iterativo ya que la salida de alguna de las fases puede hacer volver a pasos anteriores y porque a menudo son necesarias varias iteraciones para extraer conocimiento de alta calidad. Es interactivo porque el usuario o un experto en el dominio del problema debe ayudar en la preparación de los datos, validación del conocimiento extraído, etc.

El proceso de extracción de conocimiento se organiza en torno a cinco fases como se muestra en la Figura 1-1. En la fase de integración y recopilación de datos se determinan las fuentes de información que pueden ser útiles y donde conseguirlas. A continuación, se transforman todos los datos a un formato común, frecuentemente mediante un almacén de datos que consiga unificar de manera operativa toda la información recogida, detectando y resolviendo las inconsistencias. Este almacén de datos facilita enormemente la navegación y visualización previa de sus datos, para discernir qué aspectos puede interesar que sean estudiados. Dado que los datos provienen de diferentes fuentes, pueden contener valores erróneos o faltantes. Estas situaciones se tratan en la fase de selección, limpieza y transformación, en la que se eliminan o corrigen los datos incorrectos y se decide la estrategia a seguir con los datos incompletos. Además, se proyectan los datos para considerar únicamente aquellas variables o atributos que van a ser relevantes con el objetivo de hacer más fácil la tarea propia de minería y para que los resultados de la misma sean más útiles. La selección incluye tanto una fusión horizontal (filas o registros) como vertical (atributos). Las dos primeras fases se suelen englobar bajo el nombre de preparación de datos. En la fase de minería de datos, se decide cuál es la tarea a realizar (clasificar, agrupar, etc.) y se elige el método que se va a utilizar. En la fase de evaluación e interpretación se evalúan los patrones y se analizan por los expertos y, si es necesario, se vuelve a las fases anteriores para una nueva

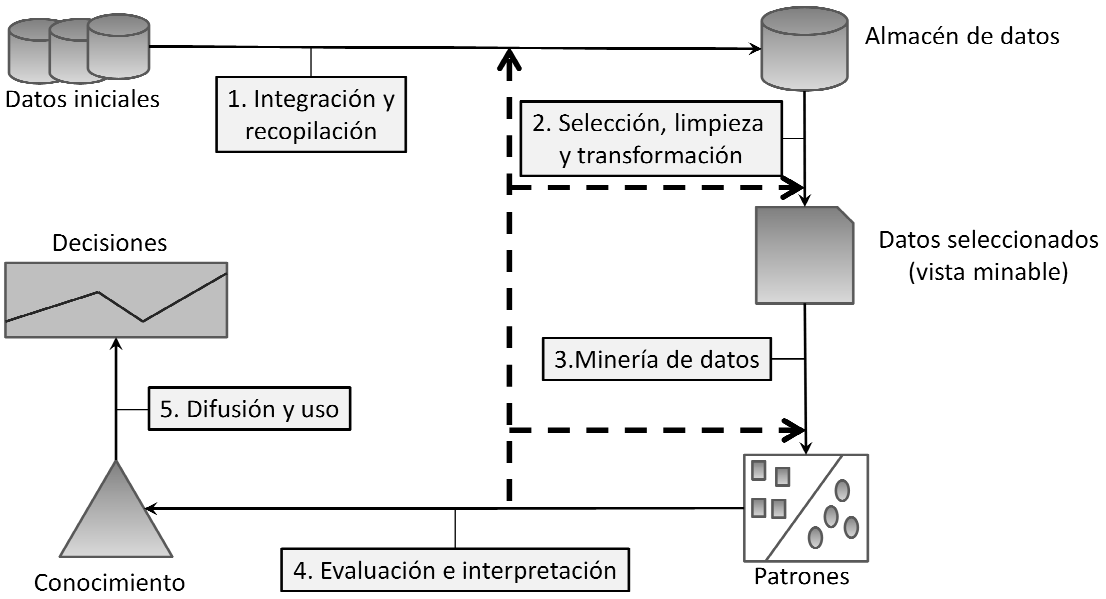

Figura 1-1. Fases del proceso de extracción de conocimiento en bases de datos.

iteración. Esto incluye resolver posibles conflictos con el conocimiento que se disponía anteriormente. Finalmente, en la fase de difusión se hace uso del nuevo conocimiento y se hace partícipe de él a todos los posibles usuarios.

2.1. Fase de integración y recopilación

Lo normal es que los datos necesarios para poder llevar a cabo un proceso de extracción de conocimiento pertenezcan a diferentes organizaciones o a distintos departamentos de una misma entidad. Incluso puede ocurrir que algunos datos necesarios para el análisis nunca hayan sido recolectados en el ámbito de la organización por no ser necesarios para sus aplicaciones. En muchos casos, además, puede que haya que adquirir datos externos desde bases de datos públicas o desde bases de datos privadas. Esta no es una tarea sencilla, ya que cada fuente de datos usa diferentes formatos de registro, diferentes grados de agregación de los datos, diferentes claves primarias, diferentes tipos de error, etc. Lo primero por lo tanto, es integrar estos datos. La idea de la integración de múltiples bases de datos ha dado lugar a la tecnología de almacenes de datos (data warehousing). Este término, hace referencia a la tendencia actual en las empresas e instituciones de coleccionar datos de las bases de datos transaccionales y otras fuentes diversas para hacerlos accesibles para el análisis y la toma de decisiones.

El primer paso en el proceso de extracción de conocimiento a partir de datos es precisamente reconocer y reunir los datos con los que se va a trabajar. Si esta recopilación se va a realizar para una tarea puntual y no involucra muchas cantidades y variedades de datos, es posible que el sentido común sea suficiente para obtener un conjunto de datos con la calidad suficiente para poder empezar a trabajar. En cambio, si se requieren datos de distintas fuentes, tanto externas como internas a la organización, con datos complejos y variados, posiblemente en grandes cantidades y además cambiantes, con los que se desee realizar a medio o largo plazo diversas tareas de minería de datos, es posible que el sentido común no sea suficiente para hacer una recopilación e integración en condiciones.

Al igual que la tecnología de bases de datos ha desarrollado una serie de modelos de datos (como el relacional), de lenguajes de consulta y actualización, de reglas de actividad, etc., para trabajar con la información transaccional de una organización, existe una tecnología relativamente reciente, denominada "almacenes de datos" (data warehouses) que pretende proporcionar metodologías y tecnología para recopilar e integrar los

datos históricos de una organización, cuyo fin es el análisis, la obtención de resúmenes e informes complejos y la extracción de conocimiento. Esta tecnología está diseñada especialmente para organizar grandes volúmenes de datos de procedencia generalmente estructurada (bases de datos relacionales, por ejemplo), aunque el concepto general es útil para la organización de pequeños conjuntos de datos en aplicaciones de minería de datos más modestas.

2.2. Fase de selección, limpieza y transformación

La calidad del conocimiento descubierto no solo depende del algoritmo de minería utilizado, sino también de la calidad de los datos minados. Por ello, después de la recopilación, el siguiente paso en el proceso de la extracción de conocimiento es seleccionar y preparar el subconjunto de datos que se va a minar, los cuales constituyen lo que se conoce como vista minable. Este paso es necesario ya que algunos datos coleccionados en la fase anterior son irrelevantes o innecesarios para la tarea de minería que se desea realizar.

Pero además de la irrelevancia, existen otros problemas que afectan a la calidad de los datos. Uno de estos problemas es la presencia de valores que no se ajustan al comportamiento general de los datos. Estos datos anómalos pueden representar errores en los datos o pueden ser valores correctos que son simplemente diferentes a los demás. Algunos algoritmos de minería de datos ignoran estos datos, otros los descartan considerándolos ruido o excepciones, pero otros son muy sensibles y el resultado se ve claramente perjudicado por ello. Sin embargo, no siempre es conveniente eliminarlos, ya que en algunas aplicaciones como la detección de compras fraudulentas por tarjeta de crédito o la predicción de inundaciones, los eventos raros pueden ser más interesantes que los regulares.

La recopilación de datos debe ir acompañada de una limpieza e integración de los mismos, para que éstos estén en condiciones para su análisis. Los beneficios del análisis y de la extracción de conocimiento a partir de datos dependen, en gran medida, de la calidad de los datos recopilados. Además, generalmente, debido a las características propias de las técnicas de minería de datos, es necesario realizar una transformación de los datos para obtener una "materia prima" que sea adecuada para el propósito concreto y las técnicas que se quieren emplear. En definitiva, el éxito de un proceso de minería de datos depende, no sólo de tener todos los datos necesarios, sino de que éstos estén íntegros, completos y consistentes.

2.2.1. Limpieza y transformación

En la mayoría de bases de datos existe mucha información que es incorrecta respecto al dominio de la realidad que se desea cubrir y un número menor, pero a veces también relevante, de datos inconsistentes. Estos problemas se acentúan cuando se realiza la integración de distintas fuentes. No obstante, mientras los datos erróneos crecen de manera lineal respecto al tamaño de los datos recopilados, los datos inconsistentes se multiplican. Un aspecto muy importante a la hora de realizar los procesos de integración, limpieza y transformación, es que se debe conocer el dominio de donde provienen los datos.

La transformación de datos engloba, en realidad, cualquier proceso que modifique la forma de los datos. Prácticamente todos los procesos de preparación de datos involucran algún tipo de transformación. Existen distintas operaciones que transforman atributos, algunas transforman un conjunto de atributos en otros, o bien derivan nuevos atributos, o bien cambian el tipo (mediante numerización o discretización) o el rango (mediante escalado).

La alta dimensionalidad es, muchas veces, un gran problema a la hora de aprender de los datos. Si existen muchas dimensiones (atributos) respecto a la cantidad de instancias o ejemplos, se estará en presencia de una situación poco deseable al existir tantos grados de libertad, los patrones extraídos pueden ser caprichosos y poco robustos. Visualmente, se está en presencia de un "espacio" prácticamente vacío y, por lo tanto, los patrones no tienen datos donde apoyarse a la hora de tomar una u otra forma. Este problema se conoce popularmente como "la maldición de la dimensionalidad" ("the curse of dimensionality"). Aunque este es el problema mayor y es motivación suficiente para intentar reducir la dimensionalidad, especialmente si se tienen pocos ejemplos, existen otros problemas, como por ejemplo, la eficiencia y la dificultad de representación de los datos (en principio, sólo podemos representar visualmente tres dimensiones).

La reducción de la dimensionalidad, se puede realizar por selección de un subconjunto de atributos o bien se puede realizar por transformación, sustituyendo el conjunto de atributos iniciales por otros diferentes que, geométricamente, se denomina proyección. Existen muchas técnicas para realizar este tipo de proyección (o que pueden ayudar en este propósito): análisis de componentes principales, algunas técnicas de análisis factorial, uso de mapas auto-organizativos, etc.

La presencia de datos faltantes o perdidos puede ser también un problema que conduzca a resultados poco precisos. No obstante, es necesario reflexionar primero sobre el significado de los valores faltantes antes de tomar ninguna decisión sobre cómo tratarlos ya que estos pueden deberse a causas muy diversas, como a un mal funcionamiento del dispositivo que hizo la lectura del valor, a cambios efectuados en los procedimientos usados durante la colección de los datos o al hecho de que los datos se recopilen desde fuente diversas. Por ello, existen muchas aproximaciones para manejar los datos faltantes.

La selección de atributos relevantes es uno de los pre-procesamientos más importantes, ya que es crucial que los atributos utilizados sean relevantes para la tarea de minería de datos. Idealmente, se podría usar todas las variables y dejar que la herramienta de minería de datos fuera probando hasta elegir las mejores variables predictoras. Obviamente, esta forma de trabajar no funciona bien, entre otras cosas porque el tiempo requerido para construir un modelo crece con el número de variables. Varios trabajos se han propuesto para resolver estos problemas (Somol, y otros, 1999) (Maji, y otros, 2010) (Zeng, y otros, 2010) (Hou, y otros, 2010) (Parthaláin, y otros, 2010).

Otra tarea de preparación de los datos es la construcción de atributos, la cual consiste en construir automáticamente nuevos atributos aplicando alguna operación o función a los atributos originales con objeto de que estos nuevos atributos hagan más fácil el proceso de minería. La motivación principal para esta tarea es fuerte cuando los atributos originales no tienen mucho poder predictivo por si solos o los patrones dependen de variaciones lineales de las variables originales.

Uno de los aspectos más importantes, sino el que más, de un atributo es el tipo. El hecho de que un atributo sea nominal o numérico determina en gran medida, cómo va a ser tratado por las herramientas de minería de datos. En algunas situaciones puede ser conveniente convertir un numérico a nominal (discretización) o viceversa (numerización) para facilitar el uso de técnicas que requieren tipos de datos específicos. Así, algunos atributos se pueden numerizar, lo que reduce el espacio y permite técnicas numéricas. El proceso inverso consiste en discretizar los atributos continuos, es decir, transformar valores numéricos en atributos discretos o nominales. Los atributos discretizados pueden tratarse como atributos categóricos con un número más pequeño de valores. La idea básica es partir los valores de un atributo continuo en una pequeña lista de intervalos, tal que cada intervalo es visto como un valor discreto del atributo.

2.2.1.1. Discretización

La discretización o cuantización es la conversión de un valor numérico en un valor nominal ordenado que representa un intervalo. No obstante, el orden del atributo nominal puede ser preservado y utilizado por los pasos subsiguientes o bien puede olvidarse y tratarse el atributo como un valor nominal sin orden.

La discretización, en general, se realiza cuando el error en la medida puede ser grande o existen ciertos umbrales significativos. Existen más razones para discretizar como, por ejemplo, el hecho de que las diferencias en ciertas zonas del rango de valores sean más importantes que en otras, o dicho más técnicamente, cuando la interpretación de la medida no sea lineal. Una última razón es cuando se tiene algunos atributos nominales y otros numéricos y se desea que todos sean nominales para, por ejemplo, establecer reglas de asociación.

Si bien la discretización de una variable puede ser realizada de manera arbitraria visualizando los valores de esa variable, se han presentado trabajos que realizan esta tarea de manera automática (Hua, y otros, 2009) (Hu, y otros, 2009).

2.2.1.2. Numerización

La numerización es el proceso inverso a la discretización. Aunque es menos común que la discretización, también existen casos donde puede ser extremadamente útil. Un primer caso donde la numerización es útil es cuando el método de minería de datos que se va a utilizar no admite datos nominales. Un claro ejemplo, es si se pretende utilizar regresión lineal para obtener la influencia de cada "factor". En general, es preciso para mucho de los métodos de modelización estadística, como la regresión logística, el análisis ANOVA, los modelos log-linear, los discriminantes paramétricos o no paramétricos, etc.

En todos estos casos lo que se suele hacer es lo que se denomina numerización "1 a *n*" que consiste en la creación de varias variables indicadoras. Si una variable nominal tiene *n* posibles valores se crean *n* variables numéricas, con valores 0 o 1 dependiendo de si la variable toma ese valor o no.

2.2.1.3. Normalización de rango: escalado y centrado

En general, en muchos métodos de minería de datos no es necesario centrar los datos ni escalar, es decir normalizar el rango de un valor numérico. De hecho, cuando se usan métodos como los árboles de decisión o los sistemas de reglas, que son comprensibles, escalar previamente hace que los modelos resultantes sean más difíciles de interpretar, ya que hay que invertir el escalado final.

Sin embargo, en otros casos, es necesario normalizar todos los atributos al mismo rango, como por ejemplo en los algoritmos basados en distancias, ya que las distancias debidas a diferencias de un atributo que va entre 0 y 100 serán mucho mayores que las distancias debidas a diferencias de un atributo que va entre 0 y 1. La normalización más común es la normalización lineal uniforme y se normaliza a una escala genérica entre cero y uno utilizando la siguiente fórmula:

$$v' = \frac{v - min}{max - min}$$

2.2.2. Exploración y selección

Una vez que los datos están recopilados, integrados y limpios, aún no se está en condiciones de realizar la tarea de minería de datos. Es necesario, además, realizar un reconocimiento o análisis exploratorio de los datos con el objetivo de conocerlos mejor de cara a la tarea de minería de datos. Incluso esta fase es imprescindible cuando se realiza minería de datos "abierta", ya que tenemos todo el volumen de datos pero hemos de determinar los datos a seleccionar y las tareas a realizar sobre esos datos.

Hace ya algunos años, apareció el término "minería de datos visual" (visual data mining) (Wong, 1999) con el significado de una minería de datos que se realiza manejando e interactuando con gráficos. El concepto de "minería de datos visual" es interesante como híbrido entre la minería de datos y la visualización de datos más tradicional (Cleveland, 1993), pero, en general, no se puede hacer minería de datos sólo con gráficas. Precisamente lo que caracteriza la minería de datos de técnicas anteriores o de la perspectiva más clásica del análisis de datos es que los modelos son extraídos por algoritmos y, por tanto, no son vistos o descubiertos visualmente por el usuario (y posteriormente simplemente validados estadísticamente).

La salida o resultado de las técnicas de la exploración es una "vista minable tipada" con instrucciones sobre qué datos trabajar, qué tarea realizar y de qué manera obtener el conocimiento. Una vista minable (Ng, y otros, 1998) consiste en una vista en el sentido más clásico de base de datos; una tabla. La mayoría de métodos de minería de datos son solo capaces de tratar una tabla en cada tarea. Por lo tanto, la vista minable ha de recoger toda la información necesaria para realizar la tarea de minería de datos.

Las técnicas de visualización de datos se utilizan fundamentalmente con dos objetivos:

- Aprovechar la gran capacidad humana de ver patrones, anomalías y tendencias a partir de imágenes y facilitar la compresión de los datos.
- Ayudar al usuario a comprender más rápidamente patrones descubiertos automáticamente por un sistema de extracción de conocimiento.

Estos dos objetivos marcan dos momentos diferentes del uso de la visualización de los datos:

- Visualización previa: ésta es la que normalmente recibe el nombre de minería de datos visual. Se utiliza para entender mejor los datos y sugerir posibles patrones o qué tipo de herramienta de minería de datos utilizar. La visualización previa se utiliza frecuentemente para ver tendencias y resúmenes de los datos.

- Visualización posterior al proceso de minería de datos: se utiliza para mostrar los patrones y entenderlos mejor. La visualización posterior se utiliza frecuentemente para validad y mostrar a los expertos los resultados de la extracción de conocimiento.

2.3. Fase de minería de datos

La fase de minería de datos es la más característica del proceso de extracción de conocimiento y muchas veces se utiliza esta fase para nombrar todo el proceso. El objetivo de esta fase es producir nuevo conocimiento que pueda utilizar el usuario. Esto se realiza construyendo un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender mejor los datos o para explicar situaciones pasadas. Para ello es necesario tomar una serie de decisiones antes de empezar el proceso:

- Determinar qué tipo de tarea de minería es el más apropiado.

- Elegir el tipo de modelo.

- Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando.

En la construcción del modelo es donde se ve mejor el carácter iterativo del proceso de minería de datos, ya que será necesario explorar modelos alternativos hasta encontrar aquel que resulte más útil para resolver el problema. Así, una vez obtenido un modelo y a partir de los resultados obtenidos para el mismo, se puede construir otro modelo usando la misma técnica pero otros parámetros, o quizás usar otras técnicas o herramientas.

El proceso de construcción de modelos predictivos requiere tener bien definidas las etapas de entrenamiento y validación para asegurar que las predicciones serán robustas y precisas. La idea básica es estimar el modelo con una porción de los datos (datos de entrenamiento) y luego validarlo con el resto de los datos (datos de testeo).

La extracción de conocimiento a partir de datos tiene como objetivo descubrir patrones que, entre otras cosas, deben ser válidos, novedosos, interesantes y comprensibles. Sea como sea la presentación del problema, una de las características en cualquier tipo de técnica de minería de datos es su carácter hipotético, es decir, lo aprendido puede, en cualquier momento, ser refutado por evidencia futura. En muchos casos, los modelos no aspiran a ser modelos perfectos, sino modelos aproximados. Dado cualquier método de aprendizaje (por muy bueno que sea), siempre existirán problemas que son resueltos mejor por otros métodos.

En el proceso de extracción de conocimiento, existen diferentes tareas y diferentes métodos, estrategias o técnicas. Las tareas más importantes dentro de la minería de datos son las predictivas y las descriptivas. En la literatura es común mencionar a las primeras como técnicas de entrenamiento supervisado y a las descriptivas como técnicas de entrenamiento no-supervisado. Si bien, la gran mayoría de las técnicas se pueden encuadrar en una u otra categoría, existen técnicas híbridas que utilizan características de ambas en lo que se conoce como aprendizaje semi-supervisado (Hasperué, y otros, 2006) (Zhang, y otros, 2011), incluso existen técnicas que además de presentar un aprendizaje semi-supervisado pueden ser utilizadas en diferentes tareas (Ni, y otros, 2012).

2.3.1. Tareas predictivas

Se trata de problemas y tareas en los que hay que predecir uno o más valores para uno o más ejemplos. Los ejemplos en la evidencia van acompañados de una salida (clase, categoría o valor numérico) o un orden entre

ellos. Dependiendo de cómo sea la correspondencia entre los ejemplos y los valores de salida y la presentación de los ejemplos es posible definir varias tareas predictivas. Los posibles elementos de entrada pertenecen a un conjunto *E* de datos. Cada instancia *e* del conjunto *E* representa un conjunto de *n* atributos (nominales o numéricos).

- Clasificación: los ejemplos se presentan como un conjunto de pares de elementos de dos conjuntos, el conjunto *E* con los valores de entrada y el conjunto *S* con los valores de salida, donde los ejemplos se construyen como pares $<e, s>$, $e \in E$, $s \in S$. El objetivo es aprender una función $\lambda: E \rightarrow S$ denominada clasificador, que represente la correspondencia existente entre los ejemplos. La función aprendida será capaz de determinar la clase para cada nuevo ejemplo sin etiquetar.

- Clasificación suave: la presentación del problema es la misma que la de la clasificación, pares de conjuntos de entrada y salida. Además de la función $\lambda: E \rightarrow S$ se aprende otra función $\theta: E \rightarrow \mathbb{R}$ que significa el grado de certeza de predicción hacha por la función λ.

- Categorización: no se trata de aprender una función, sino una correspondencia. Además de aprender $\lambda: E \rightarrow S$ se puede asignar varias categorías a un mismo elemento del conjunto *E*, a diferencia de la clasificación que solo asigna una y solo una. Dicho de otra manera, un ejemplo podría tener varias categorías asociadas.

- Preferencias o priorización: el aprendizaje de preferencias consiste en determinar a partir de dos o más ejemplos, un orden de preferencia. La definición formal de un ejemplo es una secuencia de *k* elementos de *E*, donde el orden de la secuencia representa la predicción. Una simplificación del problema sería obtener sólo preferencias entre dos elementos. El modelo aprendido sólo será capaz de decir cual prefiere entre dos elementos de *E*.

- Regresión: El conjunto de evidencias son correspondencias entre dos conjuntos de entrada y salida. Al igual que con la clasificación, los ejemplos van acompañados con una etiqueta. El objetivo es aprender una función que represente la correspondencia existente en los ejemplos, es decir, para cada valor de *E* tenemos un único valor para *S*. La diferencia respecto a la clasificación es que S es numérico.

2.3.2. Tareas descriptivas

Los ejemplos se presentan como un conjunto *E* sin etiquetar ni ordenar de ninguna manera. El objetivo, por tanto, no es predecir nuevos datos sino describir los existentes. Lógicamente, esto se puede hacer de muchas maneras y la variedad de tareas se dispara.

- Agrupamiento (clustering): el objetivo de esta tarea es obtener grupos o conjuntos entre los elementos de *E*, de tal manera que los elementos asignados al mismo grupo sean similares. Lo importante del agrupamiento respecto a la clasificación que son precisamente los grupos y la pertenencia a los grupos lo que se quiere determinar y, a priori, ni cómo son los grupos ni cuantos hay. En algunos casos se puede proporcionar el número de grupos que se desea obtener. Otras veces este número determina por el algoritmo de agrupamiento según la característica de los datos. La función a obtener es la misma que la de la clasificación, $\lambda: E \rightarrow S$, con la diferencia de que los valores de S y sus miembros se "inventan", durante el proceso de aprendizaje. La clave en este tipo de tareas es encontrar una métrica adecuada para medir las distancias entre los datos y así formar los clusters más acordes para una solución que resulte óptima. Si bien la distancia euclídea es una de las más utilizadas, se han propuesto otras alternativas (Sun, y otros, 2011) (Davis, y otros, 2007) (Beyer, y otros, 1999) (Aggarwal, y otros, 2001).

- Correlaciones y factorizaciones: el objetivo de esta tarea es la de ver la relevancia de atributos, detectar atributos redundantes o dependencias entre atributos. Los estudios correlaciónales y factoriales se centran exclusivamente en los atributos numéricos. El objetivo es ver si dos ejemplos del conjunto *E* están correlacionados linealmente o relacionados de algún otro modo.

- Reglas de asociación: el objetivo, en cierto modo, es similar a los estudios correlaciónales y factoriales, pero para los atributos nominales muy frecuentes en las bases de datos. Dados dos ejemplos del conjunto *E* una regla de asociación se define generalmente de la forma "si $A_1=v_1$ y $A_2=v_2$ y … y $A_k=v_k$ entonces $A_r=v_r$ y $A_s=v_s$ y … y $A_z=v_z$" donde todos los atributos son nominales y las igualdades se definen utilizando algún valor de los posibles para cada atributo.

2.3.3. Técnicas

Cada una de las tareas anteriores como cualquier problema, requiere métodos, técnicas, estrategias o algoritmos para resolverlas.

- Técnicas algebraicas y estadísticas: se basan, generalmente, en expresar modelos y patrones mediante fórmulas algebraicas, funciones lineales, funciones no lineales, distribuciones o valores agregados estadísticos tales como medias, varianzas, correlaciones, etc. Frecuentemente, estas técnicas, cuando obtienen un patrón, lo hacen a partir de un modelo ya predeterminado del cual, se estiman unos coeficientes o parámetros, de ahí el nombre de técnicas paramétricas.

- Técnicas bayesianas: se basan en estimar la probabilidad de pertenencia (a una clase o grupo) mediante la estimación de las probabilidades condicionales inversas o a priori, utilizando para ello el teorema de Bayes. Algunos algoritmos muy populares son el clasificador bayesiano naive, los métodos basados en máxima verosimilitud y el algoritmo EM.

- Técnicas basadas en conteo de frecuencias y tablas de contingencia: estas técnicas se basan en contar la frecuencia en la que dos o más sucesos se presenten conjuntamente. Cuando el conjunto de sucesos posibles es muy grande, existen algoritmos que van comenzando por pares de sucesos e incrementando los conjuntos solo en aquellos casos que las frecuencias conjuntas superen un cierto umbral. Ejemplos de estos algoritmos son el algoritmo Apriori y similares.

- Técnicas basadas en árboles de decisión y sistemas de aprendizajes de reglas: son técnicas que, además de su representación en forma de reglas, se basan en dos tipos de algoritmos: los algoritmos denominados "divide y vencerás", como el ID3/C4.5 (Quinlan, 1993) o el CART (Breiman, y otros, 1984), y los algoritmos denominados "separa y vencerás", como el CN2. En la sección 0 se ven con más detalle la técnica de árboles de decisión.

- Técnicas relacionales, declarativas y estructurales: la característica principal de este conjunto de técnicas es que representan los modelos mediante lenguajes declarativos, como los lenguajes lógicos, funcionales o lógico-funcionales. Las técnicas de ILP (programación lógica inductiva) son las más representativas y las que han dado nombre a un conjunto de técnicas denominadas minería de datos relacional.

- Técnicas basadas en redes neuronales artificiales: se trata de técnicas que aprenden un modelo mediante el entrenamiento de los pesos que conectan un conjunto de nodos o neuronas. La topología de la red y los pesos de las conexiones determinan el patrón aprendido. Existen innumerables variantes de organización: perceptrón simple, redes multicapa, redes de base radial, redes de Kohonen, etc.

- Técnicas basadas en núcleo y máquinas de soporte vectorial: se trata de técnicas que intentan maximizar el margen entre los grupos o las clases formadas. Para ello se basan en unas transformaciones que pueden aumentar la dimensionalidad. Estas transformaciones se llaman núcleos (kernels). Existen muchas variantes dependiendo del núcleo utilizado de la manera de trabajar con el margen (He, y otros, 2002).

- Técnicas estocásticas y difusas: bajo esta denominación se incluyen la mayoría de las técnicas que, junto con las redes neuronales, forman lo que se denomina computación flexible (soft computing). Son técnicas en las que o bien los componentes aleatorios son fundamentales, como el simulated annealing, los métodos evolutivos y genéticos, o bien al utilizar funciones de pertenencia difusas (Wenjun, y otros, 2010).

- Técnicas basadas en casos, en densidad o distancia: son métodos que se basan en distancias al resto de elementos, ya sea directamente, como los vecinos más próximos (los casos más similares), de una manera más sofisticada, mediante la estimación de funciones de densidad. Además de los vecinos más próximos, algunos algoritmos muy conocidos son los jerárquicos, como Two-step o COBWEB, y los no jerárquicos, como K-medias.

También existen varias aproximaciones que utilizan un sistema híbrido de técnicas. Por ejemplo en (Filippi, y otros, 2007) utilizan LVQ (learning vector quantization) junto con lógica difusa para la clasificación de imágenes espectrales. Junto con un método mejorado de LVQ incremental en (Wang, y otros, 2009) presentan una estrategia para la clasificación de textos.

2.3.4. Aprendizaje inductivo

Tantas tareas y métodos dan la impresión de estar tratando con problemas inconexos. Un aspecto chocante es que la misma técnica pueda utilizarse para varias tareas. Esto debe querer decir, que hay algunos procesos útiles para una tarea que también lo es, generalmente, para otras, con una ligera adaptación. Esto en realidad es así ya que todas las tareas (exceptuando quizá las reglas de asociación y las correlaciones) y los métodos se centran alrededor de la idea del aprendizaje inductivo.

El aprendizaje inductivo es un tipo especial de aprendizaje (el más importante, no obstante) que parte de casos particulares (ejemplos) y obtienen casos generales (reglas o modelos) que generalizan o abstraen la evidencia.

Además, el aprendizaje puede ser incremental o no, dependiendo de si los datos se van presentando poco a poco, o si se tienen todos desde el principio. En el caso de la minería de datos se suele considerar un aprendizaje no incremental, ya que los datos se obtienen de una base de datos o un almacén de datos. No obstante, si se considera que los datos pueden ir cambiando o incrementándose a lo largo del tiempo, podría ser interesante utilizar métodos específicos para el aprendizaje incremental, que permite revisar los modelos aprendidos y no tener que realizarlos de nuevo con todos los datos.

Dependiendo de si se puede actuar sobre las observaciones, el aprendizaje puede ser interactivo o no. En el caso de la minería de datos generalmente se tienen datos históricos que no es posible afectar, con lo que no se puede actuar sobre las observaciones. El aprendizaje interactivo es útil cuando se puede interaccionar con un entorno para generar nuevas observaciones y así facilitar la tarea de aprendizaje. Este tipo de aprendizaje puede empezar a ser relevante para la minería de datos web o minería de datos en entornos software, donde es posible preguntar a los usuarios por sus preferencias. Un tipo especial de aprendizaje interactivo es el aprendizaje por refuerzo (Sutton, y otros, 1998), en el que un agente va realizando una serie de tareas, que si son acertadas son recompensadas positivamente (premio) y si son desacertadas se penalizan (castigo).

2.3.5. Grandes bases de datos

En los últimos años la tecnología ha permitido almacenar un gran volumen de información, ante este echo, en muchas disciplinas se ha podido formar enormes bases de datos. Manejar un gran volumen de información no es una tarea trivial, no solo porque la base de datos completa podría no caber en memoria para ser analizada sino también porque esta podría estar físicamente distribuida.

Ante estos problemas se han propuesto diversas técnicas que apuntan a resolver tareas de minería de datos a partir de un gran volumen de información (Zhang, y otros, 1996) (Ramaswamy, y otros, 2010). En (Chu, y otros, 2010) realizan un cálculo de densidades de regiones para generar los clusters y mediante el manejo de diferentes umbrales lograr un número óptimo de clusters. Utilizando un algoritmo híbrido entre K-medias y PSO (Particle Swarm Optimization) en (van der Merwe, y otros, 2003) proponen una técnica para realizar clustering sobre grandes bases de datos, en (Cui, y otros, 2005) presentan un algoritmo para poder utilizar esta técnica en clustering de documentos y en (Omran, y otros, 2006) lo utilizan para realizar clustering de imágenes.

2.4. Fase de evaluación e interpretación

Los métodos de aprendizaje permiten construir modelos a partir de un conjunto de datos o evidencia. En la mayoría de los casos es necesario evaluar la calidad de las hipótesis de la manera más exacta posible. Por

ejemplo, si en el ámbito de aplicación de un modelo un error en la predicción causa importantes consecuencias (por ejemplo detección de células cancerígenas), es importante conocer con exactitud el nivel de precisión de los modelos aprendidos.

Por lo tanto, la etapa de evaluación de modelos es crucial para la aplicación real de las técnicas de minería de datos. Sin embargo establecer medidas justas y exhaustivas no es tarea sencilla. Una primera aproximación lleva a utilizar el propio conjunto de datos de entrenamiento como referencia para evaluar la calidad de un modelo. Sin embargo, esta aproximación es del todo equivocada, ya que premia los modelos que se ajustan más al conjunto de entrenamiento, por lo que favorecen los modelos que sobreajustan el conjunto de datos de entrenamiento y no generalizan para otros datos.

Consecuentemente, una mejor opción es evaluar los modelos sobre un conjunto de datos diferente al conjunto de entrenamiento.

Medir la calidad de los patrones descubiertos por un algoritmo de minería de datos no es un problema trivial, ya que esta medida puede corresponder a varios criterios, alguno de ellos bastante subjetivos. Idealmente, los patrones descubiertos deben tener tres cualidades: ser precisos, comprensibles e interesantes. Según las aplicaciones puede interesar mejorar algún criterio y sacrificar ligeramente otro.

2.4.1. Técnicas de evaluación

En los modelos predictivos, el uso de la separación de conjunto de datos de entrenamiento y de testeo es fácil de interpretar. Por ejemplo, para una tarea de clasificación, después de generar el modelo con el conjunto de entrenamiento, éste se puede usar para predecir la clase de los datos de prueba. Entonces, la razón de precisión se obtiene dividiendo el número de clasificaciones correctas por el número total de instancias. La precisión es una buena estimación de cómo se comportará el modelo de datos futuros similares a los del test. Esta forma de proceder no garantiza que el modelo sea correcto, sino que simplemente indica que si usamos la misma técnica con una base de datos con datos similares a los de prueba, la precisión media será bastante parecida a la obtenida con éstos.

2.4.1.1. Validación simple

El método de evaluación más básico reserva un porcentaje de la base de datos como conjunto de prueba y no la usa para construir el modelo. Este porcentaje suele variar entre el 5% y el 50%. La división de los datos en estos grupos debe ser aleatoria para que la estimación sea correcta.

2.4.1.2. Validación cruzada con k pliegues

En el método de validación cruzada con *k* pliegues (*k*-fold cross-validation) los datos se dividen aleatoriamente en *k* grupos. Un grupo se reserva para el conjunto de datos de prueba y con los otros *k*-1 restantes se construye un modelo y se usa para predecir el resultado de los datos del grupo reservado. Este proceso se repite *k* veces, dejando cada vez un grupo diferente para la prueba. Esto significa que se calculan *k* tasas de error independientes. Finalmente se construye un modelo con todos los datos y se obtienen sus tasas de error y precisión promediando las *k* tasas de error disponibles.

2.4.1.3. Bootstrapping

Otra técnica para estimar el error de un modelo cuando se disponen de pocos datos, es la conocida como bootstrapping. Ésta consiste en construir un primer modelo con todos los datos iniciales. Entonces, se crean numerosos conjuntos de datos, llamados bootstrap samples, haciendo un muestreo de los datos originales con reemplazo, es decir, se van seleccionando instancias del conjunto inicial, pudiendo seleccionar la misma instancia varias veces. Nótese que los conjuntos construidos de esta forma pueden contener datos repetidos. A continuación se construye un modelo con cada conjunto y se calcula su tasa de error sobre el conjunto de test (que son los datos sobrantes de cada muestreo). El error final estimado para el modelo construido con todos los datos se calcula promediando los errores obtenidos para cada muestra.

2.4.2. Medidas de evaluación de modelos

Dependiendo de la tarea de minería de datos existen diferentes medidas de evaluación de los modelos. Por ejemplo, en el contexto de la clasificación, lo normal es evaluar la calidad de los patrones encontrados con respecto a su precisión predictiva, la cual se calcula como el número de instancias del conjunto de prueba.

En el caso de que la tarea sea de reglas de asociación, se suele evaluar de forma separada cada una de las reglas con objeto de restringirnos a aquellas que pueden aplicarse a un mayor número de instancias y que tienen una precisión relativamente alta sobre estas instancias. Esto se hace en base a dos conceptos:

- Cobertura o soporte: número de instancias a las que la regla se aplica y predice correctamente.
- Confianza: proporción de instancias que la regla predice correctamente, es decir, la cobertura dividida por el número de instancias a las que se puede aplicar la regla.

Si la tarea es regresión, es decir, la salida del modelo es un valor numérico, la manera más habitual de evaluar un modelo es mediante el error cuadrático medio del valor predicho respecto al valor que se utiliza como validación. Esto promedia los errores y tiene más en cuenta aquellos errores que se desvían más del valor predicho (ponderación cuadrática).

Para la tarea de clustering, las medidas de evaluación suelen depender del método utilizado, aunque suelen ser función de la cohesión de cada grupo y de la separación entre grupos. La cohesión y separación ente grupos puede formalizar, por ejemplo, utilizando la distancia media al centro del grupo de los miembros de un grupo y la distancia media entre grupos respectivamente. El concepto de distancia y densidad son dos aspectos cruciales tanto en la construcción de modelos de agrupamiento como en su evaluación.

2.4.3. Interpretación y contextualización

Pese a todas las medidas vistas anteriormente, en muchos casos hay que evaluar también el contexto donde el modelo se va a utilizar. Por ejemplo, en el caso de la clasificación y las reglas de asociación, usar la precisión como medida de calidad tiene ciertas desventajas. En primer lugar, no tiene en cuenta el problema de tener distribuciones de clases no balanceadas. Este aspecto pone en evidencia que es necesario conocer mejor el tipo de errores y su costo asociado. En los problemas de clasificación se usa la matriz de confusión, la cual muestra el recuento de casos de las clases predichas y sus valores actuales.

La consideración de que todos los errores no son iguales puede incluso tenerse en cuenta en situaciones donde los costos de error suelen ser difíciles de estimar o incluso desconocidos para muchas aplicaciones. En estos casos, se usan estrategias alternativas como el análisis ROC (Receiver Operating Characteristic).

Como se ha mencionado anteriormente, la precisión de un modelo no garantiza que refleje el mundo real. Normalmente, esta situación se produce cuando al construir el modelo no hemos tenido en cuenta algunos parámetros que implícitamente influyen en él. En cualquier caso, se deberá contrastar el conocimiento que éste nos proporciona con el conocimiento previo que pudiéramos tener sobre el problema para detectar y en su caso resolver los posibles conflictos.

2.5. Fase de difusión, uso y monitorización

Una vez construido y validado el modelo puede usarse principalmente con dos finalidades: para que un analista recomiende acciones basándose en el modelo y en sus resultados, o bien para aplicar el modelo a diferentes conjunto de datos. También puede incorporarse a otras aplicaciones, como por ejemplo a un sistema de análisis de créditos bancarios, que asista al empleado bancario a la hora de evaluar a los solicitantes de los créditos, o incluso automáticamente, como los filtros de spam o la detección de compras con tarjeta de crédito fraudulentas.

Tanto en el caso de una aplicación manual o automática del modelo, es necesario su difusión, es decir que se distribuya y se comunique a los posibles usuarios, ya sea por cauces habituales dentro de la organización, reuniones, intranet, etc. El nuevo conocimiento extraído debe integrar el know-how de la organización.

También es importante medir lo bien que el modelo evoluciona. Aun cuando el modelo funcione bien es necesario comprobar continuamente las prestaciones del mismo. Esto se debe principalmente a que los patrones pueden cambiar.

Las técnicas de minería de datos producen modelos que son capaces de explicar los datos de entrenamiento, y al mismo tiempo aprender patrones generales desde los datos, de manera que pueden predecir futuros casos. Sin embargo, muchas técnicas producen unos modelos cuya complejidad es tan alta o su representación tan críptica, que se interpretan como cajas negras, dado que es prácticamente imposible conocer el comportamiento interno. Ejemplos de estas técnicas son las redes neuronales o las máquinas de vectores de soporte.

La comprensibilidad es, en muchos contextos, condición necesaria. Por ejemplo, podemos aprender un modelo que sea capaz de diagnosticar alguna enfermedad en un paciente a partir de los datos del resultado de su análisis de sangre, utilizando para ello las experiencias de otros pacientes. Sin embargo, en los diagnósticos es siempre el médico quien tiene la última palabra. Para que el sistema sea útil para el médico, es necesario que éste pueda comprender las razones que utiliza el sistema para determinar qué patología tiene un paciente.

En realidad, la comprensibilidad es un factor subjetivo, ya que depende en gran modo de la experiencia y conocimiento de los usuarios de los modelos. Los sistemas de reglas son considerados como una de las representaciones que permitan comprender más fácilmente el comportamiento de un modelo. Además, tienen la ventaja de que las reglas se expresan utilizando los propios atributos del problema directamente. Aun así, es necesario considerar que un modelo formado por un número elevado de reglas, y cada una de ellas con multitud de atributos, puede ser más complicado de entender que una pequeña red neuronal. Por lo tanto, también es necesario tener en cuenta el tamaño de los modelos cuando la comprensibilidad de estos últimos es un factor determinante.

El problema de la pérdida de comprensibilidad afecta también a los métodos multi-clasificadores o a los métodos híbridos. Estos métodos combinan o fusionan diferentes técnicas de aprendizaje con el fin de incrementar la precisión de los modelos aprendidos.

Por lo tanto, la comprensibilidad no puede ser obviada a la hora de elegir los métodos de minería de datos. En el caso de que este factor sea importante, se deben seleccionar métodos de aprendizaje de reglas (reglas de decisión, IPL), o equivalentemente, métodos que aprendan modelos que puedan ser expresados como reglas (árboles de decisión, métodos difusos). También los modelos estadísticos simples (por ejemplo la regresión lineal) suelen ser bastantes comprensibles. En todo caso, es también importante incluir técnicas que permitan reducir la complejidad de los modelos como por ejemplo podar reglas o filtrar atributos no relevantes.

3. Arboles de decisión

De todos los métodos de aprendizaje, los sistemas de aprendizaje basados en árboles de decisión son quizás el método más fácil de utilizar y de entender. Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas. Los árboles de decisión son especialmente apropiados para expresar procedimientos médicos, legales, comerciales, estratégicos, matemáticos, lógicos, etc.

Una de las grandes ventajas de los árboles de decisión es que, en su forma más general, las opciones posibles a partir de una determinada condición son excluyentes. Esto permite analizar una situación y, siguiendo el árbol de decisión apropiadamente, llegar a una sola acción o decisión a tomar.

La tarea de aprendizaje para la cual los árboles de decisión se adecuan mejor es la clasificación. De hecho, clasificar es determinar de, entre varias clases, a qué clase pertenece un objeto. La estructura de condición y ramificación de un árbol de decisión es idónea para este problema. La característica más importante del problema de la clasificación es que se asume que las clases son disjuntas, es decir, una instancia es de la clase A o de la clase B, pero no puede ser al mismo tiempo de las clases A y B. En este sentido, la clasificación se diferencia de la categorización, donde se permite más de una clase, etiqueta o categoría para cada instancia.

Debido al hecho de que la clasificación trata con clases o etiquetas disjuntas, un árbol de decisión conducirá un ejemplo hasta una y sólo una hoja, asignando, por tanto, una única clase de ejemplo. Para ello, las particiones existentes en el árbol deben ser también disjuntas. Es decir, cada instancia cumple o no cumple una condición.

Esta propiedad da lugar al esquema básico de los primeros algoritmos de aprendizaje de árboles de decisión; el espacio de instancias se va partiendo de arriba hacia abajo, utilizando cada vez una partición, es decir, un conjunto de condiciones excluyentes y exhaustivas. Estos algoritmos se llaman algoritmos de partición o algoritmos de "divide y vencerás". Otra característica importante de los primeros algoritmos de aprendizaje de árboles de decisión es que una vez elegida la partición, esta no se puede cambiar, aunque más adelante se pensara que había sido una mala elección. Por tanto, uno de los aspectos más importantes en los sistemas de aprendizaje de árboles de decisión es el denominado criterio de partición, ya que una mala elección de la partición (especialmente en las partes superiores del árbol) generará un árbol malo.

Simplemente, el algoritmo va construyendo el árbol (desde el árbol que sólo contiene la raíz) añadiendo particiones y los hijos resultantes de cada partición. Lógicamente, en cada partición, los ejemplos se van dividiendo entre los hijos. Finalmente, se llega a la situación en la que todos los ejemplos que caen en los nodos inferiores son de la misma clase y esa rama ya no sigue creciendo. La única condición que hay que exigir es que las particiones al menos separen ejemplos en distintos hijos, con lo que la cardinalidad de los nodos irá disminuyendo a medida que se desciende en el árbol.

Como se acaba de comentar, los dos puntos más importantes para que el algoritmo anterior funcione bien son los siguientes:

- Particiones a considerar.
- Criterio de selección de particiones.

Esto es lo que diferencia fundamentalmente los distintos algoritmos de "partición" existentes hasta la fecha, como CART (Breiman, y otros, 1984), ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) entre otros.

3.1. Particiones

Las particiones son, como se ha mencionado, un conjunto de condiciones exhaustivas y excluyentes. Lógicamente, cuantos más tipos de condiciones se permitan, habrá más posibilidades de encontrar los patrones que hay detrás de los datos. Por ejemplo, un algoritmo que permita incluir la partición ($x_i \leq a \cdot x_j^2$, $x_i > a \cdot x_j^2$) donde x_i y x_j son atributos del problema y a es una constante, va a poder encontrar patrones cuadráticos, mientras otro algoritmo que no permita dicha partición no va a poder encontrarlos. Cuantas más particiones se permitan más expresivos podrán ser los árboles de decisión generados y, probablemente, más precisos. No obstante, cuantas más particiones se elijan, la complejidad del algoritmo será mayor. Por tanto, la clave en un buen algoritmo de aprendizaje de árboles de decisión es encontrar un buen compromiso entre expresividad y eficiencia.

Debido a esto, la mayoría de algoritmos de aprendizaje de árboles de decisión sólo permiten un juego muy limitado de particiones. Por ejemplo, el algoritmo más popular, denominado C4.5 (Quinlan, 1993), y desarrollado por Ross Quinlan en la década de 1980 y principios de los 1990 a partir de un algoritmo anterior denominado ID3 (Quinlan, 1986), contiene un solo tipo de partición para los atributos nominales y un solo tipo de partición para los atributos numéricos.

La expresividad resultante de las particiones se conoce como expresividad proporcional cuadricular. El término proporcional se refiere a que son particiones que sólo afectan a un atributo de un ejemplo a la vez, es decir, ni relacionan dos atributos del mismo ejemplo, ni dos atributos de distintos ejemplos. El término cuadricular hace referencia al tipo de particiones que realizan, especialmente cuando se refiere a los atributos numéricos (Figura 1-2).

Aunque las particiones de atributos nominales y atributos numéricos resultan bastantes sencillas, permiten obtener árboles de decisión bastante precisos y muy comprensibles, ya que las condiciones ($x_i = v_1$, $x_i = v_2$...,

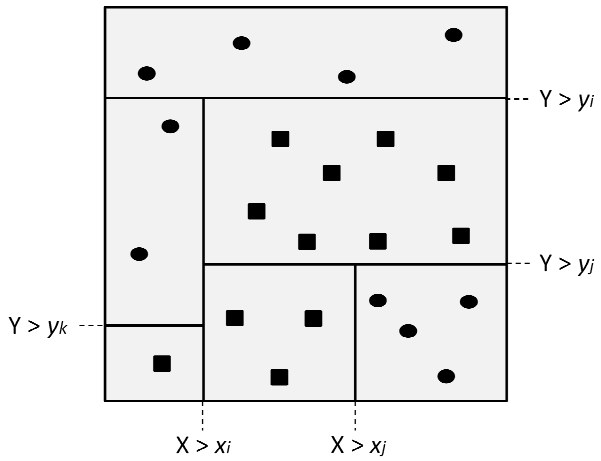

Figura 1-2. Clasificación cuadricular realizada por un árbol de decisión con dos atributos numéricos.

$x_i = v_k$) y ($x_i \leq a$, $x_i >$ a) se pueden ajustar a muchos patrones y son fácilmente interpretables por los seres humanos. En (Shali, y otros, 2007) se presenta un algoritmo el cual permite utilizar más de un atributo para realizar una partición en una estructura denominada árboles de decisión indirectos, para lo cual se utiliza programación genética.

3.2. Criterio de selección de particiones

Incluso con sólo los dos tipos de particiones sencillas, el número de particiones posibles en cada caso puede dispararse (si existen n atributos y m valores posibles para cada atributo, el número de particiones posibles es de $n \cdot m$). Como se mencionó anteriormente, los algoritmos clásicos de aprendizaje de decisión son voraces, en el sentido de que una vez elegida la partición se continúa hacia abajo la construcción del árbol y no vuelven a plantearse las particiones ya construidas. Estos dos aspectos tienen como consecuencia que se busque un criterio que permita realizar una buena elección de la partición que parece más prometedora y que esto se haga sin demasiado esfuerzo computacional. Esto obliga a que calcular la optimalidad de cada partición no sea muy costoso.

La mayoría de criterios se basan por tanto, en obtener medidas derivadas de las frecuencias relativas de las clases en cada uno de los hijos de la partición respecto a las frecuencias relativas de las clases en el padre. Por ejemplo, si un nodo posee un 50% de ejemplos de clase A y un 50% de ejemplos de clase B, una partición que de como resultado dos nodos n_1 y n_2, donde todos los ejemplos en n_1 sean clases A y todos los ejemplos en n_2 sean de la clase B, será una buena partición, porque los nodos resultantes son más puros que el padre. Por el contrario, si ambos nodos n_1 y n_2 siguen teniendo proporciones cercanas al 50% no se habrá discriminado nada y por lo tanto no se avanzará hacia un árbol que clasifique correctamente la evidencia.

Basándose en la idea de buscar particiones que discriminen o que consigan nodos más puros, se han presentado en las últimas décadas numerosos criterios de partición, tales como el criterio del error esperado, el criterio Gini (Breiman, y otros, 1984), los criterios Gain, Gain Ratio y la modificación del C4.5 (Quinlan, 1993) y el DKM (Kearns, y otros, 1995). Estos criterios de partición buscan la partición s con el menor $I(s)$, definido de la siguiente forma:

$$I(s) = \sum_{j=1..n} p_j f(p_j^1, p_j^2, \ldots, p_j^c)$$

donde n es el número de los nodos hijos de la partición (número de condiciones de la partición), p_j es la probabilidad de "caer" en el nodo j, p_j^1 es la proporción de elementos de la clase 1 en el nodo j, p_j^2 es la propoción de los elementos de la clase 2 en el nodo j, y así para las c clases. Bajo esta fórmula general, cada criterio de partición implementa una función f distinta. Estas funciones f se denominan funciones de impureza y, por lo tanto, la función $I(s)$ calcula la media ponderada (dependiendo de la cardinalidad de cada hijo) de la impureza de los hijos de una partición.

Existen muchas variantes de estos criterios (por ejemplo el GainRatio y el criterio usado en el C4.5 son variantes de la entropía), la ortogonalidad de GID3 (Fayyad, 1994), y muchos otros que no se ajustan a la fórmula anterior (por ejemplo criterios basados en el principio MDL (Minimum Description Length) (Ferri Ramírez, y otros, 2001) o criterios basados en el AUC (Area under the ROC Curve) (Ferri Ramirez, y otros, 2002). Según los experimentos realizados durante los últimos años, parece ser que tanto el criterio usado por el C4.5 (GainRatio modificado) como el criterio DKM se comportan ligeramente mejor que el GINI y bastante mejor que el Error Esperado.

3.3. Poda y reestructuración

Los algoritmos de aprendizaje de árboles de decisión obtienen un modelo que es completo y consistente con respecto a la evidencia. Es decir, el modelo cubre todos los ejemplos vistos y los cubre todos de manera correcta. Esto puede parecer óptimo a primera vista, pero se vuelve demasiado ingenuo en la realidad. En primer lugar, ajustarse demasiado a la evidencia suele tener como consecuencia que el modelo se comporte mal para nuevos ejemplos, ya que, en la mayoría de los casos, el modelo es solamente una aproximación del concepto objetivo del aprendizaje. Por tanto, intentar aproximar demasiado hace que el modelo sea demasiado específico, poco general y, por tanto, malo con otros datos no vistos. En segundo lugar, cuando la evidencia puede contener ruido (errores en los atributos o incluso en las clases), ya que el modelo intentará ajustarse a los errores y esto perjudicará el comportamiento global del modelo aprendido (sobreajuste (overfitting)).

La manera más frecuente de limitar este problema es modificar los algoritmos de aprendizaje de tal manera que obtengan modelos más generales. En el contexto de los árboles de decisión generalizar significa eliminar condiciones de las ramas del árbol o de algunas reglas. Dicho procedimiento se puede ver gráficamente como un proceso de "poda". Los nodos que están por debajo del límite de poda se eliminan, ya que se consideran demasiado específicos.

Para la poda de árboles se han propuesto varios trabajos que intentan introducir poda basados en decisiones más óptimas. En (Garcia-Almanza, 2006) se presenta una técnica que realiza una poda eliminando reglas que causan clasificaciones errónea utilizando programación genética.

3.4. Extracción de reglas

Los árboles de decisión se pueden expresar como un conjunto de reglas. Este conjunto de reglas se derivan de particiones, en las cuales para cualquier condición siempre aparece además la o las condiciones complementarias. Existen, en cambio, muchos conjuntos de reglas que no cumplen estas condiciones y, sin embargo, son capaces de clasificar la evidencia de una manera conveniente.

Los métodos que generan estos conjuntos van añadiendo reglas, una de detrás de otra, mientras vayan cubriendo ejemplos de una manera consistente. A diferencia de los árboles de decisión, no se sigue con las condiciones complementarias a la utilizada en la regla anterior (exclusión y exhaustividad), sino que se descartan los ejemplos ya cubiertos por las reglas ya obtenidas y con los ejemplos que quedan se empieza de nuevo. Esto hace que puedan aparecer nuevas condiciones que solapen (o no) con las anteriores. Esta manera de actuar es la que utilizan los métodos de cobertura.

El criterio de selección de las condiciones puede basarse en la pureza (la que elimine más contraejemplos) o en medidas derivadas de la información (como el Gain visto para árboles de decisión) o medidas que ponderen la precisión y el alcance (precision and recall), por ejemplo la medida-f (van Rijsbergen, 1979).

Las ventajas e inconvenientes de los algoritmos basados en partición y los algoritmos basados en cobertura son:

- Partición: suelen ser más eficientes debido a que cada partición en realidad genera dos o más reglas. La poda es más sencilla.

- Cobertura: permite hacer coberturas no exhaustivas, que es interesante cuando un atributo tiene muchos valores y sólo algunos son significativos. Es menos voraz y las últimas ramas se ajustan más a los ejemplos. Esto también es una desventaja, porque las últimas ramas suelen hacer sobreajuste (overfitting), es decir, intentan capturar aparentes regularidades del conjunto de entrenamiento que resultan ser ruido y el modelo es demasiado específico.

Se han presentado muchas técnicas que tienen como objetivo generar reglas por cobertura. En (Hu, y otros, 1996) utilizan técnicas de aprendizaje de máquina y la teoría de los conjuntos aproximados (Rough Sets Theory (Pawlak, 1991) (Pawlak, 2002)) para la reducción de atributos y de esa manera eliminar aquellos que resulten innecesarios de las reglas. En la misma línea se han desarrollado técnicas que combinan diferentes modelos de multi-clasificadores (Hu, 2001) o técnicas híbridas entre lógica difusa y Rough Sets Theory (Zhao, y otros, 2010).

Los mapas auto-organizativos de Kohonen también han sido utilizados para generar modelos de datos a partir de los cuales se extraen un conjunto de reglas (Hasperué, y otros, 2006) (Malone, y otros, 2006) (Darrah, y otros, 2004), además de los mapas auto-organizativos de Kohonen también se han presentado propuestas de extracción de reglas sobre otras arquitecturas de las redes neuronales artificales (Ma, y otros, 2009) (Andrews, y otros, 1995), incluso hay técnicas que utilizan redes neuronales junto con técnicas de optimización (Hung, y otros, 2010). Las máquinas de soporte de vectores (SVM - Support Vector Machine) también han sido utilizadas para la extracción de reglas (Barakat, y otros, 2007).

Muchas veces las reglas extraídas por las distintas técnicas existentes resultan en un número muy elevado transformando al conocimiento adquirido en un conocimiento poco útil, dada la enorme cantidad de reglas. En esta dirección existen varios trabajos que realizan una poda en el conjunto de reglas, dejando solamente aquellas que resulten más relevantes. En (Marinica, y otros, 2010) se presenta una técnica interactiva para poda de reglas basada en ontologías.

4. Algoritmos evolutivos

El término Computación Flexible (Soft Computing) agrupa a un conjunto de metodologías cuya característica principal es la tolerancia a imprecisión e incertidumbre, lo que le confiere una capacidad de adaptación que permite solucionar problemas en entornos cambiantes de forma robusta y con bajo coste. De hecho, el principio que subyace en la computación flexible es la hibridación de técnicas para el desarrollo de métodos computacionales que obtengan una solución aceptable a bajo coste mediante la búsqueda de una solución aproximada a un problema formulado de forma precisa o imprecisa. Dentro de la computación flexible se incluyen metodologías como la lógica difusa, las redes neuronales y la computación evolutiva (Tettamanzi, y otros, 2001).

En el proceso de extracción de conocimiento en bases de datos y, en concreto, en el proceso de minería de datos existen distintas tareas o problemas que se pueden enfocar y resolver como problemas de optimización y búsqueda. Los algoritmos evolutivos imitan los principios de la evolución natural para formar procedimientos de búsqueda y optimización global y son aplicables para el desarrollo de algoritmos de minería de datos propiamente dichos, como algoritmos de pre o pos-procesamiento o como herramientas para la optimización de los parámetros de otros algoritmos.

Los algoritmos evolutivos tienen un carácter de búsqueda global que hace que sean especialmente adecuados para resolver problemas presentes en las distintas etapas del proceso de descubrimiento de conocimiento (Freitas, 2002). Por ejemplo, en procesos de extracción de reglas, los algoritmos evolutivos tratan de forma adecuada las interacciones entre atributos porque evalúan una regla como un todo mediante la función de adaptación en lugar de evaluar el impacto de añadir y/o eliminar una condición de una regla, como ocurre en los procesos de búsqueda local incluidos en la mayoría de los algoritmos de inducción de reglas y árboles de decisión. Entre las distintas clases de algoritmos evolutivos, los algoritmos genéticos y la programación genética

son los más utilizados para el descubrimiento de reglas. Estas dos clases de algoritmos difieren fundamentalmente en la representación de los individuos. Como se ha comentado, para los algoritmos genéticos, los individuos se representan como una cadena lineal de condiciones, y en el caso de reglas cada condición suele ser una pareja atributo-valor, mientras que en programación genética un individuo suele representarse mediante un árbol, y en este caso particular los nodos hoja o terminales son condiciones de reglas y/o valores de atributos, y los nodos internos representan las funciones.

Además, en otras etapas del proceso de descubrimiento del conocimiento es necesario seleccionar variables, ejemplos u optimizar parámetros. Para cualquiera de estas tareas se han utilizado distintas propuestas de algoritmos evolutivos.

Cualquier propuesta de algoritmo genético de extracción de reglas, cuyo objetivo sea obtener reglas, debe determinar el esquema de representación utilizado para codificar cada una de las soluciones, los operadores genéticos y la función de adaptación.

Si un algoritmo genético tiene como objetivo descubrir una regla entonces deberá codificar en un cromosoma el antecedente de la regla, es decir, sigue el esquema de codificación "Cromosoma=Regla". El consecuente no lo codifica, sino que cada ejecución del algoritmo genético se realiza para cada uno de los valores del atributo objetivo.

- Representación. Cada cromosoma codifica sólo el antecedente de la regla ya que el consecuente es fijo durante cada ejecución del algoritmo genético. El antecedente de la regla se representa en un cromosoma de longitud fija con un esquema de codificación entera (la posición *i*-ésima indica el valor que toma la variable *i*-ésima). Para las variables numéricas, en una etapa de pre-procesamiento, se determina un conjunto de intervalos, y la base de ejemplos se discretiza asignándole a cada valor numérico el número de intervalo al que pertenece. Existen múltiples propuestas para la discretización, la más inmediata es la división uniforme (equidistante) del dominio de cada variable en un número de intervalos. El esquema de codificación incluye siempre un valor adicional para cada gen para indicar que la variable correspondiente no participa en la regla.

- Función de adaptación. La función de adaptación es la media aritmética de la completitud (soporte) y la confianza de la regla representada en el cromosoma. La confianza determina la precisión de la regla, ya que refleja el grado con el que los ejemplos pertenecientes a la zona de espacio delimitado por el antecedente verifican la información indicada en el consecuente de la regla. Esta medida se calcula como el número de ejemplos de la clase que pertenecen a la zona determinada por el antecedente entre el número de todos los ejemplos (independientemente de su clase) que pertenecen a la misma zona. La completitud mide el grado de cobertura de la regla con respecto a los ejemplos de la clase, y se calcula como el cociente entre el número de ejemplos de la clase que pertenecen a la zona determinada por el antecedente y el número total de ejemplos de la clase.

- Operadores genéticos. El algoritmo genético utiliza un modelo de reproducción de estado estacionario. En éste, todos los individuos de población se mantienen en el proceso evolutivo salvo dos, los dos peores que serán sustituidos por los individuos resultantes de cruzar y mutar los dos mejores individuos de la población. El operador de cruce que se ha utilizado es el cruce en dos puntos que funciona de la siguiente forma: para cada pareja de cromosomas a cruzar se seleccionan aleatoriamente dos puntos en la longitud del cromosoma y se intercambia el material genético entre estos dos puntos dando lugar a dos descendientes. El operador de mutación que se utiliza es la mutación uniforme que selecciona aleatoriamente una posición del cromosoma (un valor de una variable del antecedente de la regla) y cambia u valor por uno obtenido aleatoriamente de entre los valores posibles de la variable.

El carácter estocástico de los algoritmos genéticos determina que para mostrar su buen funcionamiento deban realizarse varias ejecuciones, y evaluar el mejor, el peor y la desviación típica de los resultados, entre otras medidas.

El algoritmo genético permite obtener una regla para cada clase con un valor alto de confianza y un nivel de completitud adecuado. Con esta primera aproximación al problema se consigue un algoritmo genético sencillo para obtener reglas adecuadas. Los resultados se podrían mejorar con un enfoque que permita obtener más de una regla, un esquema de codificación que permita representar reglas más flexibles, un mejor método de discretización, etc.

En varios trabajos es posible ver el uso de algoritmos genéticos para diferentes tareas de la minería de datos (García, y otros, 2009). Incluso, otras técnicas de optimización como ACO y PSO han sido utilizadas para resolver problemas de minería de datos (Holden, y otros, 2008) (Martens, y otros, 2007). En la tarea de extraer un conjunto de reglas como conocimiento los algoritmos genéticos también han sido aplicados con frecuencia (Konig, y otros, 2007).

5. Minado de datos incremental

Los modelos obtenidos en la fase de minería de datos, convenientemente validados e integrados en el uso de la organización no duran siempre. Las condiciones del entorno del problema pueden alterarse, haciendo que un modelo que en un principio era muy útil, deje de ser válido para su aplicación debido a que no se ha adaptado a la nueva situación (Al-Hegami, 2007).

Por otra parte, en muchas ocasiones, las bases de datos desde donde extraer información con técnicas de minería de datos están formadas por un número tan alto de registros o de atributos, que hace inviable su tratamiento directamente por parte de los algoritmos de aprendizaje, ya que desborda la capacidad de los sistemas.

Si bien es posible aplicar una reducción de la dimensionalidad de los datos, otra aproximación diferente es la utilización de algoritmos incrementales. Un algoritmo incremental es capaz de aprender de modelos utilizando tan sólo una parte de la información disponible, y posteriormente, utilizar el resto para actualizar o revisar los modelos aprendidos. Si el algoritmo utiliza los ejemplos gradualmente, se denomina incrementalidad vertical. En el caso de que el algoritmo emplee todos los ejemplos, pero la incrementalidad se realice con respecto al número de atributos, se llama incrementalidad horizontal.

En realidad, un algoritmo incremental es una aproximación más realista que un algoritmo no incremental, ya que en muchos casos es imposible tener ejemplos de todos los casos en el momento de realizar el aprendizaje. En esta situación, es mejor aprender un modelo inicial para posteriormente aplicarlo para predecir nuevos casos, y cuando se conozca la clase real de los nuevos casos (siempre que sea posible), utilizar esta información para realimentar el modelo. Esta realimentación puede suponer la actualización del modelo mediante la modificación o supresión de parte del modelo, o bien la inclusión de nuevas partes.

Se debe tener en cuenta que, en muchos problemas, el comportamiento no es estático, sino que va evolucionando constantemente, por lo que utilizar un algoritmo de aprendizaje incremental permite que el modelo vaya evolucionando al mismo tiempo que el problema.

Existen versiones incrementales para la mayoría de los algoritmos de aprendizaje, en (Alippi, y otros, 2011) utilizan reglas de intersección de intervalos de confianza junto a estimadores de aproximaciones polinomiales locales, lo que ha dado lugar a una técnica que se comporta de una manera aceptable tanto en condiciones normales de trabajo como ante cambios abruptos, lo que la transforma en una buena opción para sistemas de monitoreo y detección de fallas.

Cabe destacar que, se ha estudiado más esta extensión en algoritmos que no son particularmente escalables para el tratamiento de grandes volúmenes de datos, como por ejemplo los árboles de decisión (Schlimmer, y otros, 1986) (Chao, y otros, 2009) (Utgoff, 1988) (Utgoff, 1989) (Utgoff, 1994) (Utgoff, y otros, 1996) (Piao, y otros, 2010) (Qingyun, 2011) (Shaorong, y otros, 2010).

5.1. Adaptabilidad del modelo

Otro posible escenario para la minería de datos incrementales es aquellos donde por la propia naturaleza del problema aparecen nuevos datos, desaparecen otros o los que formaban parte de la base de datos original sufren modificaciones. Ejemplos de estos tipos de problemas son: análisis sobre el rendimiento de los alumnos de primer año de una universidad, tareas de marketing que dependen de los gustos actuales de un grupo de la sociedad, aceptación de créditos bancarios que dependen del sistema económico del gobierno de turno, etc.

Si bien es posible, para cualquiera de estos problemas, generar en un momento dado un modelo con los datos que se fueron recolectando y almacenando en un largo período de tiempo, es de esperar también que con el paso del tiempo sean recolectados nuevos datos, eliminados aquellos que son considerados superfluos o adaptados los datos actuales en nuevas categorías o clases.

En este tipo de problemas es inviable pensar en generar un nuevo modelo con la base de datos actualizada, ya que este procedimiento no solo puede generar un consumo de tiempo considerable sino que no es posible garantizar que el conocimiento que se poseía con el modelo anterior aparezca en el nuevo modelo generado. Por estos motivos aparecen técnicas que resuelven los escenarios planteados por la minería de datos incremental. En (Kasten, y otros, 2007) se presenta un clasificador que es capaz de aprender de una manera dinámica y on-line y que puede tomar decisiones de manera automática.

En (Ahmed, y otros, 2009) se presentan tres variantes de un algoritmo que construye árboles de decisión para el minado de los datos más comunes (frequent itemsets) el cual permite de manera dinámica e interactiva la re-estructuración ante los cambios que sufren los datos. En (Vachkov, 2010) se presenta una técnica basado en análisis de similitudes difusas asistidas por humanos para la clasificación incremental de imágenes. En (Shaorong, y otros, 2010) se presenta una técnica basada en árboles para la detección de malware, como la aparición de nuevo malware ocurre periódicamente, esta misma técnica es capaz de adaptarse a los cambios sin necesidad de re-entrenarse.

En (Kasten, y otros, 2004) se presenta una técnica de minado incremental la cual trabaja con una estructura en forma de árbol, donde cada nodo es una esfera que mide ciertos estímulos del ambiente de trabajo y que es utilizada para monitorear una red wireless y supervisar la sobrecarga del tráfico de paquetes. Esta técnica además presenta una novedosa forma de trabajar ya que posee un sistema de memoria que le permite recordar respuestas a estímulos anteriores para poder repetirlos en el futuro.

6. Toma de decisiones

El área de ayuda a la toma de decisiones (Mladenić, 2003) constituye un área multidisciplinar cuyo objetivo es la introducción de métodos y/o herramientas que ayuden a los humanos en la toma de decisiones clave. En esta sección analizamos brevemente cómo podemos utilizar técnicas de minería de datos, más concretamente los modelos construidos mediante técnicas de minería de datos, para ayudar en el proceso de seleccionar la mejor decisión entre varias alternativas.

En el campo de la ayuda de toma de decisiones, la fase de toma de decisiones usualmente se refiere al proceso completo de realizar la selección. Este proceso incluye: conocer el problema, recoger información sobre el problema, identificar alternativas, anticipar lógicos y coherentes basándose en la información disponible, etc. Es posible, entonces, definir el área de la ayuda de decisiones como el área que concierne a la toma de decisiones por parte de seres humanos, y especialmente, el estudio de técnicas que asistan a las personas a mejorar las decisiones tomadas. Entre estas técnicas podemos ubicar a la minería de datos.

La minería de datos tiene como propósito el descubrimiento de patrones novedosos y útiles desde un conjunto de datos. Estos modelos se pueden utilizar como herramientas de ayuda en la toma de decisiones. Por ejemplo, el aprendizaje de árboles de decisión aprende un modelo que, tal y como su nombre lo indica, es capaz de recomendar una decisión a partir de ciertos datos.

En los últimos años se ha popularizado el uso de la minería de datos en los sistemas de ayuda a la toma de decisiones. Normalmente, el típico escenario para aplicar técnicas de minería de datos para ayudar en la toma de decisiones se sitúa en un contexto empresarial.

Esta tendencia de toma de decisiones nace debido a que las técnicas tradicionales de minería de datos tienen un enfoque centrado en el desarrollo y uso de algoritmos específicos y por lo general el proceso de minería de datos se detiene una vez alcanzada la identificación de los patrones. En consecuencia a este enfoque se destacan tres grandes problemas:

- Se han desarrollado muchos algoritmos los cuales se centran en un problema en particular y difícilmente son adaptables a otros problemas.
- Estos algoritmos por lo general extraen una gran cantidad de patrones aunque solo una mínima proporción de ellos son realmente de interés para la solución del problema base.
- A los usuarios finales por lo general no le resulta sencillo entender el uso de estas técnicas y por lo tanto no pueden explotar las capacidades de las técnicas de la minería de datos.

Por estos motivos se está invirtiendo mucho esfuerzo en promover la "accionabilidad" de la extracción de conocimiento para la toma de decisiones. Para este fin aparece la minería de datos dirigida al dominio (domain-driven data mining (D^3M)) la cual promueve un paso de la "extracción de conocimiento centrada en los datos" a la "entrega de conocimiento accionable dirigida al dominio". En D^3M se incorpora la inteligencia ubicua en los modelos y tareas del proceso de minería de datos (Zhang, y otros, 2010) (Cao, 2010).

7. Hiper-rectángulos

Cada una de las técnicas descriptas en este capítulo presentan un modo distinto de representar los datos y la información extraída del modelo de datos. Por ejemplo, los árboles de decisión utilizan, como su nombre lo índica, una estructura de datos en forma de árbol donde cada nodo representa a un subconjunto de los datos utilizados en el entrenamiento. Las redes neuronales almacenan la información en los pesos de las conexiones entre distintas neuronas. Las técnicas de optimización, como los algoritmos evolutivos o PSO, utilizan la representación del mejor individuo o de la mejor partícula para representar el conocimiento.

Los hiper-rectángulos es otro modo de representar los datos en un modelo. Un hiper-rectángulo en un espacio de D dimensiones puede ser visto como un rectángulo en el plano o un cubo en el espacio de tres dimensiones. Los hiper-rectángulos como descriptores de los datos se utilizan con el criterio de que un hiper-rectángulo es descriptor de todos los datos que están contenidos dentro de él. Además un hiperrectángulo puede ser representado por dos vectores D dimensionales, los cuales representan a los valores mínimos y máximos para cada atributo.

Los hiper-rectángulos ofrecen dos grandes ventajas para ser utilizados como representación para los datos:

- Al estar descriptos por solo dos vectores, las operaciones entre ellos resultan sencillas (división, reducción o aumento de volumen, detección de superposiciones, etc.)
- Al ser representante de los datos que están incluidos en el propio hiper-rectángulo, los límites de este pueden ser utilizados como descriptores del conjunto de datos representado y por lo tanto ofrece una buena oportunidad para extraer conocimiento de una manera directa.

El uso de los hiper-rectángulos para tareas de clasificación ha dado lugar a la técnica presentada en esta tesis y se verá con más detalle en el siguiente capítulo.

7.1. El uso de los hiper-rectángulos en minería de datos

Desde finales de los 80's se utilizan a los hiper-rectángulos para tareas de clasificación (Thornton, 1987). En los años siguientes aparecieron trabajos que, basados en la teoría de los k vecinos más cercanos (Hart, 1968)

(Cover, y otros, 1967), introducen diferentes técnicas con una filosofía similar, la de los *k* hiper-rectángulos más cercanos (Salzberg, 1991) (Wettschereck, y otros, 1995). Estos trabajos, basados en la teoría de la generalización, forman grupos de datos similares a partir de la cercanía de los datos analizados. Como ocurre en todas las técnicas basadas en distancias, la elección de una métrica adecuada para medir similitudes es crucial para lograr buenos resultados.

Debido a su fácil implementación y manejo los hiper-rectángulos han sido utilizados en diversas tareas y en diferentes áreas de aplicación. En (Bouillant, y otros, 2002) utilizan los hiper-rectángulos para construir un modelo de monitoreo de imperfecciones industriales. En (Xiang, y otros, 2011) se presenta una técnica de minería del ítem más frecuente para el resumen de bases de datos transaccionales. En (Hasperué, y otros, 2010) se utilizan los hiper-rectángulos para encontrar el número y forma de clusters óptimos. En (Wei, y otros, 2010) utilizan una combinación de clasificadores basados en hiper-rectángulos aproximados (Rough Hypercuboid) para la clasificación de cáncer.

El uso de hiper-rectángulos también ha sido fusionado con otras técnicas para resolver diferentes problemas. En (García, y otros, 2009) se ha presentado una técnica que evoluciona un conjunto de posibles hiper-rectángulos para realizar tareas de clasificación. También se han propuesto varias técnicas que construyen un árbol de hiper-rectángulos, donde cada nodo representa un hiper-rectángulo y sus correspondientes hijos representan hiper-rectángulos que están contenidos por el hiper-rectángulo del nodo padre (Beckmann, y otros, 1990) (Katayama, y otros, 1997). Algunas variantes incluso, arman el árbol en un sentido bottom-up (Chen, 2006).

Dado que los hiper-rectángulos son representaciones de datos "abstractas" en un espacio *d*-dimensional numérico, es posible representar cualquier tipo de información en dicho espacio. Por ejemplo, en (Lee, y otros, 2002) presentan una técnica de clustering de video clips donde los datos analizados son vectores *d*-dimensionales, los cuales representan un conjunto de características entre cuadros consecutivos dentro de la secuencia de video.

Capítulo 2

Clasificación utilizando hiper-rectángulos. Armado del modelo de datos y obtención de reglas de clasificación

Divide las dificultades que examinas en tantas partes como sea posible para su mejor solución.
(René Descartes)

En este capítulo se hace una introducción a los hiper-rectángulos como cuerpos geométricos en el espacio euclídeo de *D* dimensiones, se presentan distintos tipos de hiper-rectángulos y cuáles de ellos son de interés en este trabajo. Se explica cómo se relacionan los hiper-rectángulos con los datos presentes en la base de datos a analizar y como son utilizados para ayudar a extraer conocimiento en forma de reglas de clasificación.

Dos o más hiper-rectángulos pueden presentar superposiciones en el espacio euclídeo, estas superposiciones no son deseables en el modelo de datos a formar y se explica cómo tratar estas superposiciones para eliminarlas o reducirlas hasta alcanzar un resultado que satisfaga las necesidades del usuario del modelo de datos. Para esta tarea de eliminación de superposiciones se presenta, como aporte original, un conjunto de índices de superposición los cuales son utilizados para decidir de qué manera se elimina o minimiza una superposición. Cada índice mide una característica distinta de una superposición, se muestra en detalle la función de cada uno y cómo es posible introducir en la técnica presentada en esta tesis nuevos índices y personalizar el método de clasificación con un subconjunto de índices específico.

Se detalla la técnica de clasificación propuesta y el funcionamiento e implementación del algoritmo correspondiente. Al finalizar el armado del modelo de datos se explica como del conjunto de hiper-rectángulos formados se extraen las reglas de clasificación, las cuales conforman el conocimiento extraído que le son de utilidad al usuario del modelo de datos. Una de las tareas que puede realizar un modelo armado es el de predecir las clases de los nuevos datos que se obtengan, así se detalla como el modelo hace la predicción de los datos que se le presentan.

Finalmente, se discute como un experto en el dominio del problema puede supervisar el armado del modelo de datos durante la ejecución de la técnica de clasificación.

1. Hiper-rectángulos

Se conocen como hiper-rectángulos a los politopos[1] *D*-dimensionales donde todas sus caras tienen forma de rectángulo. Así, si un problema dado se presenta en dos dimensiones se trabaja simplemente con rectángulos y si es en tres dimensiones se hace con los cuerpos en el espacio conocidos como ortoedros o cuboides. En éste y en los restantes capítulos se hace uso del término hiper-rectángulo de manera genérica sin importar la dimensión del espacio, incluso si es en dos o tres dimensiones.

En el espacio *D*-dimensional existen hiper-rectángulos los cuales pueden estar rotados en uno o más ejes (Figura 2-1). Para la resolución de problemas de clasificación solo es de interés trabajar con aquellos que tienen todas sus caras paralelas a los ejes (Figura 2-2). La razón por la cual solo es de interés este tipo de hiper-rectángulos es simple y será explicada luego de definir algunos conceptos.

Resulta de interés poder definir de manera simple los límites de un hiper-rectángulo en cada una de las dimensiones del espacio, ya que estos límites pueden ser utilizados como descriptores de un determinado conjunto de datos. La principal ventaja que posee un hiper-rectángulo que posee todas sus caras paralelas a los ejes, es que cada una de estas caras presenta un único valor en una dimensión determinada. En cada eje *i* del espacio un hiper-rectángulo tiene dos caras ortogonales a dicho eje, por lo tanto cada cara tiene un valor distinto, así es posible utilizar estos dos valores para definir los límites del hiper-rectángulo en la dimensión *i*. De esta manera, cada dimensión queda definida dentro de un intervalo cerrado y los límites de todo el hiper-rectángulo queda definido por todos los intervalos de cada una de las dimensiones del espacio. Un hiper-rectángulo de estas características tiene la particularidad que es posible describir todos los valores contenidos en él definiendo solo los valores mínimos y máximos para cada dimensión.

> **Definición 1**: Sea *H* un hiper-rectángulo con todas sus caras paralelas a los ejes, cada dimensión *i* está definida en el intervalo [Hn_i, Hx_i]. Los límites de *H* son definidos por los puntos *D*-dimensionales (Hn_1, Hn_2, Hn_3, …, Hn_D) y (Hx_1, Hx_2, Hx_3, …, Hx_D).
>
> **Definición 2**: Un punto *p* en el espacio *D*-dimensional está incluido en un hiper-rectángulo *H* si se cumple la siguiente condición:

$$Hn_i \leq p_i \leq Hx_i \,, \forall i = 1..D$$

A partir de un hiper-rectángulo con la característica de poseer todas sus caras paralelas a los ejes, es posible describir sus límites utilizando el lenguaje natural. Por ejemplo, si de una base de datos de pacientes, donde cada fila de la base de datos consta de información referida a la edad, peso y talla de un paciente, el modelo de datos a armar es definido en el espacio de tres dimensiones. Si el modelo de datos posee un hiper-rectángulo cuyos límites son *Hn*= (20, 60, 100) y *Hx* = (30, 95, 125) es posible expresar al hiper-rectángulo de manera natural diciendo que un grupo de pacientes determinado se caracteriza por estar formado por pacientes que tienen entre 20 y 30 años, su peso está entre 60 y 95 Kg. y su talla es mayor que 100 cm. y menor a 125 cm.

Por este motivo, la estrategia presentada en esta tesis trabaja solo con hiper-rectángulos cuyas caras son paralelas a los ejes, ya que de esta manera y como se verá en la sección 5 es posible extraer de un hiper-rectángulo una regla que puede ser expresada con el lenguaje natural. Nótese, que cualquier hiper-rectángulo que tenga al menos una cara que no sea paralela a un eje, presenta el inconveniente de que el límite establecido por dicha cara será imposible de poder expresarlo solo con un valor y, por lo tanto, imposible de extraer una regla utilizando el lenguaje natural.

[1] En geometría un politopo es un polígono de *D*-dimensiones.

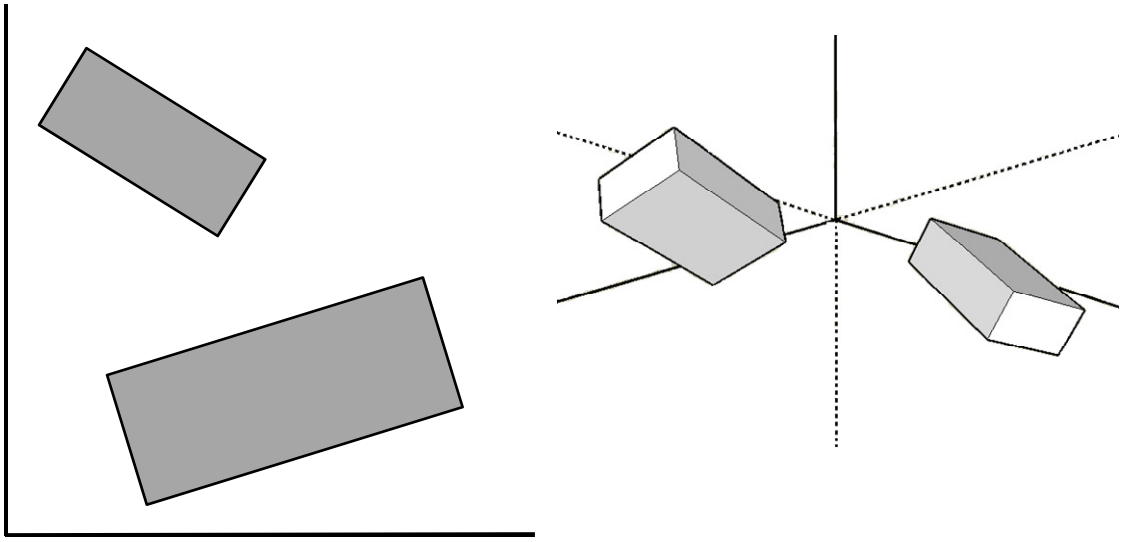

Figura 2-1. Rectángulos y cuboides, cada uno en su correspondiente espacio, los cuales se encuentran rotados con respecto a los ejes del espacio.

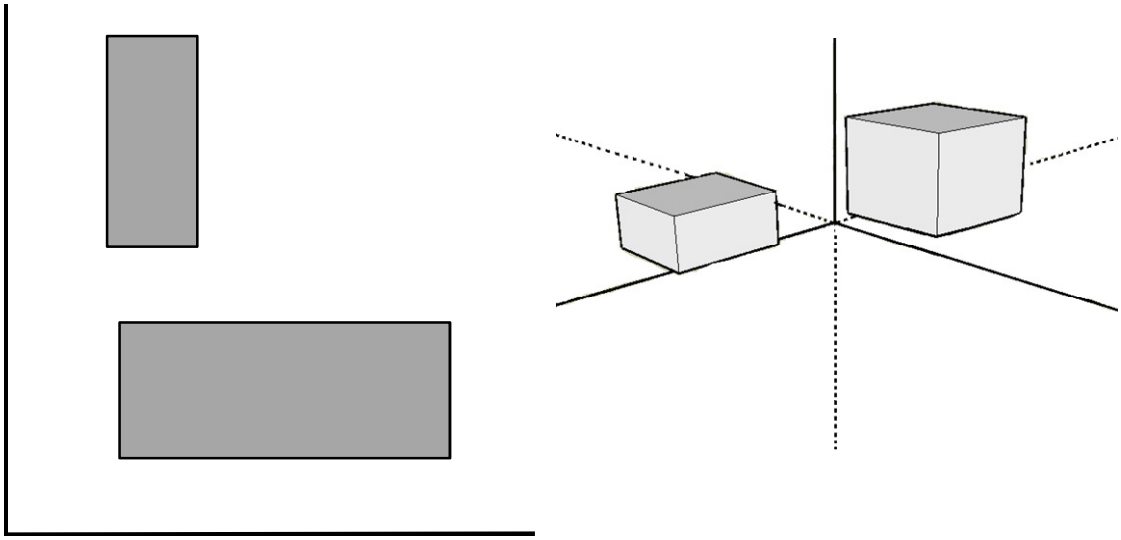

Figura 2-2. Rectángulos y cuboides, cada uno en su correspondiente espacio, los cuales tienen la característica de poseer todas sus caras paralelas a los ejes.

1.1. Creación de hiper-rectángulos a partir de una base de datos

En una base de datos donde todos sus atributos están definidos de manera numérica, ya sea con valores enteros o valores reales, es posible ver cada fila de la tabla como un punto en el espacio *D*-dimensional donde *D* es la cantidad de atributos numéricos de la tabla. A partir de esta tabla es posible armar un hiper-rectángulo lo suficientemente grande como para que incluya a todos los datos de la base de datos. A tal hiper-rectángulo se lo define como el "hiper-rectángulo representativo de un conjunto de datos" ya que dicho hiper-rectángulo incluye en su interior a todos los datos de la base de datos con la cual se desea trabajar.

Definición 3: Sea *B* una tabla de una base de datos con *N* filas y *D* atributos numéricos en cada fila. Un hiper-rectángulo representativo *H* de *B* es aquel que cumple las siguientes condiciones:

$$Hn_i \leq b_i \leq Hx_i \,, \forall i = 1..D \,, \forall b \in B$$

Es posible ver que en un espacio de *D* dimensiones no acotado existen infinitos hiper-rectángulos representativos de una base de datos. Se denomina "hiper-rectángulo representativo mínimo de *B*" a aquel hiper-rectángulo representativo de *B* que tenga el menor volumen. Este hiper-rectángulo representativo mínimo puede ser formado utilizando los valores mínimos y máximos de cada atributo entre todas las filas de *B*.

Definición 4: Sea *B* una tabla de una base de datos con *N* filas y *D* atributos numéricos en cada fila. El hiper-rectángulo representativo mínimo *H* de *B* es aquel que cumple con lo siguiente:

$$Hn_i = \min\{b_i \mid b \in B\} \,, \forall i = 1..N$$

$$Hx_i = \max\{b_i \mid b \in B\} \,, \forall i = 1..N$$

Si cada fila de la tabla de una base de datos *B* además de tener *D* atributos numéricos tiene un *D*+1 atributo que determina la clase a la cual pertenece el dato de dicha fila, es posible armar para un subconjunto $B' \subset B$ su hiper-rectángulo representativo mínimo, tal que todos los elementos de B' correspondan a la misma clase. De esta manera, si en la base de datos *B* existen *C* clases, es posible armar para cada clase C_a su hiper-rectángulo representativo mínimo H_a. Nótese que si se etiqueta a H_a con el valor C_a se puede considerar un hiper-rectángulo representativo de la clase C_a.

El poder lograr armar un hiper-rectángulo representante para una clase es suficiente para poder extraer una regla de clasificación que le transfiera al usuario del modelo de datos las características de los datos pertenecientes a dicha clase como conocimiento útil. En principio esta tarea parece demasiado sencilla y lo es si los datos pertenecientes a distintas clases pudieran ser representados con hiper-rectángulos que no presenten intersección en el espacio (Figura 2-3) ya que en estos casos cada hiper-rectángulo, con los valores de sus límites, explican por sí mismos las características de cada clase y así es posible extraer reglas de clasificación que resultan disjuntas.

En la práctica, estos casos no son comunes de encontrar y es de esperar que los hiper-rectángulos representantes de cada clase presenten una intersección en el espacio (Figura 2-4). El objetivo central de la técnica propuesta en esta tesis y, que es presentada en la sección 0, es la de detectar las intersecciones entre hiper-rectángulos representantes de distintas clases y, o bien, eliminarlas o disminuirlas hasta lograr resultados que resulten satisfactorios para el usuario del modelo de datos. Para lograr esto se procede a realizar divisiones de los hiper-rectángulos y/o disminución del volumen de los mismos tal como se explica en la sección 2 hasta lograr la eliminación completa de intersecciones o lograr una intersección mínima aceptable.

2. Superposiciones

La superposición en el espacio de dos o más hiper-rectángulos representativos de distintas clases no es deseable ya que, si ese caso ocurre, un mismo dato podría pertenecer a más de una clase al estar incluido en ambos hiper-rectángulos. Una técnica de clasificación para poder ser confiable y precisa no puede ofrecer como resultado una predicción tal que para un determinado dato, este pueda ser clasificado en más de una clase.

El objetivo de la técnica propuesta en esta tesis es el de reducir todo lo posible las superposiciones existentes entre hiper-rectángulos representativos de distintas clases y en el mejor de los casos eliminar toda superposición existente.

Definición 5: Sean *H* y *J* dos hiper-rectángulos. Existe superposición en el espacio entre ambos si para cada una de las *D* dimensiones se cumple al menos una de las siguientes cuatro condiciones:

1. $Hn_i \leq Jn_i \leq Hx_i$
2. $Hn_i \leq Jx_i \leq Hx_i$
3. $Jn_i \leq Hn_i \leq Jx_i$
4. $Jn_i \leq Hx_i \leq Jx_i$

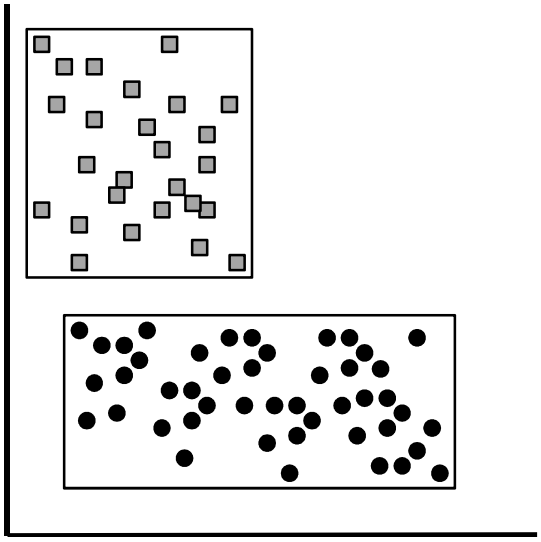

Figura 2-3. En la figura se pueden ver datos de dos clases distintas (cuadrados y círculos) y sus correspondientes hiper-rectángulos representativos mínimos, los cuales no presentan superposición en el espacio entre si.

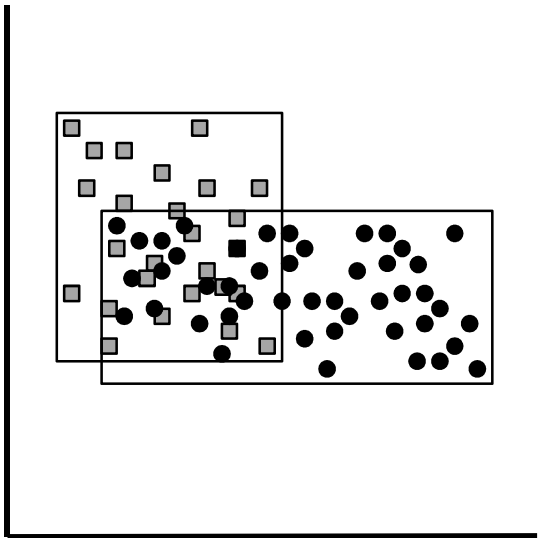

Figura 2-4. En la figura se pueden ver datos de dos clases distintas (cuadrados y círculos) y sus correspondientes hiper-rectángulos representativos mínimos, los cuales están intersectados el uno al otro.

Por lo tanto, solo es necesario encontrar una dimensión *i* tal que no cumpla ninguna de las cuatro condiciones anteriores para poder asegurar que no existe una superposición entre ambos hiper-rectángulos.

Definición 6: Dos hiper-rectángulos *H* y *J* no presentan superposición en el espacio si existe una dimensión *i* donde se cumple una de estas condiciones:

1. $Hx_i < Jn_i$
2. $Hn_i > Jx_i$

Por lo tanto, el objetivo de la estrategia propuesta en esta tesis es, dada una superposición, encontrar una dimensión *i* donde poder modificar uno o ambos hiper-rectángulos y así eliminar dicha superposición. La elección de la dimensión *i* en la cual llevar a cabo el ajuste de hiper-rectángulos, ya sea dividiendo uno de ellos o ajustando uno o ambos, se basa en el cálculo de una serie de índices de superposición que se presentan en detalle en la sección 3.

Definición 7: Sean *H* y *J* dos hiper-rectángulos que presentan superposición en el espacio, la superposición formada es a su vez otro hiper-rectángulo *S* cuyos límites quedan definidos de la siguiente manera:

$$Sn_i = \max(Hn_i, Jn_i)$$

$$Sx_i = \min(Hx_i, Jx_i)$$

Resulta interesante destacar que si entre dos hiper-rectángulos *H* y *J* ocurre una intersección *S* y se cumple que *S* = *H* entonces significa que *H* está incluido completamente en *J*. Si al mismo tiempo se cumple que *S* = *H* = *J* entonces significa que ambos hiper-rectángulos *H* y *J* son iguales.

2.1. Tipos de superposiciones

En esta tesis se destacan tres tipos de superposiciones que pueden ocurrir entre dos hiper-rectángulos, los cuales serán estudiados por separado. Para eliminar cada uno de estos tipos de superposiciones se propone una solución diferente.

2.1.1. Superposición sin datos involucrados

El caso más simple para resolver ocurre cuando se presenta una superposición donde no hay datos involucrados (Figura 2-5). Este tipo de intersección es el más fácil de solucionar ya que basta con la división de uno de los dos hiper-rectángulos (Figura 2-6).

Según el grado de superposición de un hiper-rectángulo contra el otro se procede a dividir el que se encuentre más involucrado. En el caso mostrado en la Figura 2-5 no se puede destacar que un hiper-rectángulo esté más involucrado que el otro en la superposición. Ante la pregunta de cual de los dos hiper-rectángulos dividir, no se sugiere ninguna solución en particular, ya que la solución más adecuada dependerá de cada problema en particular y de las necesidades que tiene el usuario en función de la importancia que tengan las distintas clases de datos.

En problemas complejos de la vida real es posible encontrar más de dos clases involucradas en el proceso de clasificación, la división de uno u otro hiper-rectángulo es clave para lograr como resultado un modelo de datos confiable. Si un hiper-rectángulo en particular tiene superposición con al menos otros dos hiper-rectángulos, la decisión de que hiper-rectángulo dividir provocará un resultado y un modelo de datos diferente (Figura 2-7).

En problemas que presentan un cierto grado de complicación, donde intervienen muchas clases y se forman muchas intersecciones entonces la mejor solución para resolver no es clara a simple vista. Aunque es posible asegurar que la mejor solución es aquella que menos hiper-rectángulos finales obtenga.

Si bien la utilización de índices de superposición en el proceso de armado del modelo, propuestos en esta tesis, intenta resolver estas incertidumbres, se verá en la sección 0 una solución alternativa mediante la participación de un experto en el dominio del problema.

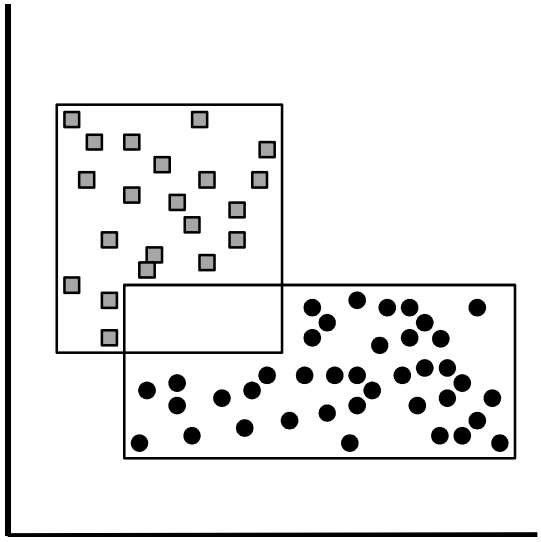

Figura 2-5. Una intersección entre hiper-rectángulos donde en la superposición entre ambos no hay datos de ninguna de las dos clases.

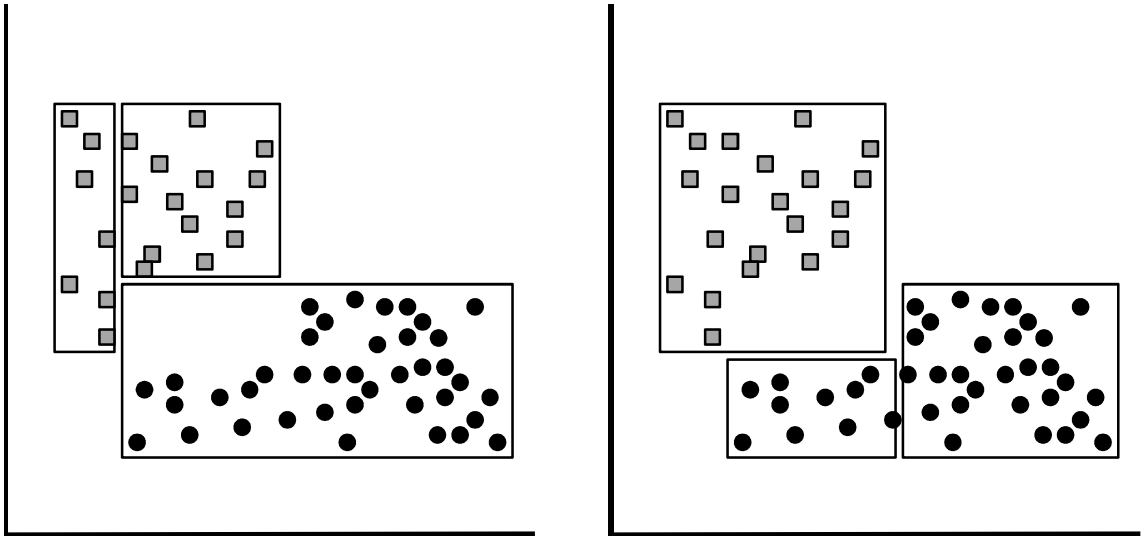

Figura 2-6. Eliminación de la superposición mostrada en la Figura 2-5. Se muestran las dos posibles soluciones mediante la división de uno u otro hiper-rectángulo.

2.1.2. Superposición con datos de una clase

Este tipo de superposiciones es similar al anterior con la diferencia que para lograr el menor número de hiper-rectángulos la división debe hacerse en el hiper-rectángulo que no presenta datos en dicha superposición (Figura 2-8). Como ocurre en las superposiciones sin datos involucrados, cuando el número de clases involucradas o el número de intersecciones son elevados, entonces la decisión de que hiper-rectángulo dividir no es trivial y al mismo tiempo resulta clave para obtener un modelo de datos confiable.

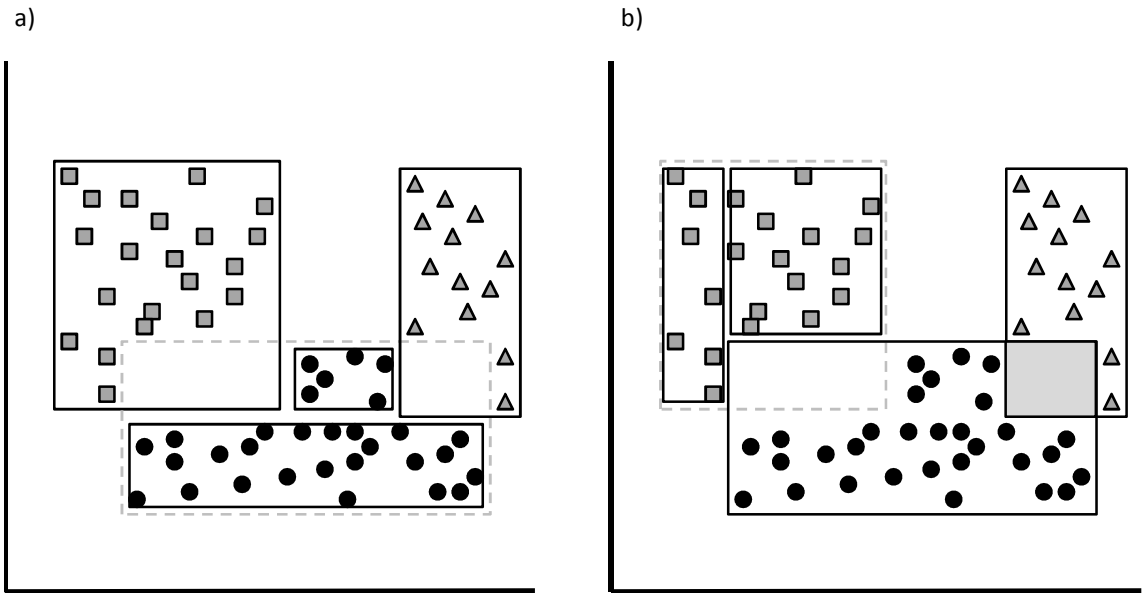

Figura 2-7. En a) se muestra como dividiendo un hiper-rectángulo es suficiente para eliminar dos superposiciones al mismo tiempo. En b) se muestra el caso de como dividir el hiper-rectángulo de la clase cuadrados solo elimina una superposición, por lo que es necesaria otra división de hiper-rectángulos para eliminar la segunda superposición (zona gris entre la clase de círculos y la de triángulos). En líneas punteadas se muestra el hiper-rectángulo original que fue dividido.

En los casos donde un hiper-rectángulo *H* está incluido completamente en otro hiper-rectángulo *J* entonces es posible ver que cualquier tipo de división de *H* no causa la eliminación de la superposición ya que los nuevos hiper-rectángulos producto de la división seguirán incluidos en *J* (Figura 2-9 a), en estos casos se procede con la obligada división de *J* para lograr eliminar la superposición. Según como se de la superposición de *H* contra la distribución de los datos de *J* puede ocasionar que *J* se divida solo en dos nuevos hiper-rectángulos (Figura 2-9 b) o hasta cuatro hiper-rectángulos (Figura 2-10).

En estos casos es muy importante la dimensión de los datos, ya que cuánto más alta sea la dimensión del problema más hiper-rectángulos serán los que se generen al momento de dividir un hiper-rectángulo. En espacios de más de dos dimensiones, la cantidad mínima de hiper-rectángulos que se pueden generar es siempre dos y la cantidad máxima que se generen será 2**D*.

2.1.3. Superposición con datos de ambas clases

Este tipo de superposiciones son las más complejas de tratar ya que la simple división de un hiper-rectángulo produce que datos de una clase queden fuera de los límites de sus correspondientes hiper-rectángulos representativos (Figura 2-11). La aparición de este tipo de superposiciones trae aparejada una posible pérdida de precisión en el modelo de datos, ya que no siempre existen soluciones para eliminar las superposiciones asegurando una precisión completa. Aunque si bien es posible dividir los hiper-rectángulos una y otra vez hasta lograr una precisión del 100% formando hiper-rectángulos representativos de uno o dos datos, no es deseable llegar a este extremo ya que se obtendría un modelo de datos del cual se extrae un número muy elevado de reglas de clasificación. Por lo tanto, es deseable encontrar un equilibrio entre armar un modelo de datos con una precisión del 100% y minimizar la cantidad de reglas. Esta es una decisión que deberá tomar el usuario final del modelo de datos.

Existen casos donde en este tipo de superposiciones es posible asegurar una completa precisión con pocas divisiones (Figura 2-12), aunque no es un caso frecuente de encontrar. Para poder hallar estos casos de pocas divisiones o decidir qué operación realizar sobre una intersección para minimizar la pérdida de precisión, la técnica propuesta en esta tesis utiliza los índices de superposición propuestos y que se analizan en detalle en la sección 3.

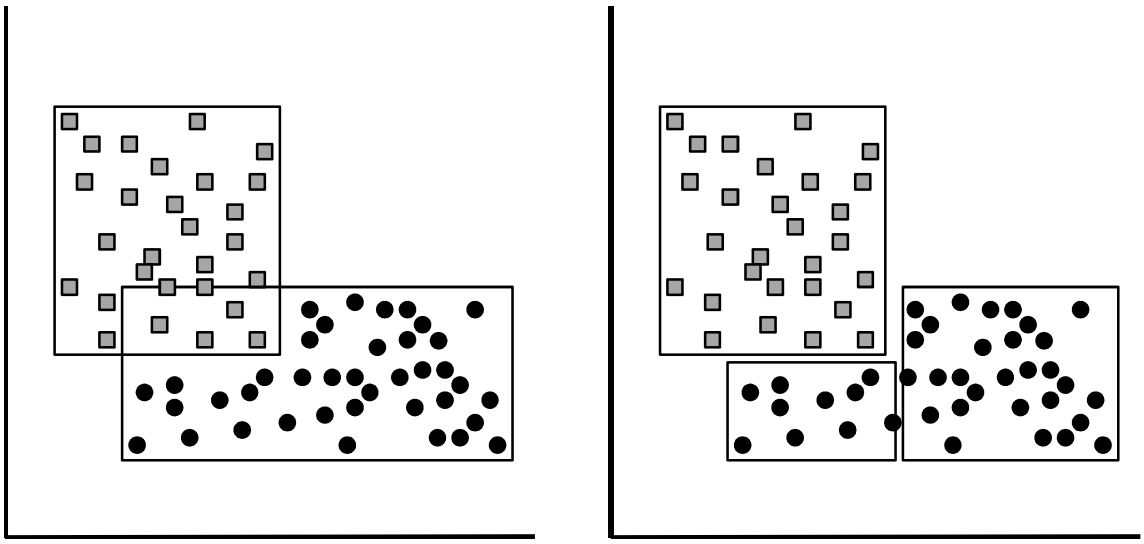

Figura 2-8. Eliminación de la superposición entre dos hiper-rectángulos. La división se realiza en el hiper-rectángulo que no presenta datos en la superposición.

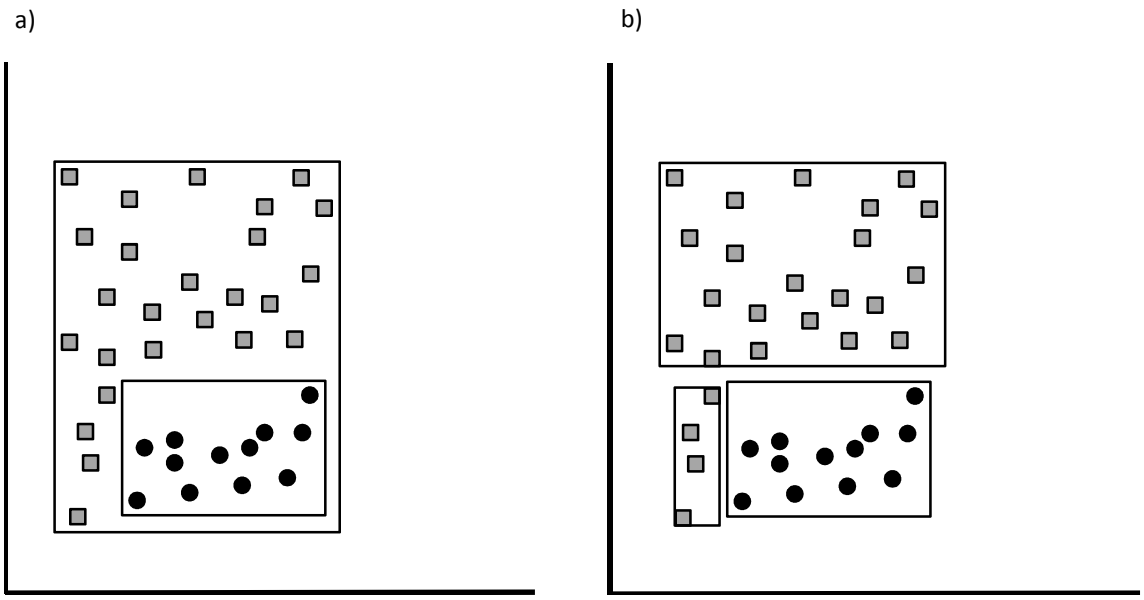

Figura 2-9. Un hiper-rectángulo H incluido completamente dentro de otro, cualquier división de H no elimina la superposición.

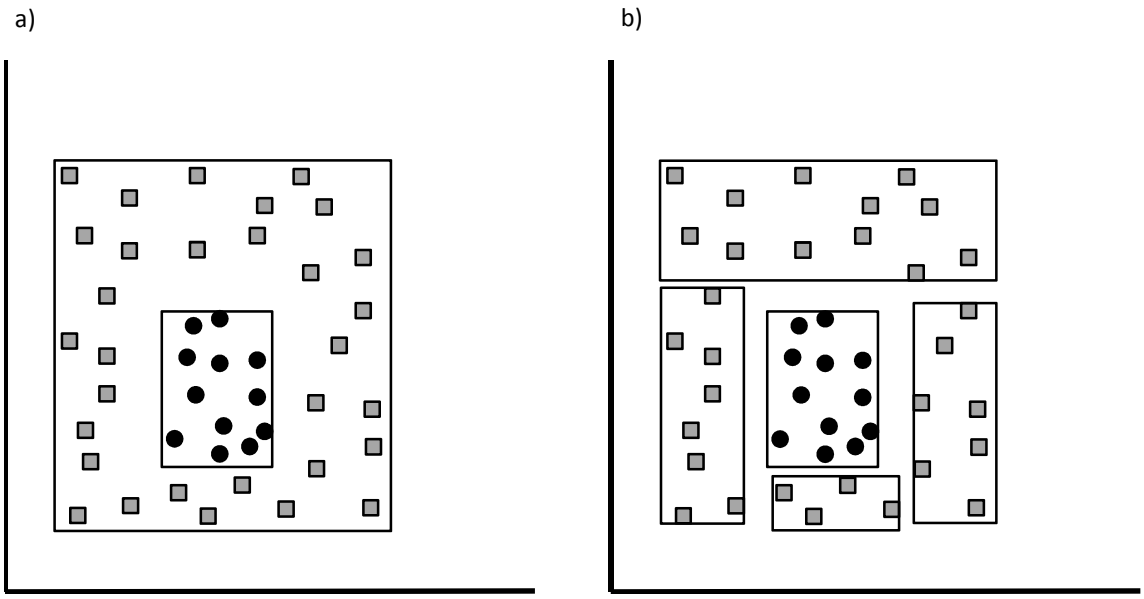

Figura 2-10. Un caso donde un hiper-rectángulo es dividido produciendo cuatro nuevos hiper-rectángulos.

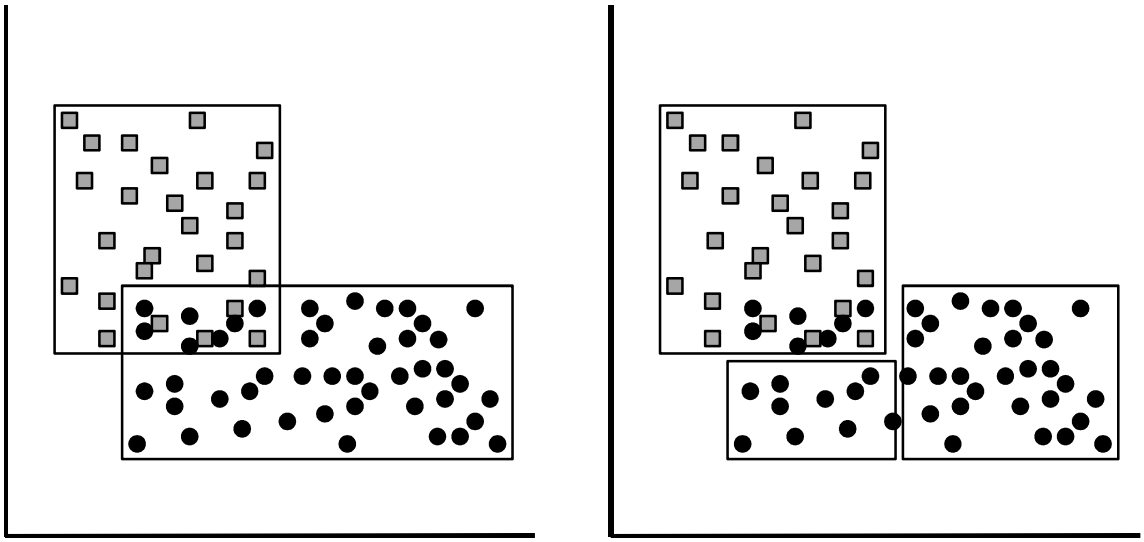

Figura 2-11. Eliminación de la superposición entre dos hiper-rectángulos mediante la división de uno de ellos. Esta división produce que queden datos de su clase fuera de los hiper-rectángulos representativos.

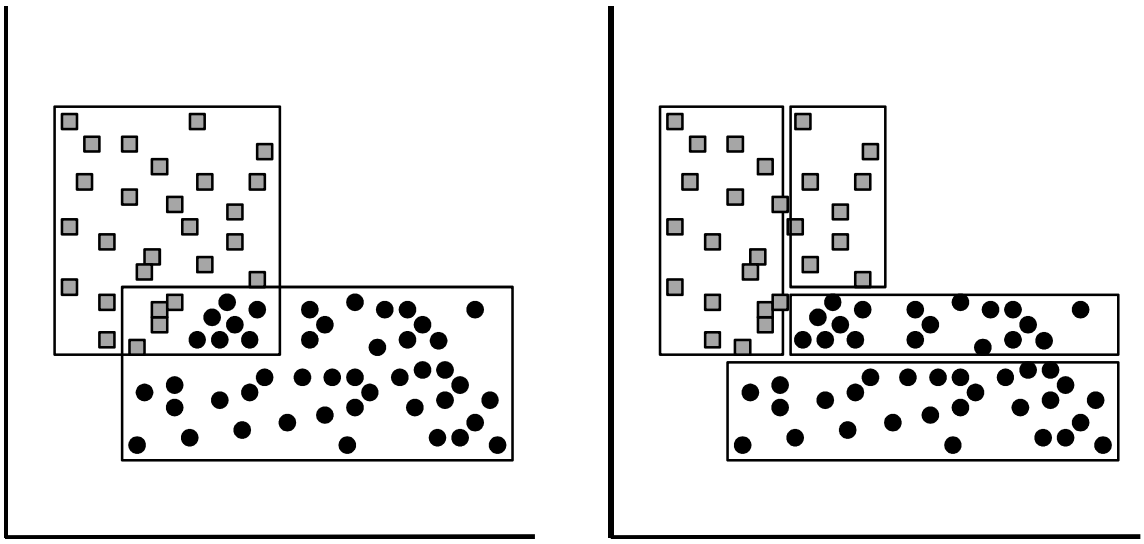

Figura 2-12. Ejemplo hipotético donde es posible eliminar una superposición de dos hiper-rectángulos cuando están presenten datos de ambas clases utilizando un número mínimo de divisiones y asegurando una completa precisión en el modelo de datos.

2.2. Eliminación de superposiciones

Como se mencionó al principio de esta sección dos hiper-rectángulos *H* y *J* no se superponen si al menos en una dimensión *i* el límite inferior de uno es mayor que el límite superior del otro (Definición 6). Por lo tanto la tarea de eliminar una superposición entre dos hiper-rectángulos *H* y *J* consiste en dividir a uno de los dos en otros dos hiper-rectángulos. Para eso se "corta" a *H* o a *J* por un valor en la dimensión *i*, y los hiper-rectángulos que resultan del "corte" se ajustan a los datos para convertirlos en hiper-rectángulos representativos mínimos. El dividir un hiper-rectángulo por una u otra dimensión hará que el modelo de datos produzca resultados diferentes (Figura 2-13).

Cómo se explicó anteriormente, muchas veces no es posible encontrar una división que asegure un modelo de datos completamente preciso. En estos casos hay que encontrar un equilibrio entre el número de reglas extraídas y un modelo de datos con una buena precisión. En estos casos no se lleva a cabo la división de uno de los dos hiper-rectángulos sino que se ajustan los volúmenes de uno o ambos hiper-rectángulos participantes. Este ajuste puede hacerse en favor de uno u otro hiper-rectángulo o bien encontrando un punto medio con algún criterio (Figura 2-14). La técnica propuesta en esta tesis no propone ningún método en particular ya que esta decisión dependerá exclusivamente del problema que se quiera resolver.

La elección de la dimensión *i* para llevar a cabo el ajuste o la división se hace mediante el cálculo de unos índices de superposición que se analizan con más detalle en la sección 3.

En relación a los distintos tipos de superposición vistos en la sección 2.1 se detallan los correspondientes procedimientos para eliminar la superposición.

2.2.1. Sin datos involucrados

Cuando en la superposición no hay datos de ambas clases entonces se procede a realizar la división de uno de los hiper-rectángulos involucrados. En esta tesis no se propone la elección de un hiper-rectángulo en particular ya que, según que hiper-rectángulo se divida, puede producir modelos de datos distintos y esto dependerá exclusivamente del problema que se quiera resolver (Figura 2-6).

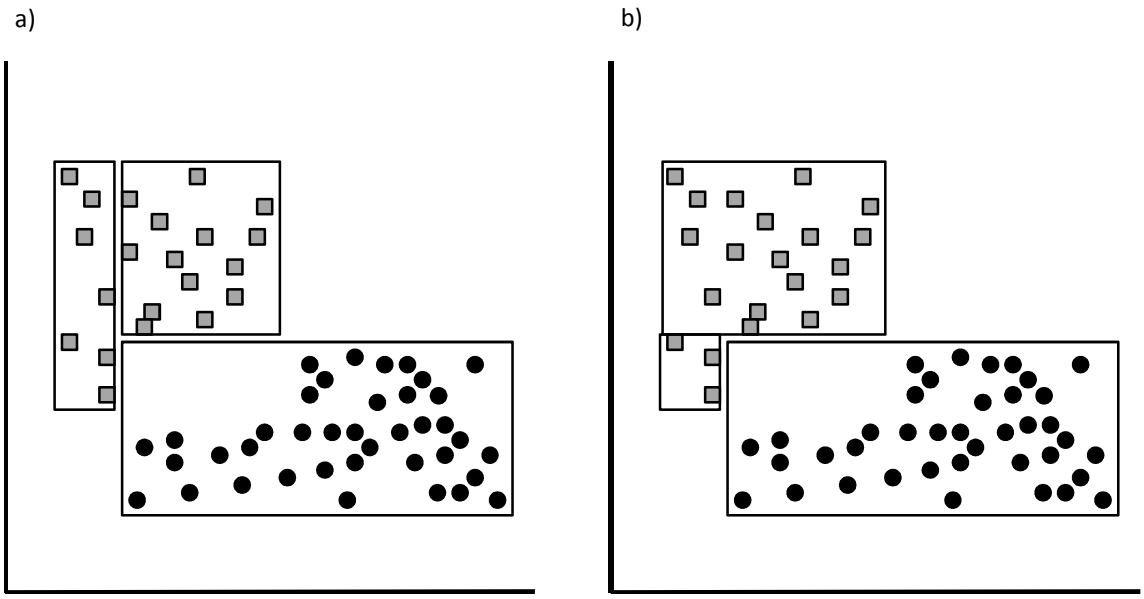

Figura 2-13. Dos posibles formas de eliminar una superposición entre dos hiper-rectángulos, dividiendo el mismo hiper-rectángulo por un valor en el eje X (a) o por un valor en el eje Y (b).

Decidido que hiper-rectángulo *H* se va a dividir entonces se procede a dividir a *H* por la dimensión *i*. La elección de la dimensión *i* se verá con detalle en la sección 3. El valor *v* por el cual cortar *H* estará dado por el límite inferior o superior del otro hiper-rectángulo *J* en la dimensión *i* y queda establecido de la siguiente manera:

$$v = \begin{cases} Jn_i & si\ Hn_i < Jn_i < Hx_i \\ Jx_i & si\ Hn_i < Jx_i < Hx_i \end{cases}$$

Del conjunto de datos representados en *H* se forman dos nuevos hiper-rectángulos de la siguiente manera:

$$H_1 = \{h \mid h \in H, h_i < v\}$$

$$H_2 = \{h \mid h \in H, h_i \geq v\}$$

Finalmente para los dos conjuntos H_1 y H_2 formados se arman los correspondientes hiper-rectángulos representativos mínimos como se detalla en la definición 4.

2.2.2. Con datos de una clase en la superposición

Para los problemas similares a los de la Figura 2-8 se procede a dividir el hiper-rectángulo que no presenta datos en la superposición de la misma forma que se detallo en la sección anterior.

Los casos especiales, que se dan cuando un hiper-rectángulo está incluido dentro de otro requieren más operaciones ya que la simple división por un valor en una dimensión no es suficiente para la eliminación de la superposición. Este tipo de casos, pueden ser solucionados utilizando un proceso iterativo con las ecuaciones presentadas en la sección anterior. Para cada dimensión se realizan cortes en los límites del hiper-rectángulo incluido hasta eliminar todas las superposiciones. Este proceso irá creando un hiper-rectángulo por vez a medida que se realiza un corte.

El pseudocódigo de este procedimiento iterativo es el siguiente:

```
J = hiper-rectángulo a dividir
H = hiper-rectángulo que está incluido en J
i = 1
mientras exista superposición entre H y J hacer
        si (Jni < hni < Jxi) entonces
                J1 = todos los datos j de J tal que ji < hni
                J = J – J1
        fin si
        si (Jni < hxi < Jxi) entonces
                J1 = todos los datos j de J tal que ji > hni
                J = J – J1
        fin si
        i = i + 1
fin mientras
```

2.2.3. Con datos de ambas clases

Este es el caso más complicado de resolver ya que no es posible encontrar un corte de hiper-rectángulos tal que logre la eliminación de la superposición y por lo tanto es posible tomar distintas decisiones de cómo resolver la superposición: 1) cortar en favor de una clase, 2) cortar en favor de la otra clase y 3) encontrar un punto de corte intermedio (Figura 2-14). La decisión de que opción elegir para resolver estos casos queda en manos del usuario del modelo ya que utilizar una u otra forma depende exclusivamente del problema que se quiere resolver.

Existe una cuarta opción que se explica en la sección 4.2.2.1, luego de presentar los índices de superposición (sección 3). Esta cuarta alternativa consiste en dividir ambos hiper-rectángulos por un valor de una cierta dimensión *i* generando así cuatro nuevos hiper-rectángulos; dos para cada clase. Estos nuevos cuatro hiper-rectángulos presentan dos nuevas superposiciones, donde cada una está formada por un par de los hiper-rectángulos creados (Figura 2-15). Estas nuevas superposiciones pueden volver a analizarse y buscar si existe otra dimensión por la cual cortar los nuevos hiper-rectángulos formados y así eliminar las superposiciones.

3. Índices

En la sección 2 se presentaron diferentes tipos de superposición y la forma de eliminar o minimizar la superposición dependiendo de su tipo. También se mostró que la eliminación se hace modificando los hiper-rectángulos en una dimensión *i* en particular ya sea por la división de uno de ellos o por la reducción de volumen de uno o ambos. La elección de esta dimensión *i* en la cual llevar a cabo la tarea de ajuste se basa en el cálculo de una serie de índices de superposición, donde cada uno de ellos mide un aspecto diferente de la superposición. Estos índices se combinan en un único valor para cada dimensión de manera que el índice con el valor más alto determina la dimensión por la cual llevar a cabo el ajuste.

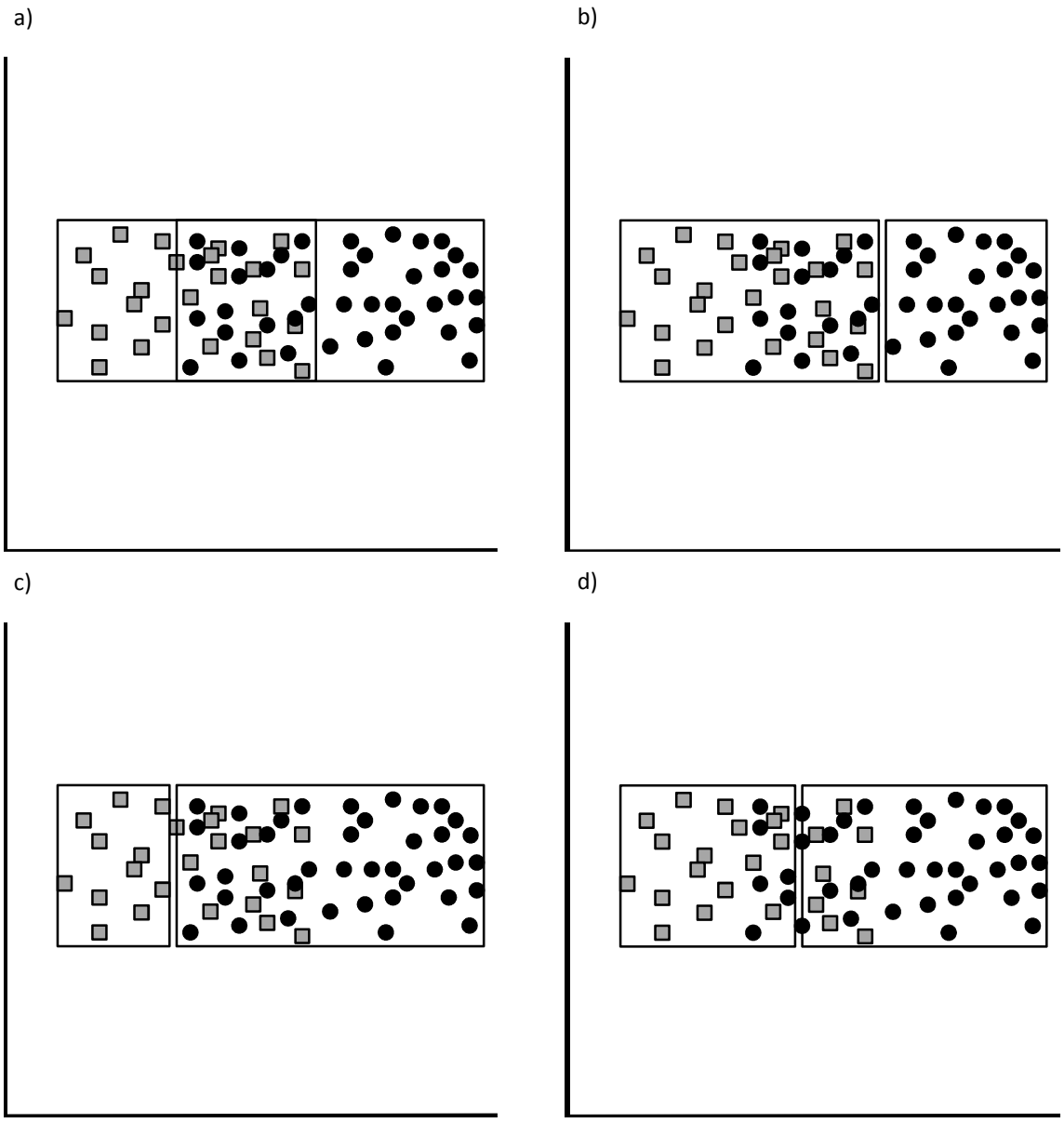

Figura 2-14. Tres posibles formas de eliminar una superposición entre dos hiper-rectángulos (a), ajustando los límites de los mismos en favor de una de las dos clases (b) y (c), o bien encontrando algún punto medio (d).

Cada índice es calculado para una dimensión *i* dada, y se calculan todos los índices para todas las dimensiones. Para llevar a cabo el cálculo de un índice se utilizan los valores mínimos y máximos de ambos hiper-rectángulos para la dimensión *i*. También se utilizan los valores del atributo *i* de los datos que caen en la superposición. De esta manera se define una superposición de área que queda determinada por los límites de la superposición y una superposición de datos que queda determinada por los datos incluidos en la superposición.

Definición 8: Sea *S* una superposición entre dos hiper-rectángulos, el límite de área para la dimensión *i* está determinado por el intervalo formado por los valores sn_i y sx_i (Definición 7).

Se denomina "subconjunto de datos participantes" de un hiper-rectángulo *H* al subconjunto de datos representados por *H* que están incluidos en la superposición.

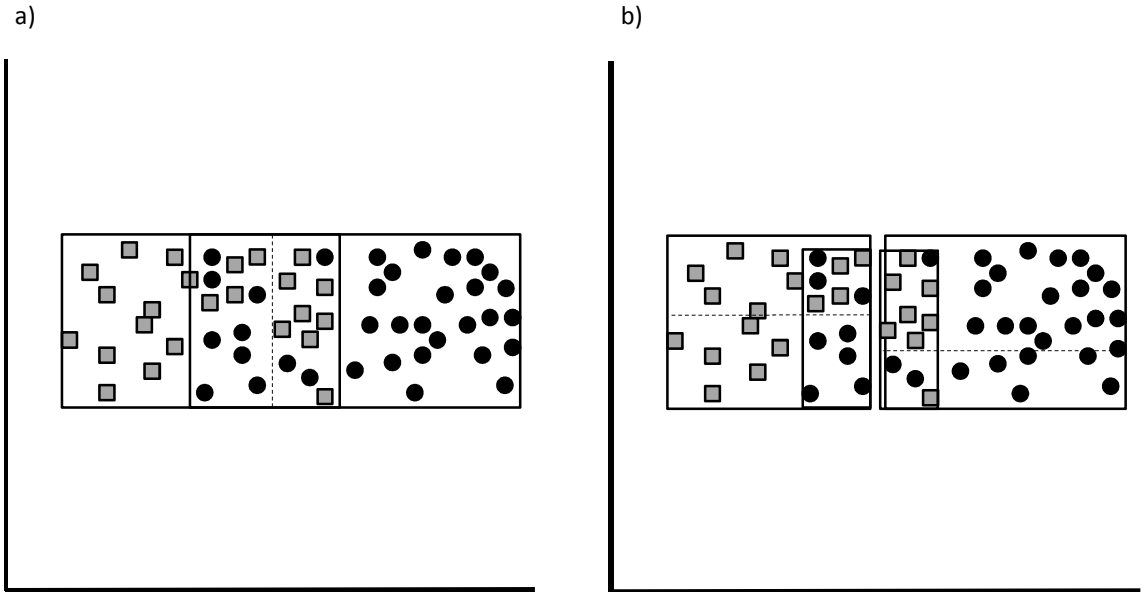

Figura 2-15. División de los dos hiper-rectángulos formando dos nuevas superposiciones. En a) se muestra una línea punteada por el cual se dividen ambos hiper-rectángulos. En b) se muestran como quedan los nuevos cuatro hiper-rectángulos y con líneas punteadas se muestran posibles valores del eje Y que ayudarían a dividir los hiper-rectángulos para eliminar la superposición.

Definición 9: Sea *S* una superposición entre dos hiper-rectángulos *H* y *J*, el subconjunto de datos participantes *E* de *H* queda definido de la siguiente manera:

$$E = \{e \mid e \in H \; y \; e \in S\}$$

Para cada dimensión *i*, el subconjunto *E* tiene sus correspondientes valores mínimos y máximos los cuales se definen como En_i y Ex_i respectivamente. Dentro de una superposición entre dos hiper-rectángulos, los cuales presentan sus respectivos subconjuntos participantes *E* y *F*, los intervalos que se forman en una determinada dimensión pueden presentar una superposición la cual se denomina superposición de datos.

Definición 10: Sea *S* una superposición entre dos hiper-rectángulos y sean *E* y *F* los subconjuntos de datos participantes de ambos hiper-rectángulos presentes en *S*. Si en una dimensión *i* los intervalos [En_i y Ex_i] y [Fn_i y Fx_i] presentan una intersección, la superposición de datos *T* entre *E* y *F* en *S* queda formada por el intervalo [Tn_i y Tx_i], donde:

$$Tn_i = \max(En_i, Fn_i)$$

$$Tx_i = \min(Ex_i, Fx_i)$$

Sea *G* el subconjunto de *E* formado por aquellos datos del hiper-rectángulo *H* que caen dentro del intervalo de superposición de datos *T* que se denomina como "subconjunto de datos intersectados" de *H*.

Definición 11: Sea *E* un subconjunto de datos participante y *T* una superposición de datos. El subconjunto de datos intersectados *G* queda definido de la siguiente manera:

$$G = \{g \mid g \in E, Tn_i \leq g_i \leq Tx_i\}$$

En cada una de las dimensiones *i*, el subconjunto *G* tiene valores mínimos y máximos a los cuales se los denomina Gn_i y Gx_i respectivamente.

La Figura 2-16 muestra los conceptos de superposición de área y superposición de datos.

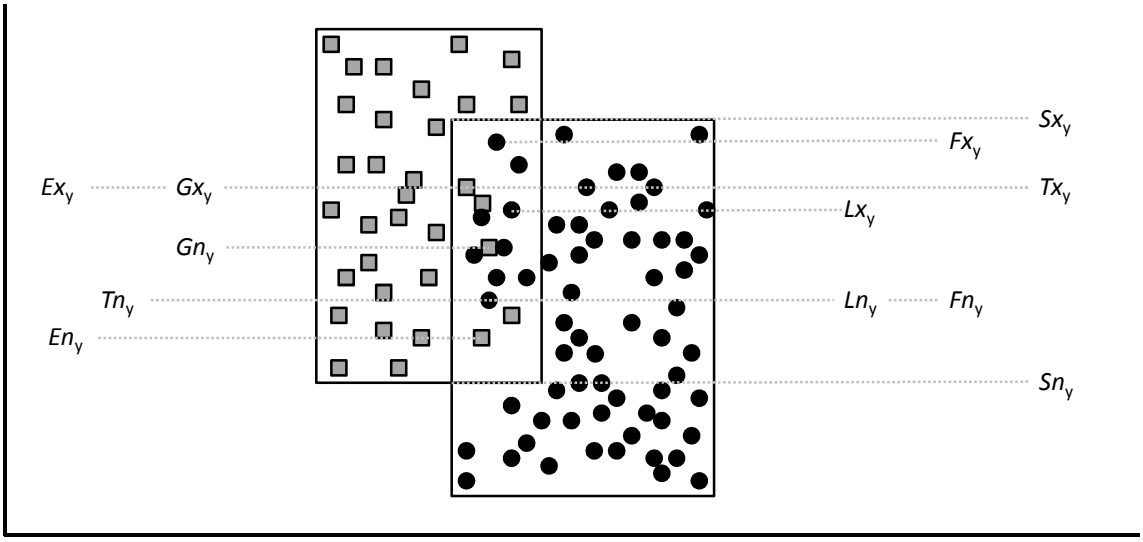

Figura 2-16. Se muestra una intersección entre dos hiper-rectángulos y como quedan formados para la dimensión del eje Y el intervalo de intersección de área S, los intervalos de los subconjuntos de datos participantes para ambas clases (E y F), el intervalo de intersección de datos T y los intervalos del subconjunto de datos intersectados de ambas clases (G y L).

3.1. Índices de superposición

En esta sección se analizan en detalle seis índices propuestos en esta tesis para medir diferentes aspectos de una superposición. Cada uno de estos índices, denominados índices Z, son calculados para los dos hiper-rectángulos presentes en una superposición *S* determinada y se calcula en cada una de las dimensiones del espacio del problema.

Los seis índices están diseñados para devolver un valor entre 0 y 1. El índice calculado en una dimensión *i* vale 0 cuando en esa dimensión no exista un aporte importante para la decisión de dividir los hiper-rectángulos por un valor en esa dimensión *i* y vale 1 cuando esa dimensión es candidata para dividir los hiper-rectángulos en la dimensión *i*.

Finalmente, se utilizan los valores calculados de los índices Z para el cálculo del índice Ω en cada una de las dimensiones. Este índice Ω será utilizado para determinar el grado de separabilidad de los hiper-rectángulos en dicha dimensión. Por lo tanto, la dimensión en la cual realizar el ajuste de hiper-rectángulos queda determinada por el índice Ω con mayor valor.

3.1.1. $Z1_i$ – Proporción del ancho de la intersección de área respecto al ancho del hiper-rectángulo

Para un hiper-rectángulo *H* presente en una superposición *S*, $Z1_i(H)$ se define como:

$$Z1_i(H) = 1 - (Sx_i - Sn_i)/(Hx_i - Hn_i)$$

Este índice mide cuan grande es el ancho de la superposición *S* en relación al hiper-rectángulo *H*. Este índice toma el valor 0 cuando el intervalo de superposición en la dimensión *i* es igual a la de *H*, que ocurre cuando un hiper-rectángulo está incluido completamente en el otro en la dimensión *i*. A medida que el ancho de la superposición en una dimensión *i* decrece, el valor de este índice tiende a 1.

Este índice le da más peso a los casos donde el ancho de una superposición en una dimensión es pequeña en relación al ancho del hiper-rectángulo, como lo muestra el ejemplo de la Figura 2-17 y la Tabla 2-1.

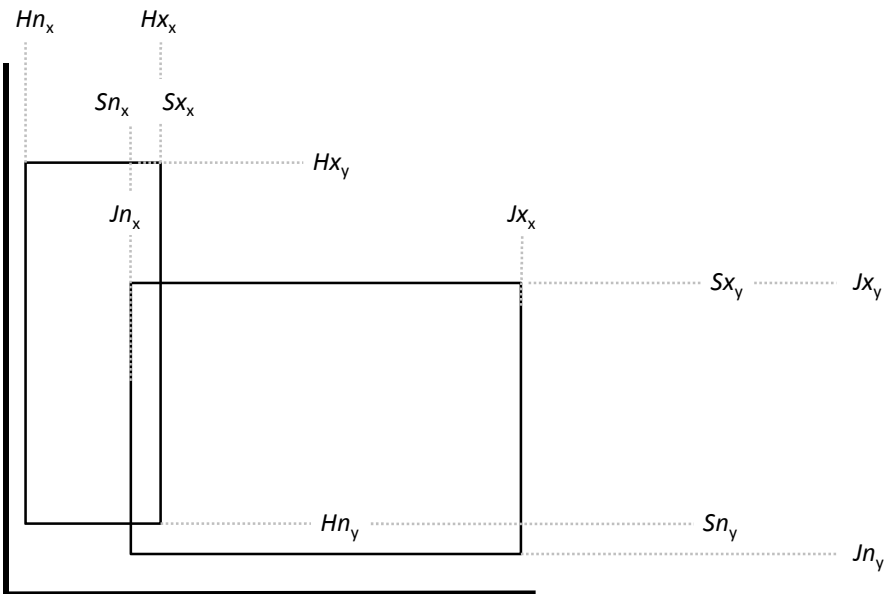

Figura 2-17. Ejemplo que muestra el aspecto que mide el índice $Z1_i$. Se calcula el índice para los dos hiper-rectángulos observando que en la dimensión del eje X el índice tiende a 1 mientras que en la dimensión del eje Y el valor del índice tiende a 0.

Tabla 2-1. Cálculo de los índices $Z1_x(H)$, $Z1_y(H)$, $Z1_x(J)$, $Z1_y(J)$ del ejemplo de la Figura 2-17.

	X	Y
Hn	3	9
Hx	21	57
Jn	17	5
Jx	69	41
Sn	17	9
Sx	21	41
Z1(*H*)	0,78	0,33
Z1(*J*)	0,92	0,11

3.1.2. $Z2_i$ – Proporción del ancho del intervalo de la intersección de datos con respecto al ancho del intervalo del subconjunto de datos participante

Para un hiper-rectángulo *H* presente en una superposición *S* y su correspondiente subconjunto de datos participante *E* y el intervalo de la superposición de datos *T*, $Z2_i(H)$ se define como:

$$Z2_i(H) = 1 - (Tx_i - Tn_i)/(Ex_i - En_i)$$

Este índice mide dentro de una superposición cuan diferente es el intervalo formado por los elementos de *E* y el intervalo formado por la superposición de datos *T*. Así, este índice tiende a 0 cuando la superposición de datos es muy similar al propio intervalo *E* y tiende a 1 cuando el intervalo *T* es pequeño en relación a *E*.

Hay que considerar que si la superposición *S* no tiene datos participantes de ambas clases, o bien, hay datos participantes pero no existe superposición de datos (ejemplo de la Figura 2-12) entonces este índice no puede ser calculado ya que o bien no es posible calcular *T*, al no existir intersección de datos, o bien no se puede calcular *E*, al no haber datos participantes de *H* dentro de *S*. Cualquiera sea el caso, se establece que el valor de $Z2_i$ sea igual a 1 ya que al no haber superposición de datos, la dimensión *i* puede ser candidata para la división de los hiper-rectángulos.

Este índice considera más importante para la elección de que dimensión utilizar para llevar a cabo la división de hiper-rectángulos a aquellos casos donde, si bien hay superposición de datos, el intervalo formado por esta

superposición se puede considerar despreciable en relación al intervalo de los datos pertenecientes a *E*. La Figura 2-18 junto con la Tabla 2-2 ilustran el funcionamiento de este índice.

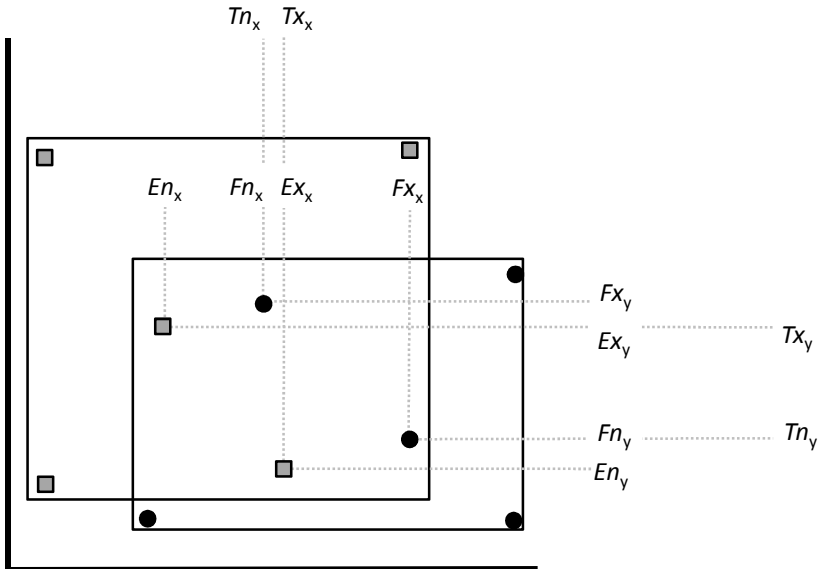

Figura 2-18. Ejemplo que muestra el aspecto que mide el índice $Z2_i$. Se calcula el índice para los dos hiper-rectángulos observando que en la dimensión del eje X el índice tiende a 1 mientras que en la dimensión del eje Y el valor que toma el índice tiende a 0.

Tabla 2-2. Cálculo de los índices $Z2_x(H)$, $Z2_y(H)$, $Z2_x(J)$, $Z2_y(J)$ del ejemplo de la Figura 2-18.

	X	Y
En	21	13
Ex	37	32
Fn	34	17
Fx	53	35
Tn	34	17
Tx	37	32
Z2(*H*)	0,81	0,21
Z2(*J*)	0,84	0,17

3.1.3. $Z3_i$ – Proporción del ancho del intervalo del subconjunto de datos intersectados en relación al ancho del intervalo del subconjunto de datos participante

Para un hiper-rectángulo *H* presente en una superposición *S* y su correspondiente subconjunto de datos intersectado *G* en el intervalo de la superposición de datos *T*, $Z3_i(H)$ se define como:

$$Z3_i(H) = 1 - (Gx_i - Gn_i)/(Tx_i - Tn_i)$$

El objetivo de este índice es el de medir cuán importante es la participación de los datos del subconjunto de datos intersectados dentro del intervalo de la intersección de datos. Así, este índice tiende a 0 cuando los datos pertenecientes a *G* abarcan todo el intervalo de intersección de datos y tiende a 1 cuanta más pequeña es la participación de los datos de *G* en *T*.

Este índice, al igual que el anterior, tampoco se podrá calcular si no existe superposición de datos para lo cual $Z3_i$ toma el valor 1 señalando que la dimensión *i* es candidata a ser elegida para la división de los hiper-rectángulos.

Este índice considera más importantes a los casos donde el intervalo formado por los datos del subconjunto de datos intersectados es ínfimo en relación al intervalo de intersección de datos tal como lo muestran la Figura 2-19 y la Tabla 2-3.

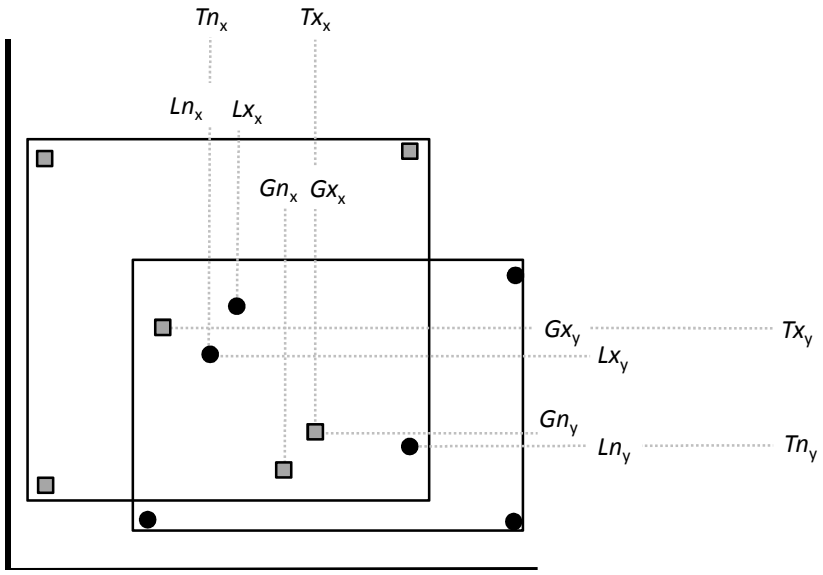

Figura 2-19. Ejemplo que muestra el aspecto que mide el índice $Z3_i$. Se calcula el índice para los dos hiper-rectángulos observando que en la dimensión del eje X el índice tiende a 1 mientras que en la dimensión del eje Y el valor que toma el índice tiende a 0.

Tabla 2-3. Cálculo de los índices $Z3_x(H)$, $Z3_y(H)$, $Z3_x(J)$, $Z3_y(J)$ del ejemplo de la Figura 2-19.

	X	Y
Gn	37	18
Gx	41	32
Ln	27	16
Lx	31	28
Tn	27	16
Tx	41	32
Z3(*H*)	0,71	0,13
Z3(*J*)	0,71	0,25

3.1.4. $Z4_i$ – Proporción del ancho del intervalo del subconjunto de datos participantes en relación al ancho de la superposición de área

Para un hiper-rectángulo *H* presente en una superposición *S* y su correspondiente subconjunto de datos participantes *E*, $Z4_i(H)$ se define como:

$$Z4_i(H) = 1 - (Ex_i - En_i)/(Sx_i - Sn_i)$$

Este índice tiene como objetivo medir el grado de participación de los datos involucrados en la intersección en relación al ancho de la intersección de área. Cuando los datos del subconjunto *E* abarquen el mayor espacio del área de superposición entonces el índice tiende a 0, por el contrario, tiende a 1 cuanto más chico sea el intervalo de los datos de *E* en relación al ancho de *S*.

Si el conjunto *E* es vacío entonces se determina que $Z4_i$ vale 1, ya que al no haber datos participantes en la intersección, esta dimensión es candidata para ser usada en la división de hiper-rectángulos.

Este índice le da mayor importancia a los casos donde el intervalo formado en la intersección de datos es pequeño en relación al intervalo de *S* (Figura 2-20, Tabla 2-4). Hay que notar, que en los casos donde el intervalo de *E* sea pequeño en relación a *S*, el mismo puede estar ubicado en diferentes posiciones dentro de *S* (Figura 2-21, Tabla 2-5). Si bien en todos estos casos el índice $Z4_i$ toma el mismo valor, los hiper-rectángulos que resulten de la división podrían ser ligeramente distintos, aunque igual de válidos para el objetivo final de lograr un modelo de los datos.

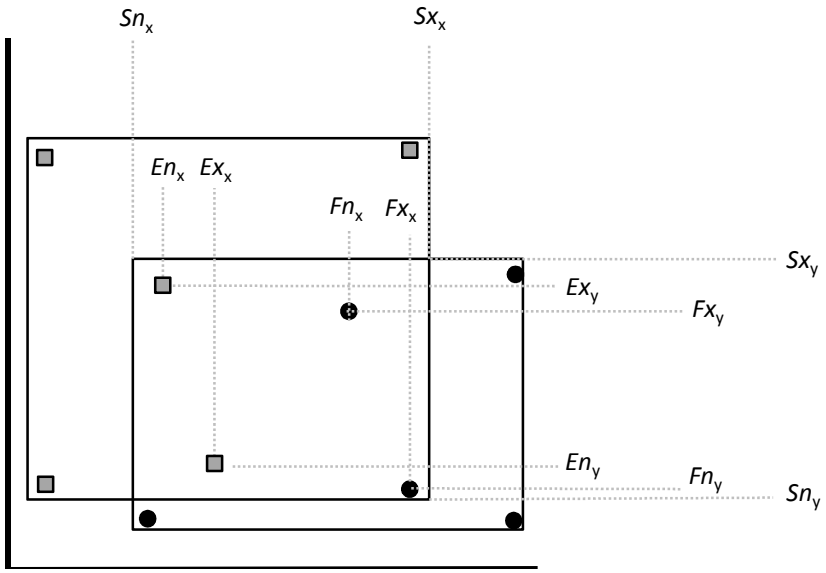

Figura 2-20. Ejemplo que muestra el aspecto que mide el índice $Z4_i$. Se calcula el índice para los dos hiper-rectángulos observando que en la dimensión del eje X el índice tiende a 1 mientras que en la dimensión del eje Y el valor que toma el índice tiende a 0.

3.1.5. $Z5_i$ – Desplazamiento del intervalo del subconjunto de datos intersectados de un hiper-rectángulo en relación al mínimo del intervalo de subconjunto de datos participantes del otro hiper-rectángulo.

Para un hiper-rectángulo *H* presente en una superposición *S* junto a un segundo hiper-rectángulo *J*, dados el subconjunto de datos intersectados *G* de *H* y el subconjunto de datos participante *F* de *J*, $Z5_i(H)$ se define como:

$$Z5_i(H) = 1 - |Gn_i - Fn_i|/(Fx_i - Fn_i)$$

Así, como en el índice anterior se vio que el desplazamiento de un intervalo dentro de otro da el mismo valor, este índice tiene como objetivo todo lo contrario; medir cuan desplazado está el intervalo del subconjunto de datos intersectados en el intervalo subconjunto de datos participantes del otro. Este índice tiende a 0 cuanto más cerca está el valor mínimo de *G* con respecto al valor máximo de *F* y tiende a 1 cuanto más cerca está el valor mínimo de *G* en relación al valor mínimo de *F*.

Si el subconjunto *F* o el subconjunto *G* son vacíos entonces este índice no puede ser calculado y el valor $Z5_i$ toma el valor 1 ya que al no haber superposición esta dimensión podría ser elegida como candidata para dividir los hiper-rectángulos.

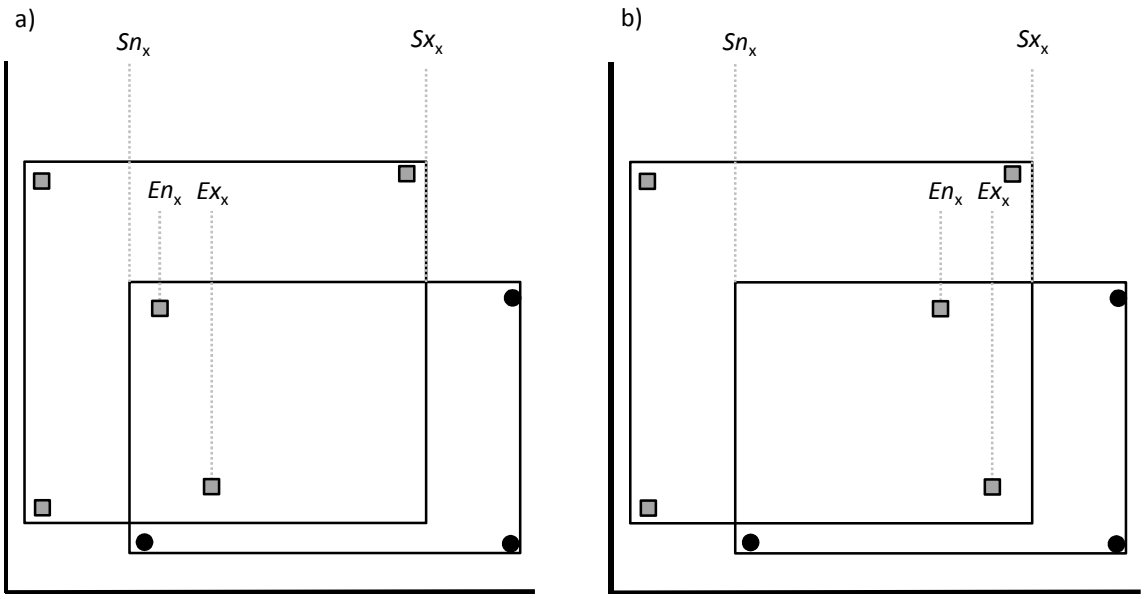

Figura 2-21. Ejemplo donde se ve que el índice $Z4_i$ puede tomar el mismo valor ante casos distintos.

Tabla 2-4. Cálculo de los índices $Z4_x(H)$, $Z4_y(H)$, $Z4_x(J)$, $Z4_y(J)$ del ejemplo de la Figura 2-20.

	X	Y
En	20	14
Ex	27	38
Fn	45	10
Fx	53	34
Sn	16	9
Sx	56	41
Z4(*H*)	0,825	0,25
Z4(*J*)	0,8	0,25

Tabla 2-5. Cálculo de los índices $Z4_x(H)$, para los ejemplos de la Figura 2-21.

	(a)	(b)
En	20	44
Ex	27	51
Sn	16	16
Sx	56	56
Z4(*H*)	0,825	0,825

Este índice da más peso a aquellos casos donde los datos del subconjunto de intersección de un hiper-rectángulo están más cerca del mínimo valor del intervalo de intersección de datos participantes del otro. De esa manera, existen más chances de asegurar una mayor precisión en el modelo de datos si en esta dimensión se lleva a cabo la división de los hiper-rectángulos (Figura 2-22, Tabla 2-6).

3.1.6. $Z6_i$ – Desplazamiento del intervalo del subconjunto de datos intersectados de un hiper-rectángulo en relación al máximo del intervalo de subconjunto de datos participantes del otro hiper-rectángulo.

Para un hiper-rectángulo *H* presente en una superposición *S* junto a un segundo hiper-rectángulo *J*, dados el subconjunto de datos intersectados *G* de *H* y el subconjunto de datos participante *F* de *J*, $Z6_i(H)$ se define como:

$$Z6_i(H) = |Fx_i - Gx_i|/(Fx_i - Fn_i)$$

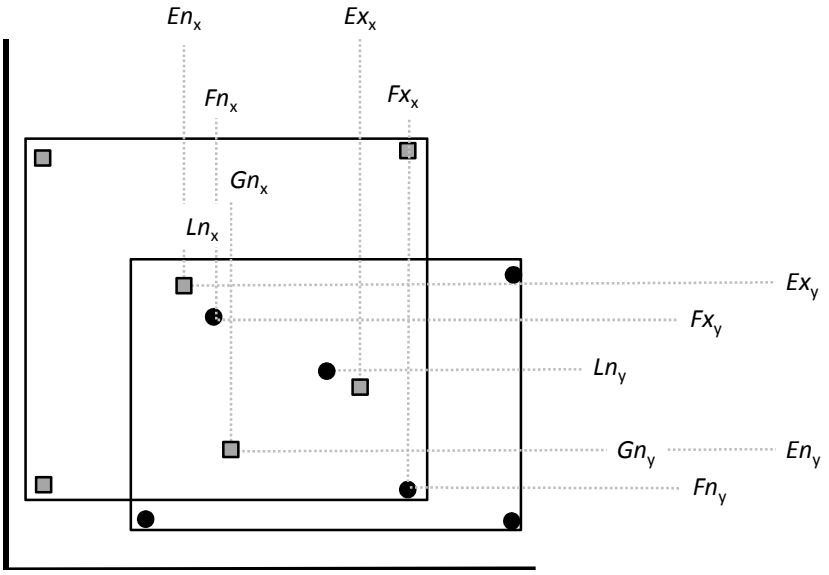

Figura 2-22. Ejemplo que muestra el aspecto que mide el índice $Z5_i$. Se calcula el índice para los dos hiper-rectángulos observando que en la dimensión del eje X el índice tiende a 1 mientras que en la dimensión del eje Y el valor que toma el índice tiende a 0.

Tabla 2-6. Cálculo de los índices $Z5_x(H)$, $Z5_y(H)$, $Z5_x(J)$, $Z5_y(J)$ del ejemplo de la Figura 2-22.

	X	Y
En	23	16
Ex	47	38
Fn	28	11
Fx	53	33
Gn	30	16
Ln	28	26
Z5(*H*)	0,92	0,77
Z5(*J*)	0,79	0,55

Este índice, similar al anterior, tiene como objetivo medir el grado de separación que hay entre el valor máximo del intervalo del subconjunto de datos intersectados *G* en relación al valor máximo del intervalo de datos del subconjunto de participación *F*. Así, tiende a 0 cuanto más cerca está el valor máximo de *G* en relación al valor máximo de *G* y tiende a 1 cuanto más alejado se encuentre.

Al igual que el índice anterior, el valor de $Z6_i$ vale 1 cuando no se puedan calcular ni *F* ni *G* al no haber intersección en los datos, dándole así más peso a la dimensión *i* para ser elegida para una posible separación.

Este índice le da más importancia a los casos donde el valor máximo del subconjunto de datos de intersección *G* está más cerca del valor mínimo del subconjunto de datos intersectado *F*, sugiriendo de esa manera más chances de lograr un modelo de datos más preciso al dividir por la dimensión *i* (Figura 2-23, Tabla 2-7).

3.2. Índice de separabilidad Ω

En la sección anterior, se presentaron seis índices denominados índices Z, donde cada uno de ellos mide una característica distinta en una superposición de dos hiper-rectángulos. Para poder decidir qué acción llevar a cabo y eliminar una superposición dada hace falta un único valor que indique la dimensión en la cual llevar a cabo el ajuste de los hiper-rectángulos. Este único valor es denominado "índice de separabilidad Ω" y para su cálculo se utilizan los índices de superposición Z vistos en la sección anterior.

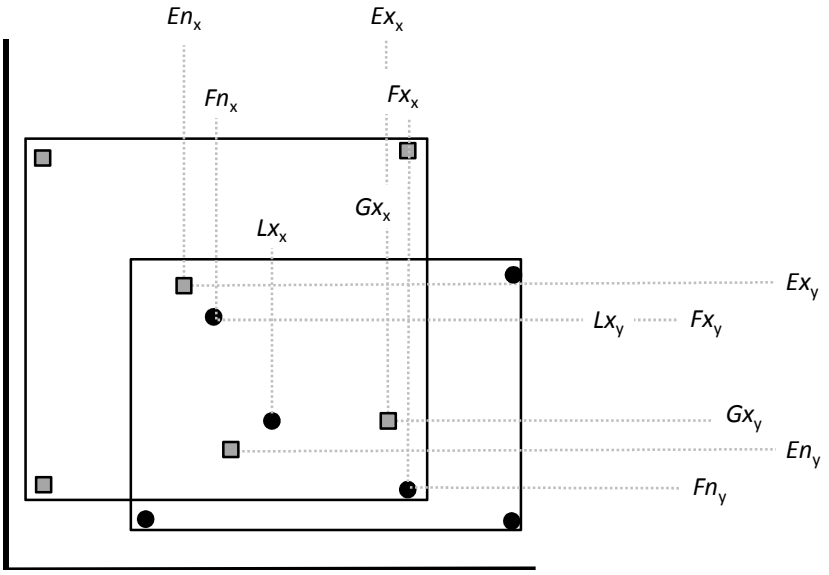

Figura 2-23. Ejemplo que muestra el aspecto que mide el índice $Z6_i$. Se calcula el índice para los dos hiper-rectángulos observando que para la dimensión del eje X el índice tiende a 1 para J y a 0 para H, mientras que en la dimensión del eje Y ocurre todo lo contrario.

Tabla 2-7. Cálculo de los índices $Z6_x(H)$, $Z6_y(H)$, $Z6_x(J)$, $Z6_y(J)$ del ejemplo de la Figura 2-23.

	X	Y
En	24	16
Ex	50	38
Fn	28	11
Fx	53	33
Gx	50	20
Lx	35	33
Z6(*H*)	0,12	0,59
Z6(*J*)	0,58	0,23

En la técnica de clasificación presentada en esta tesis se propone utilizar como índice Ω el promedio de los seis índices Z. Como todos los índices Z dan como resultado un valor en el intervalo [0, 1] donde 1 significa que la dimensión que mide el índice es la más apta para llevar a cabo el ajuste y 0 lo contrario, el índice Ω, al ser calculado como un promedio, también dará un único valor en el intervalo [0, 1] con el mismo significado de que el valor 1 significa que es una dimensión candidata y 0 todo lo contrario.

El índice de separabilidad Ω_i para un hiper-rectángulo *H* basado en *k* índices independientes entre si, se calcula de la siguiente manera:

Ecuación 2-1: $$\Omega_i(H) = \frac{\sum_{j=1}^{k} Zj_i(H)}{k}$$

De esta manera, se calcula un índice Ω_i para cada dimensión *i* el cual es calculado además para cada uno de los dos hiper-rectángulos presentes un una superposición, calculando así pares de índices Ω en todas las superposiciones presentes en el modelo de datos. De todos los índices Ω_i calculados, el índice Ω_i con mayor valor determina el hiper-rectángulo a dividir y en que dimensión hacer la división.

3.2.1. Ponderando por la cantidad de datos participantes

La Figura 2-24 junto con la Tabla 2-8 muestra un ejemplo donde luego de realizar los cálculos de los índices Ω, resulta mayor el correspondiente al hiper-rectángulo *J* en la dimensión *Y*. Por lo que el algoritmo de la técnica

presentada en esta tesis divide a este hiper-rectángulo por algún valor según el criterio elegido. La Figura 2-25 a) muestra un posible resultado de esta división. El problema que se presenta en este ejemplo está dado porque los índices solo miden proporciones entre límites de intervalos sin importar la cantidad de datos que están involucrados en la superposición.

Para resolver este tipo de inconvenientes se propone ponderar cada uno de los índices Z por la cantidad de datos participantes. De esa manera, ante valores similares comienza a tener peso la cantidad de datos involucrados en la superposición. El peso V para cada índice Z se analiza a continuación. Todos los pesos son definidos con ecuaciones que dan como resultado valores en el intervalo [0, 1].

3.2.1.1. $Z1_i$

Este índice no se calcula en base a datos intersectados por lo que el peso $V1_i$ siempre es igual a 1.

3.2.1.2. $Z2_i$

El índice $Z2_i$ mide la relación del ancho del intervalo de datos en relación al ancho del intervalo del subconjunto de datos participante por lo tanto el peso de este índice estará dado por la ecuación:

$$V2_i(H) = \frac{\#G}{\#E}$$

Donde *E* es el subconjunto de datos participante de *H* y *G* su subconjunto de datos intersectados.

3.2.1.3. $Z3_i$

El índice $Z3_i$ mide la relación del ancho del intervalo de datos intersectados en relación al ancho del intervalo de datos participantes por lo tanto el peso de este índice estará dado por la ecuación:

$$V3_i(H) = \frac{\#G}{\#E}$$

Al igual que el anterior, *E* es el subconjunto de datos participante de *H* y *G* su subconjunto de datos intersectados.

3.2.1.4. $Z4_i$

El índice $Z4_i$ mide la relación del ancho del intervalo del conjunto de datos participante en relación al ancho del intervalo de área, por lo tanto el peso de este índice se define por la ecuación:

$$V4_i(H) = \frac{\#E}{\#H}$$

Donde *E* es el subconjunto de datos participante de *H*.

3.2.1.5. $Z5_i$

El índice $Z5_i$ mide el desplazamiento del intervalo del subconjunto de datos intersectados *G* de un hiper-rectángulo *H* en relación al mínimo del intervalo del subconjunto de datos intersectados *L* del otro hiper-rectángulo *J*. Así, es de interés saber cuántos datos de *L* quedan "encerrados" entre el mínimo de *L* y el mínimo de *G* para saber cuán importante es separar por este criterio. El peso para este índice queda definido por la ecuación:

$$V5_i(H) = \frac{\#L'}{\#L}$$

Donde *L* es el subconjunto de datos participante de *J* y *L´* el subconjunto de datos que están entre el valor mínimo de *L* y el valor mínimo del subconjunto de datos intersectados *G* de *H*. *L´* queda definido por la siguiente ecuación:

$$L'_i = \{l \mid l \in L, l \leq Gn_i\}$$

Este peso tiende a 1 cuanto más elementos tenga *L´*, es decir cuantos más elementos estén entre *Ln* y *Gn*, indicando que el desplazamiento de *G* en relación a *L* es importante.

3.2.1.6. $Z6_i$

El índice $Z6_i$ mide el desplazamiento del intervalo del subconjunto de datos intersectados en relación al máximo del intervalo del subconjunto de datos participantes, por lo tanto el peso para este índice queda definido por la ecuación:

$$V6_i(H) = \frac{\#L''}{\#L}$$

Donde *L* es el subconjunto de datos participante de *J* y *L''* el subconjunto de datos que están entre el valor máximo de *L* y el valor máximo del subconjunto de datos intersectados *G* de *H*. *L''* queda definido por la siguiente ecuación:

$$L''_i = \{l \mid l \in L, l \geq Gx_i\}$$

3.2.1.7. Re-definición del cálculo de Ω_i ponderado por los pesos V

Al introducir los pesos V para cada uno de los índices Z, la Ecuación 2-1 es redefinida de la siguiente manera:

$$\Omega_i(H) = \frac{\sum_{j=1}^{k} Zj_i(H)Vj_i(H)}{k}$$

En la Tabla 2-8 es posible ver los valores de los índices Ω ponderados y como dan como resultado la sugerencia de dividir el hiper-rectángulo *H* por la dimensión X. La figura Figura 2-25 b) muestra el resultado de realizar esta acción. Como lo muestra el ejemplo, esta última acción resulta ser mucho más adecuada que la división anterior, logrando así un mejor modelo de datos con más precisión que la opción mostrada en la Figura 2-25 a), la cual incluso, deja varios datos de *H* fuera de su hiper-rectángulo representativo.

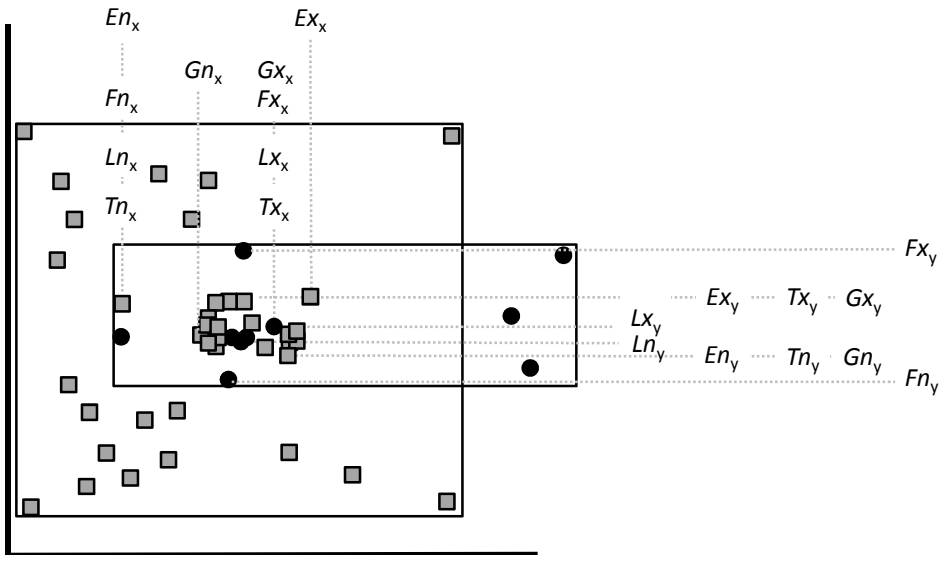

Figura 2-24. Ejemplo que muestra que la diferencia entre los anchos de dos hiper-rectángulos puede ocasionar resultados no deseables (ver Tabla 2-8 y Figura 2-25).

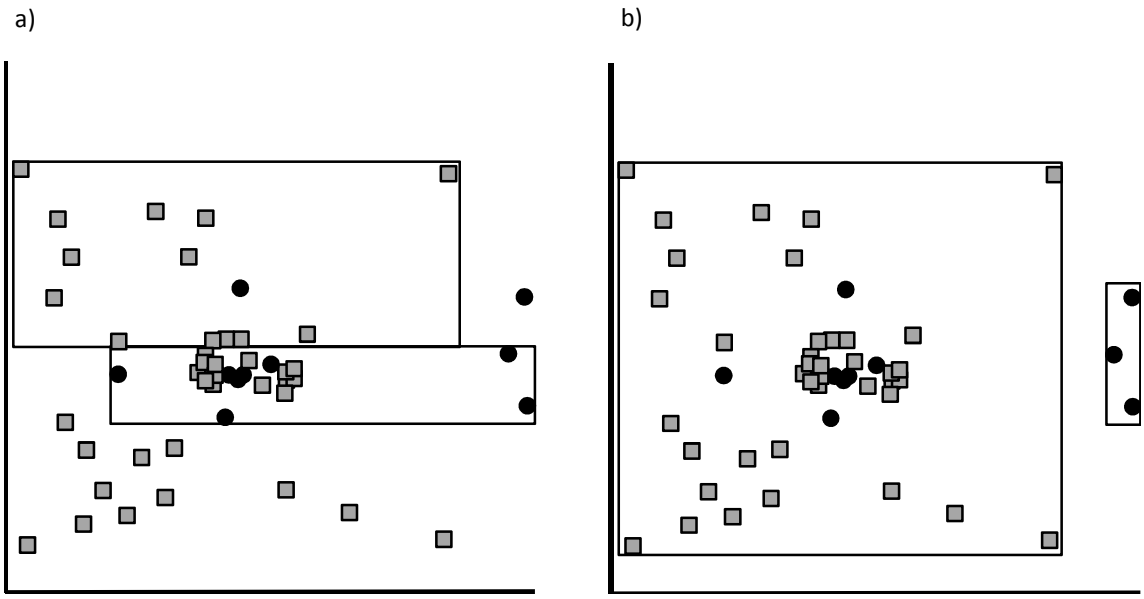

Figura 2-25. a) Posible división del hiper-rectángulo J del ejemplo de la Figura 2-24 por el eje Y. b) División del hiper-rectángulo H del mismo ejemplo por la dimensión X.

3.2.2. Ponderando los índices por otros criterios

Aunque no es objetivo de esta tesis estudiar otros criterios de ponderación, se hace mención a algunos casos que podrían ser interesantes para estudiar y que podrían dar resultados diferentes.

De algunos ensayos, detallados en el capítulo 4, que se realizaron con la técnica de clasificación presentada en esta tesis, se encontraron dudas y preguntas de cómo actuaría la técnica presentada y cuál sería el modelo de datos final si se aplicara alguna modificación a los criterios de decisión de qué intersección eliminar. Por ejemplo, resulta interesante estudiar aquellos casos donde sin importar los valores de todos los índices Z, el índice Z1 sea quien determine la dimensión por la cual dividir los hiper-rectángulos cuando, por ejemplo, ocurre que no hay datos participantes de ninguna clase en la superposición. Así, el índice Ω en vez de calcularse como el promedio ponderado de los índices, simplemente toma el valor de $Z1_i$ cuando ocurren las condiciones detalladas anteriormente.

Como se verá en la siguiente sección, el uso de los índices de superposición en el armado del modelo es completamente flexible y el experto del dominio del problema puede determinar la manera de trabajar para que la estrategia propuesta arme un modelo de los datos de diferentes formas según el problema que se quiera resolver.

3.3. Una estrategia de clasificación flexible

La técnica propuesta en esta tesis, en relación a calcular analíticamente diferentes características de una superposición entre dos hiper-rectángulos, presenta como principal ventaja la de que un experto del dominio del problema, y encargado en armar un modelo de datos, pueda determinar que índices usar en cada una de las decisiones que deba tomar el algoritmo de clasificación

Esta estrategia permite que el experto del dominio del problema utilice todos los índices Z presentados en la sección 3.1 o solo un subconjunto de ellos. Al igual que será capaz de elegir de que manera combinar los índices Z y si estos son ponderados o no y de qué manera calcular el peso para cada uno de ellos. De esta manera, la técnica propuesta en esta tesis se transforma en una poderosa técnica de clasificación que resulta completamente flexible y personalizable.

Tabla 2-8. Esta tabla muestra el resultado numérico del ejemplo de la Figura 2-24. En la parte superior se muestran los valores de los límites de cada uno de los intervalos para ambos hiper-rectángulos en la izquierda y los valores de cardinalidad de los subconjuntos a la derecha. En el medio se muestran los resultados de los índices Z a la izquierda y de los pesos V a la derecha. Finalmente en la parte inferior están los índices Ω calculados sin peso (a la izquierda) y con peso (a la derecha).

	X	Y		X	Y
Hn	1	5	*#H*	40	40
Hx	60	57	*#J*	10	10
Jn	14	22	*#E*	20	20
Jx	70	41	*#F*	7	7
En	15	26	*#G*	14	14
Ex	40	34	*#G'*	1	6
Fn	15	23	*#G''*	1	5
Fx	35	40	*#L*	7	5
Gn	25	26	*#L'*	3	1
Gx	35	34	*#L''*	3	1
Ln	15	28			
Lx	35	30			
Índices Z			Pesos V		
Z1(*H*)	0,22	0,63	V1(*H*)	1	1
Z1(*J*)	0,18	0	V1(*J*)	1	1
Z2(*H*)	0,2	0	V2(*H*)	0,7	0,7
Z2(*J*)	0	0,53	V2(*J*)	0,7	0,5
Z3(*H*)	0,5	0	V3(*H*)	0,7	0,7
Z3(*J*)	0	0,75	V3(*J*)	0,7	0,5
Z4(*H*)	0,46	0,58	V4(*H*)	0,75	0,75
Z4(*J*)	0,57	0,11	V4(*J*)	0,7	0,7
Z5(*H*)	0,5	0,82	V5(*H*)	0,43	0,14
Z5(*J*)	1	0,75	V5(*J*)	0,07	0,43
Z6(*H*)	0	0,35	V6(*H*)	0,43	0,14
Z6(*J*)	0,2	0,5	V6(*J*)	0,07	0,36
Sin pesos			Con pesos		
Ω(*H*)	0,31	0,40	Ω(*H*)	**0,211**	0,206
Ω(*J*)	0,32	**0,44**	Ω(*J*)	0,110	0,202

Como línea de investigación futura es posible pensar en definir otros índices Z que midan otras características de una superposición. También es posible pensar en lograr que dichos índices sean capaces de determinar las características del problema y de esa manera tener la capacidad de sugerir al experto el uso del subconjunto de índices Z que resulte óptimo para lograr un mejor resultado en determinados problemas.

4. CLUHR

En la sección 1 se explicó en detalle cómo pueden ser utilizados como descriptores de datos los hiper-rectángulos en un espacio *D*-dimensional. Estos hiper-rectángulos pueden ser vistos como una forma de generalizar los datos de una clase, además permiten la extracción de reglas de manera directa. En la sección 2 se presentaron los problemas que existen cuando dos o más hiper-rectángulos presentan una superposición en el espacio y como pueden ser resueltas. Finalmente, en la sección 3 se presentaron en detalle una serie de índices de superposición los cuales miden de manera analítica las características que tiene una superposición entre dos hiper-rectángulos. Estos índices son utilizados para decidir que superposición, de todas las presentes, debe ser eliminada o minimizada.

En esta sección presentamos una técnica de minería de datos para llevar a cabo tareas de clasificación la cual utiliza a los hiper-rectángulos como descriptores de los datos y que minimiza el volumen de intersección entre los hiper-rectángulos utilizando los índices de superposición presentados en la sección 3.

La técnica propuesta denominada CLUHR (Clasificación utilizando hiper-rectángulos) permite, una vez que se consigue armar el modelo de datos, extraer como conocimiento reglas de clasificación, las cuales se obtienen de los hiper-rectángulos formados. Es posible encuadrar esta técnica en la categoría de los algoritmos de "divide y vencerás", ya que dicho algoritmo comienza formando un gran hiper-rectángulo para cada clase y luego mediante la división de estos logra armar el modelo de datos. Finalizado el modelo de datos es posible extraer un conjunto de reglas de clasificación, este conjunto, como se verá en la sección 6, representa a un conjunto de reglas por cobertura, es decir, no todo el espacio está cubierto por las reglas, e incluso y dependiendo de como se configure el algoritmo para el armado del modelo, algunas de ellas podrían cubrir una misma parte del espacio de los datos.

Como se mencionó en las secciones anteriores la técnica de clasificación propuesta en esta tesis consiste en detectar superposiciones entre hiper-rectángulos de distintas clases y luego determinar, mediante el cálculo de ciertos índices, cual es la dimensión por la cual llevar a cabo un ajuste de los hiper-rectángulos ya sea para minimizar la superposición o eliminarla de manera completa.

La filosofía del algoritmo de clasificación es simple:

mientras existan superposiciones de hiper-rectángulos, eliminarlas.

El algoritmo tiene una etapa de inicialización en la cual crea, para cada clase de datos presente en la base de datos, su correspondiente hiper-rectángulo representativo mínimo inicial. De esta manera, el algoritmo comienza con tantos hiper-rectángulos como clases haya en la base de datos.

El proceso de clasificación y armado del modelo de datos consiste en un proceso iterativo en el cual se buscan todas las superposiciones existentes entre dos hiper-rectángulos de distintas clases. Luego se procede a realizar el cálculo de los índices Ω para cada uno de los hiper-rectángulos presentes en las superposiciones encontradas y en todas las dimensiones del espacio del problema, para luego buscar aquella superposición S con sus correspondientes hiper-rectángulos y la dimensión i en la cual el índice Ω tenga el valor más alto. En esa superposición S se realiza el ajuste, por la dimensión i, de los hiper-rectángulos involucrados dividiendo al que tiene el valor Ω más alto. Finalmente se ajustan los hiper-rectángulos que fueron modificados y los nuevos hiper-rectángulos que fueran creados a sus correspondientes datos formando así nuevos hiper-rectángulos representativos mínimos.

El algoritmo vuelve a buscar superposiciones entre todos los hiper-rectángulos y se repite el proceso anterior. Este proceso iterativo continúa hasta eliminar todas las superposiciones existentes.

Al finalizar el algoritmo, cada clase tendrá uno o más hiper-rectángulos representativos y cada uno de ellos describe una regla de clasificación para la clase. La extracción y preparación de reglas se ve en detalle en la sección 5.

Como lo ilustra el ejemplo de la Figura 2-26, existen casos de superposición donde la tarea de dividir un hiper-rectángulo solo logra "aislar" unos pocos datos de una clase. El necesitar "aislar" estos datos para obtener una regla que los describa depende exclusivamente del problema que se intenta resolver y las necesidades del usuario del modelo. Nótese que incluso puede resultar en un hiper-rectángulo representativo de un único dato, en cuyo caso se obtiene una regla de clasificación para un único dato.

Para lidiar con este problema, el algoritmo posee un parámetro μ cuya tarea es la de determinar que superposiciones se analizan cuando la cantidad de datos involucrados de una o ambas clases es mayor que el valor especificado en μ.

Con el uso de este parámetro, el algoritmo solo elimina aquellas superposiciones que tienen una cantidad mínima de datos participantes dejando sin modificar las que el experto o el usuario del modelo consideran insignificantes. Obviamente, el no eliminar estas superposiciones con pocos datos produce como resultado un modelo de datos menos preciso, por lo que el uso de este parámetro resulta clave para encontrar un equilibrio entre obtener un modelo de datos preciso versus una cantidad mínima de reglas de clasificación.

Según el problema que se quiera resolver y dependiendo de las necesidades del usuario del modelo de datos, el valor μ establecido se puede interpretar de diferentes maneras, aunque la esencia de esa interpretación es siempre la misma; el número mínimo de datos presentes para eliminar una superposición.

Las distintas interpretaciones que pueden tenerse en cuenta para usar este parámetro son las siguientes:

- μ debe ser menor que la suma de los datos de dos hiper-rectángulos presentes en una superposición. Por lo general, este criterio debe ser utilizado en aquellos casos donde la importancia de todas las clases involucradas es la misma. Este criterio puede producir un modelo de datos con una buena precisión pero un número grande de reglas de clasificación.

- μ debe ser menor que la cantidad de datos del hiper-rectángulo que presente menos datos en la superposición. Este criterio puede ser utilizado para lograr un modelo de datos con un número reducido de reglas.

- μ debe ser menor que la cantidad de datos del hiper-rectángulo si este representa a una clase en particular. Este criterio, al igual que el anterior, logra un modelo con un número reducido de reglas y debe ser utilizado en aquellos problemas donde lograr una buena precisión para una clase en particular es mucho más importante que hacerlo para la otra clase. Por ejemplo, el caso de detección de operaciones bancarias fraudulentas, donde se busca un modelo de datos que represente de manera más precisa las operaciones fraudulentas que las que no lo son.

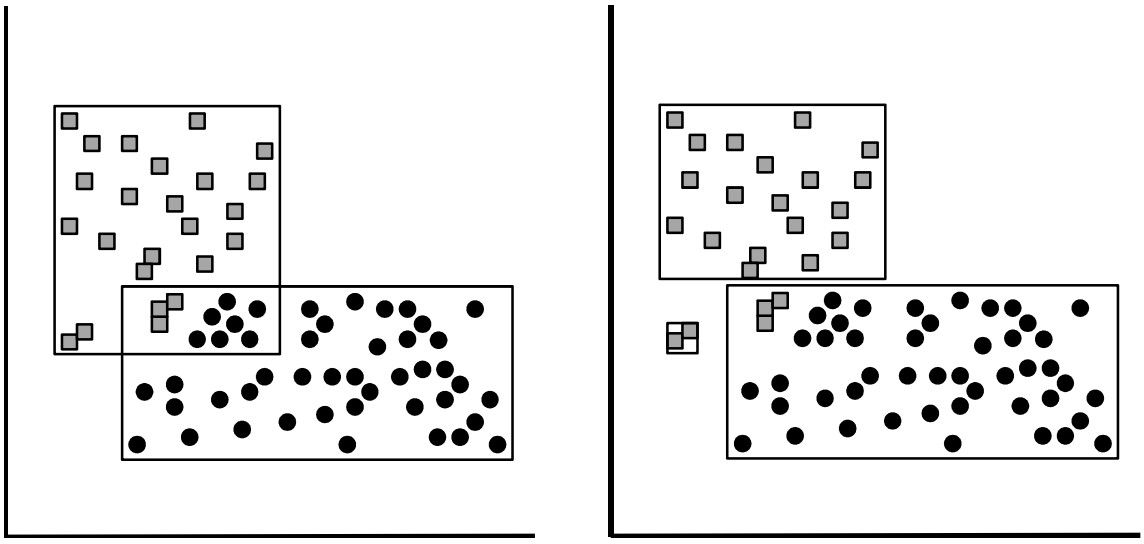

Figura 2-26. Ejemplo que muestra como la división de un hiper-rectángulo logra "aislar" a solo dos datos de la clase en su propio hiper-rectángulo.

El pseudocódigo del algoritmo de CLUHR es el siguiente:

```
Inicialización

Buscar superposiciones iniciales

mientras existan superposiciones a eliminar hacer

        Calcular índices Ω

        Dividir o ajustar los hiper-rectángulos correspondientes

        Buscar nuevas superposiciones

fin mientras

Finalización del algoritmo

Extraer reglas de clasificación
```

En las siguientes secciones se ve en detalle cada uno de los pasos del algoritmo.

4.1. Inicialización del algoritmo

La inicialización del algoritmo consiste en armar, para cada clase de datos presente en la base de datos con la que se desea trabajar, su correspondiente hiper-rectángulo representativo mínimo (sección 1.1).

De esa manera, se crea un conjunto de hiper-rectángulos *CH* formado por los hiper-rectángulos representativos mínimos de cada clase. Así, el algoritmo comienza con tantos hiper-rectángulos como clases haya en la base de datos.

Se establece el valor para el parámetro μ y con qué criterio de los comentados anteriormente será utilizado.

4.1.1. Detectar superposiciones iniciales

Se revisan todos los pares de hiper-rectángulos de *CH* detectando aquellos que presenten una superposición de hiper-volumen (ver sección 2).

Así se crea el conjunto *CS* formado por todas las superposiciones detectadas. Este conjunto estará formado por ternas (*S*, *H*, *J*) donde *H* y *J* son los dos hiper-rectángulos de clases distintas que presentan una superposición *S*.

Serán excluidos de *CS* aquellas superposiciones donde los datos participantes sea menor que el indicado por el parámetro μ.

4.2. Eliminar todas las superposiciones

Esta sección describe el funcionamiento de la parte iterativa del algoritmo y es ejecutada mientras *CS* no sea el conjunto vacío.

4.2.1. Calcular los índices Ω

Se arma una matriz con tantas filas como elementos tenga *CS* y tantas columnas como dimensiones tenga el problema a tratar. Cada celda de esta matriz tendrá una dupla de valores correspondiente a los valores calculados para los índices Ω de los dos hiper-rectángulos presentes en la superposición dada por la fila de la matriz, en la dimensión establecida por la columna de la matriz.

Luego se busca en toda la matriz cual es el índice Ω más grande y la fila y la columna donde fue hallado ese valor determina que superposición modificar y en que dimensión realizar los ajustes de los hiper-rectángulos

intervinientes en dicha superposición. De la dupla de índices Ω encontrada en dicha celda, el índice mayor, por lo general sugiere cual de los dos hiper-rectángulos dividir. Aunque la división dependerá del tipo de superposición, como se vio en la sección 2.1, y del tipo de problema a resolver.

En el caso que se encuentre el mismo valor de Ω en varios elementos de la matriz, en esta tesis no se propone ningún método para elegir un caso en particular, por lo tanto, la elección final de la superposición y dimensión se hace de manera arbitraria. En este sentido, resulta igualmente válido utilizar cualquier otro criterio incluyendo aquel que determina si una clase tiene más importancia que otra.

4.2.2. Realizar el ajuste

Una vez establecida la terna (*S*, *H*, *J*) a dividir y la dimensión *i* en la cual hacer la división se procede a determinar el tipo de superposición *S* según lo visto en la sección 2.1. Del par de valores de índice Ω, el mayor por lo general sugiere que ese hiper-rectángulo sea el que debe dividirse, pero la división o ajuste propiamente dicho depende exclusivamente del problema. Así, la división o disminución de *H* o *J* depende de si en la superposición *S* hay datos participantes o no, y si los hay, si pertenecen a ambos hiper-rectángulos o no (ver sección 2.2).

4.2.2.1. Método alternativo para la división de hiper-rectángulos cuando hay datos de ambas clases en la superposición

Como se mencionó en la sección 2.2.3 la eliminación de superposiciones cuando hay presentes datos de ambas clases puede hacerse en favor de alguna de las dos clases o bien buscar un punto intermedio de corte. La decisión de cortar en favor de una de las dos clases depende del problema que se quiere resolver y las necesidades que tenga el usuario del modelo.

Si se eligen cortar los hiper-rectángulos en un punto intermedio la elección de este punto puede ser totalmente arbitraria. En esta sección se propone la búsqueda de un punto de corte que resulte lo más equitativa posible para ambos hiper-rectángulos.

El punto de corte estará dado en una dimensión *i* dentro del intervalo de intersección *T* y se determinan analizando los subconjuntos de datos intersectados *G* y *L* (ver sección 3). Para un dato *pc* perteneciente a *G* se determina cuantos datos de *G* son mayores que *pc* y cuantos datos de *L* son menores que *pc*. De esta forma, se intenta realizar la contabilidad de cuantos datos quedan "mal clasificados" si se usara a *pc* como punto de corte (Figura 2-27).

> **Definición 12**: Sea *pc* un dato perteneciente al subconjunto de datos intersectados *G* y sea *L* el otro subconjunto de datos intersectados participante en la superposición. Definimos a *GM* y a *GL* como los subconjuntos de *G* y *L* respectivamente formado por los datos que quedan "mal clasificados" usando a *pc* como punto de corte.

$$GM = \{g \mid g \in G, g > v\}$$

$$GL = \{l \mid l \in L, l < v\}$$

Como la cantidad de datos mal clasificados depende de cuantos datos existen en la superposición, el índice de pérdida de precisión φ es calculado como la proporción de datos que queda mal clasificados, utilizando la siguiente ecuación:

$$\varphi = \frac{\#GM}{\#G} + \frac{\#GL}{\#L}$$

Para cada dato de G y de L se calcula su correspondiente índice φ y aquel que se obtenga el menor valor se utilizará como punto de corte, asegurando así la menor pérdida de precisión.

El pseudocódigo del procedimiento que lleva a cabo la búsqueda del punto de corte óptimo es el siguiente:

```
G = subconjunto de datos intersectados de un hiper-rectángulo H

L = subconjunto de datos intersectados de un hiper-rectángulo J

para todos los datos pc en (G+L) hacer

        GM =subconjunto de datos mal clasificados de G usando a pc como
            punto de corte

        GL =subconjunto de datos mal clasificados de L usando a pc como
            punto de corte

        fi =(#GM / #G) + (#GL / # L)

        si fi < fiMin entonces

                fiMin = fi

                puntoCorte = pc

        fin si

fin para
```

4.2.3. Actualizar los hiper-rectángulos representativos mínimos

Una vez que se realizó el ajuste se procede a determinar para el hiper-rectángulo modificado (o ambos si fuera el caso) cuales son los datos que caen en los límites del nuevo hiper-rectángulo formado.

Como los nuevos hiper-rectángulos formados H_1 y H_2 caen dentro de la misma zona del espacio que el hiper-rectángulo *H* que fue dividido, solo es necesario recorrer los datos representados por *H* y asignárselos a H_1 o a H_2 según corresponda, todos los datos estarán incluidos en uno u otro hiper-rectángulo. Así, cada nuevo hiper-rectángulo será representante de un subconjunto disjunto de los datos representados originalmente por *H*.

Luego de tener, para cada nuevo hiper-rectángulo su conjunto de datos representados, se procede a encontrar para cada conjunto de datos, su correspondiente hiper-rectángulo representativo mínimo. Esta acción es la que logra eliminar la superposición entre los dos hiper-rectángulos involucrados en una superposición (Figura 2-28).

4.2.4. Detectar las nuevas superposiciones

Finalizado el ajuste deben actualizarse los conjuntos *CS* y *CH* de la siguiente manera:

- Se elimina de *CS* la terna (*S*, *H*, *J*), ya que la superposición *S* fue eliminada en el paso previo.
- Si hubo una división se elimina de *CH* el hiper-rectángulo *H* que fue dividido. Si hubo un ajuste no se hace nada.
- Si hubo una división se agrega a *CH* los nuevos hiper-rectángulos que resultaron de la división de *H*. Si hubo un ajuste no se hace nada.
- Se eliminan de *CS* aquellas ternas donde está involucrado el hiper-rectángulo que fue modificado, ya sea por división o por reducción de volumen.

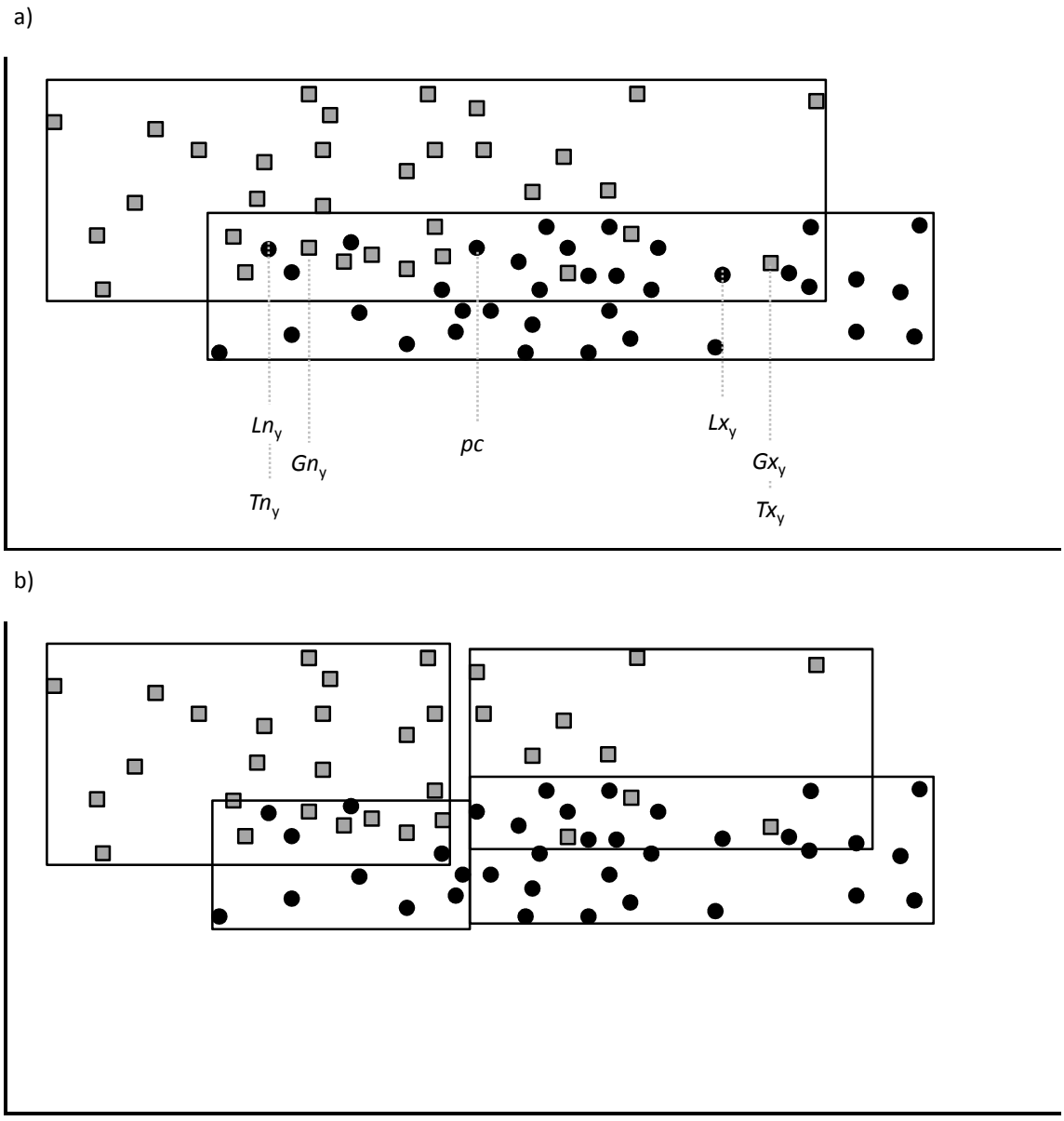

Figura 2-27. En a) se muestran como están formados los subconjuntos T, G y L en una superposición y un dato del eje X señalado como pc para dividir los hiper-rectángulos. En b) se muestran como quedan divididos los hiper-rectángulos donde se observan que cuatro datos de la clase Círculo y tres datos de la clase Cuadrado quedan "mal clasificados".

Finalmente se procede a detectar las nuevas superposiciones entre todos los hiper-rectángulos. Hay que notar, que en realidad no es necesario revisar todos los pares de hiper-rectángulos presentes en *CH*. La división o reducción de un hiper-rectángulo *H* solo afecta a aquellas superposiciones donde estaba involucrado *H*, entonces para que este paso resulte eficiente solo es necesario volver a revisar las superposiciones donde estaba presente *H*.

De esta manera, si *H* sufrió una reducción de volumen solo es necesario revisar las ternas (*S*, *H*, *J*) donde está presente *H* y analizar si el nuevo *H* presenta una superposición con *J*. En caso de que exista una división de *H* entonces todas las ternas donde *H* estaba presente se eliminan y se analizan los nuevos hiper-rectángulos creados buscando superposiciones solo con *J*.

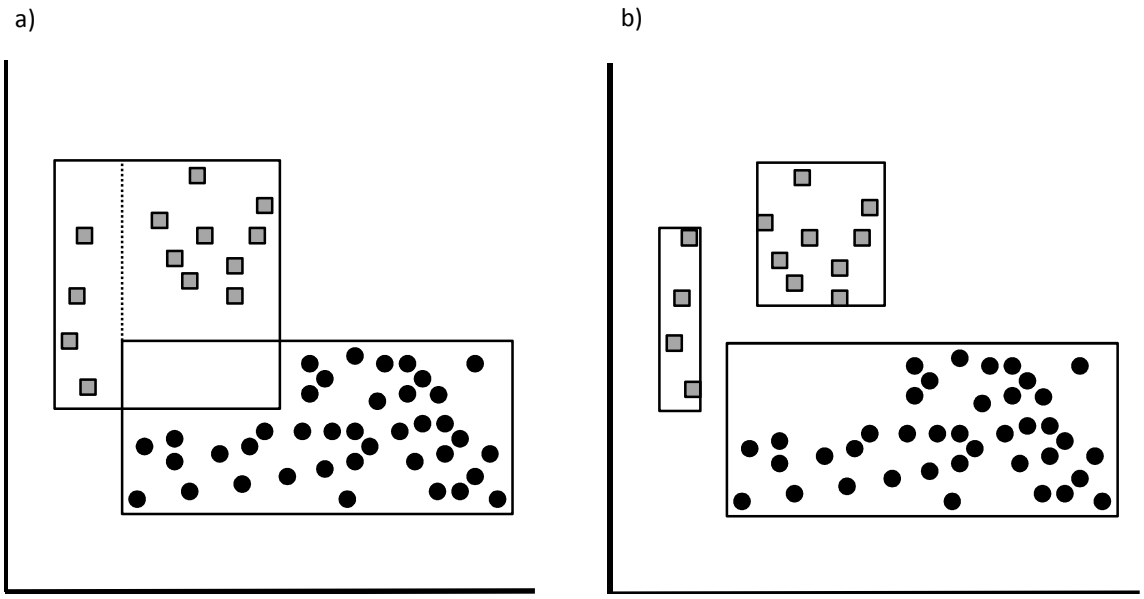

Figura 2-28. a) muestra un caso de superposición y con una línea punteada por donde se llevará a cabo la división. En b) se muestran los dos hiper-rectángulos ajustados a sus respectivos datos.

4.3. Finalizar con el armado del modelo de datos

Finalizado el algoritmo se obtiene para cada clase de la base de datos uno o más hiper-rectángulos representativos. En este punto, si μ se inicializó con el valor cero, entonces no quedan superposiciones de ninguna clase en el conjunto *CS* y de cada uno de los hiper-rectángulos formados se puede extraer una regla que resulta descriptiva para los datos que están incluidos en tal hiper-rectángulo, y esta descripción resulta en un modelo que asegura una precisión del 100%.

Por otra parte, si μ fue inicializado con un valor mayor que cero entonces finalizado el proceso iterativo es probable que existan superposiciones que no fueron tratadas justamente por no superar el número mínimo de datos establecido por el parámetro μ. Si de los hiper-rectángulos formados se extraen reglas entonces en algunos casos habrá pares de reglas no-disjuntas. Para resolver este problema el usuario del modelo tiene dos posibles alternativas.

- Permitir que existan pares de reglas no-disjuntas y darle un orden de prioridad a las reglas para que el modelo de datos al momento de predecir futuras muestras, lo haga sin contradicciones.

- Eliminar las superposiciones de hiper-rectángulos para lograr un modelo más preciso pero con el agregado de nuevas reglas, las cuales en su mayoría describirán pocos datos.

La elección de una u otra alternativa dependerá del problema a tratar y del tipo de modelo de datos que desea obtener el usuario final. Finalmente, se extrae una regla por cada hiper-rectángulo formado de la manera que se detalla en la sección 5.

4.4. Estructura del modelo de datos

Una vez que finalice la ejecución del armado del modelo de datos este estará compuesto por los siguientes elementos:

- Un conjunto de reglas de clasificación.

- Un conjunto de hiper-rectángulos, donde cada uno de ellos representa un subconjunto de datos de una determinada clase. Cada hiper-rectángulo almacena el subconjunto de datos que están incluidos en el.

El motivo de que cada hiper-rectángulo almacene el subconjunto de datos a los cuales representa, es debido a que será necesario revisarlo al momento de adaptar el modelo a cambios en los datos como se detalla en el capítulo 3.

4.5. Datos faltantes

Es común encontrar problemas donde en la base de datos a tratar existen datos (filas) que presentan la particularidad de no poseer, por la razón que sea, un valor en un cierto atributo. En este sentido, CLUHR es capaz de manejar sin mayores inconvenientes datos cuyos atributos posean valores desconocidos.

Al momento de armar un hiper-rectángulo representativo mínimo de un subconjunto de datos, ya sea el hiper-rectángulo representativo mínimo inicial de una clase, o aquel que resulte de una división de hiper-rectángulos, se debe hallar el valor mínimo y el valor máximo para cada dimensión del espacio. Si en cierta dimensión, algunos datos no poseen valores, entonces el valor mínimo y el valor máximo son calculados con los valores del resto de los datos, ignorando aquellos que no lo poseen.

Un dato, con un valor desconocido en un cierto atributo, dentro de un hiper-rectángulo está representado por una recta, ya que para ese atributo el valor del dato podría ser cualquiera, dentro de los límites del hiper-rectángulo para esa dimensión.

Cuando se calculan los índices de superposición, estos datos no intervienen en el cálculo del intervalo de datos, ni en el cálculo de los índices Z5 y Z6.

El problema real surge cuando se debe dividir un hiper-rectángulo por una cierta dimensión, y en esa dimensión existen datos que no poseen valores en el atributo correspondiente. Para resolver este problema, es posible aplicar dos soluciones distintas. La implementación de una de estas formas de lidiar con el problema, a priori, no garantiza armar un mejor modelo que si se utilizara la otra solución.

La primear opción consiste en, de manera totalmente arbitraria, asignar los datos que tienen valor faltante en esa dimensión a uno de los dos nuevos hiper-rectángulos. Diferentes criterios para asignar los datos, entre muchos, son: asignar los datos de manera que la cantidad de datos de ambos hiper-rectángulos sean lo más parecida posible, asignar los datos al hiper-rectángulo con más (o con menos) cantidad de datos, asignar los datos al subconjunto que más cerca está (obviamente, la distancia se mide utilizando los atributos que si poseen datos, y esta medida de distancia puede ser cualquiera).

La segunda opción consiste en replicar la muestra para que siga presente en ambos hiper-rectángulos. La decisión de replicar la muestra es que, en principio la información que posea ese dato en otras dimensiones puede resultar útil para futuras divisiones. Y el hecho de asignar de manera arbitraria el dato a un hiper-rectángulo, sólo por no conocer el valor en la dimensión por la cual se está realizando la división, podría resultar en pérdida de información para el hiper-rectángulo que no se queda con dicho dato. La desventaja de esta solución es la que se está duplicando información en la base de datos, aunque el hiper-rectángulo guarde el índice al dato, se estaría multiplicando este índice en todos los hiper-rectángulos que resulten de sucesivas divisiones. En una base de datos con pocos datos faltantes, este no es un gran problema, pero en bases de datos con muchos atributos con valores faltantes, e incluso si un mismo dato posee valores faltantes en más de un atributo, entonces esta solución no podría resultar óptima.

4.6. Una metodología determinista

Como se puede observar, la metodología propuesta en esta tesis es completamente determinista y el resultado solo depende de la configuración elegida por el usuario al momento de armar el modelo. A diferencia de otras estrategias de clasificación (Freitas, 2002) (García, y otros, 2009) (Hasperué, y otros, 2006) (Holden, y otros, 2008), el resultado de esta metodología no depende ni del azar ni del orden en que los datos de entrada son ingresados al algoritmo durante su ejecución.

Esta es una de las principales ventajas de esta metodología ya que una misma entrada, con la misma configuración del algoritmo, produce siempre la misma salida.

4.7. Limitaciones de CLUHR

Como se mencionó anteriormente, CLUHR es una técnica que es capaz de trabajar con bases de datos cuyos atributos están definidos en el espacio numérico, ya sea en el dominio de los enteros o de los reales. CLUHR trabaja "cómodamente" con atributos cuya cardinalidad es alta, por lo que se debe prestar especial atención cuando esta es pequeña.

El cálculo de los índices de superposición está diseñado para poder determinar la dimensión a "cortar" los hiper-rectángulos presentes en una superposición cuando la cardinalidad de los datos en esa dimensión es alta. Y aunque es completamente factible trabajar con atributos cuya cardinalidad es baja, incluso cuando solo hay dos posibles valores como sucede con los atributos binarios, hay que tener en cuenta que los índices de superposición podrían no estar eligiendo la dimensión más adecuada para llevar a cabo el corte de los hiper-rectángulos.

El siguiente ejemplo muestra de que manera el cálculo del índice Ω en una dimensión representada por un atributo binario puede traer dificultades.

Para dicho atributo, al ser binario, existe dos posibles valores que pueden tomar los datos, y es de esperar que todas las clases presentes en la base de datos posean datos que tomen esos dos valores.

El problema se presenta al calcular los índices Z. En el problema mencionado los límites de los hiper-rectángulos H_n y H_x valen lo mismo para ambos hiper-rectángulos (0 y 1), por lo tanto la superposición es igual que los límites de éstos hiper-rectángulos. El intervalo de la superposición T también resulta ser igual al ancho de los hiper-rectángulos y lo mismo sucede con el intervalo de la superposición de datos G. Ante esta situación todos los índices arrojan un valor de 0, excepto el índice Z6 que devuelve 1.

Los pesos para los primeros cinco índices no aportan al cálculo del índice Ω, ya que el valor del propio índice Z vale cero. El único que si participa en el cálculo del índice Ω es el índice Z6 y su correspondiente peso. Hay que tener en cuenta que al tomar el valor uno el índice Z6 el valor final del índice Ω será el valor que tome el peso para este índice. Este peso, detallado en la sección 3.2.1, mide la proporción que hay entre el conjunto de datos participante (en atributos binarios son todos los datos del hiper-rectángulo) y la cantidad de valores que son mayores o iguales que el máximo valor del otro hiper-rectángulo. El máximo valor del otro hiper-rectángulo es 1, por lo tanto el peso estará dado por la cantidad de datos con valor 1, cuanto más cerca este este valor del número total de datos del hiper-rectángulo, o dicho de otra manera, cuantos menos ceros tengan los datos de este hiper-rectángulo, más alto será el peso. Por lo tanto, una dimensión que representa a un atributo binario es muy factible que devuelve un valor alto para el índice Ω.

En una base de datos donde todos los atributos son binarios, es probable que CLUHR no logre buenos resultados, ya que la divisiones estarán dadas por aquellos datos donde más "unos" haya. Por otro lado, si la base de datos tiene una mezcla de atributos binarios y no binarios hay que tener en cuenta que es muy probable que el índice Ω de los atributos binarios arrojen los valores más alto (por lo visto en el ejemplo anterior), por lo que CLUHR tendrá preferencia de dividir siempre por este tipo de atributos y, en este tipo de bases de datos, tal vez no sea la mejor elección para resolver el problema.

A priori no es posible determinar si es bueno o malo dividir primero por este tipo de atributos, pero si es importante tener este problema presenta al momento de usar CLUHR en bases de datos con estas características.

Este problema, si bien menor, se presenta también en bases de datos cuyos atributos son ternarios o cuaternarios (tres o cuatro posibles valores para los atributos). Si bien no se hizo un análisis detallado del problema es posible asegurar por la práctica adquirida en el desarrollo de esta técnica, que un número mínimo de posibles valores para un atributo debería ser mayor que seis, de esta manera el cálculo del índice Ω en base a los índices de superposición propuestos en esta tesis devuelven valores "coherentes" para la elección de la dimensión por la cual dividir los hiper-rectángulos.

Una forma de resolver este problema es, o bien introducir un índice que contemple este tipo de atributos, o bien cambiar la forma de ponderar los índices Z. Estos aspectos se verán con más detalle en el capítulo 5.

5. Extracción de las reglas

Cuando el algoritmo de clasificación finaliza su tarea, se obtiene como resultado un conjunto de hiper-rectángulos donde cada uno de ellos representa a un subconjunto de datos de una clase específica. Estos hiper-rectángulos presentan la particularidad de que todas sus caras son paralelas a algún eje. Por lo tanto y por lo visto en la sección 1 cada uno de estos hiper-rectángulos puede ofrecer al usuario una regla de clasificación del tipo IF-THEN.

De un hiper-rectángulo *H* el cual representa a una clase *C* se extrae una regla de clasificación que tendrá la siguiente forma:

$$IF\ (cl_1\ AND\ cl_2\ AND\ \dots\ AND\ cl_D)\ THEN\ C$$

Así la regla tendrá tantas cláusulas como dimensiones tenga el problema. Y cada cláusula será de la forma:

$$Hn_i \leq x_i \leq Hx_i$$

que resulta equivalente a la siguiente forma de expresión:

$$(x_i \geq Hn_i)\ AND\ (x_i \leq Hx_i)$$

Por lo tanto un hiper-rectángulo puede ser expresado en forma de regla IF-THEN de la siguiente manera:

$$IF\ ((x_1 \geq Hn_1)\ AND\ (x_1 \leq Hx_1)\ AND\ \dots\ AND\ (x_D \geq Hn_D)\ AND\ (x_D \leq Hx_D))\ THEN\ C$$

Esta regla estará formada por $2*D$ cláusulas y como es fácil imaginar la utilización de este tipo de reglas puede resultar muy engorroso para el usuario, en especial si la dimensión del problema a tratar es alta. Por lo tanto es necesario un proceso post-clasificación que simplifique al máximo la complejidad de estas estas reglas de clasificación. Este proceso consiste en eliminar las cláusulas que no son necesarias para la descripción del problema. Así, se denomina conjunto de reglas extendidas a las reglas formadas por las $2*D$ cláusulas, y conjunto de reglas simplificadas a aquellas que tienen el conjunto mínimo de cláusulas necesarias para describir el conjunto de datos del propio hiper-rectángulo sin que se presente contradicción o ambigüedad con otra regla de otro hiper-rectángulo de otra clase.

Considerando el ejemplo de la Figura 2-26, el correspondiente conjunto de reglas de clasificación extendidas para los tres hiper-rectángulos es el siguiente:

1. IF (x ≥ 7) AND (x ≤ 12) AND (y ≥ 24) AND (y ≤ 48) THEN Cuadrado

2. IF (x ≥ 20) AND (x ≤ 37) AND (y ≥ 38) AND (y ≤ 47) THEN Cuadrado

3. IF (x ≥ 16) AND (x ≤ 67) AND (y ≥ 10) AND (y ≤ 33) THEN Círculo

Por simple inspección visual es posible simplificar las reglas de cada hiper-rectángulo de la siguiente manera:

1. IF (x ≤ 12) THEN Cuadrado

2. IF (y ≥ 38) THEN Cuadrado

3. IF (x ≥ 16) AND (y ≤ 33) THEN Círculo

Estas reglas simplificadas describen a cada hiper-rectángulo de manera precisa, ya que no existe un dato de ninguna de las dos clases que satisfagan alguna regla de la clase opuesta. Aun así, las dos reglas de la clase *Cuadrado* pueden ser unidas en una única regla mediante el operador lógico OR, dejando solo dos reglas para la descripción del modelo de datos.

1. IF (x ≤ 12) OR (y ≥ 38) THEN Cuadrado

2. IF (x ≥ 16) AND (y ≤ 33) THEN Círculo

Aunque no es el objetivo de esta tesis lograr un método eficiente de simplificación de reglas presentamos a continuación un método greedy de simplificación de reglas. En la literatura es posible encontrar varios trabajos que realizan esta tarea de una manera más eficiente (Darrah, y otros, 2004) (Ma, y otros, 2009) (Shehzad, 2011).

5.1. Método greedy

El criterio utilizado para la eliminación de cláusulas es simple: eliminar aquellas cláusulas tal que, si se elimina, la regla formada no satisface a ningún dato de otra clase.

Así, el algoritmo de simplificación de una regla consiste en tomar una regla, eliminar una cláusula y ver si la nueva regla formada satisface datos de otra clase. Si no ocurre esa condición entonces la cláusula puede ser eliminada definitivamente de la regla.

Dada una regla *R* a ser simplificada el pseudocódigo del algoritmo de simplificación es el siguiente:

```
para i=1 hasta 2*D hacer
        R' = R sin la cláusula i
        c = número de datos de otra clase que satisfacen R'
        si c = 0 entonces
                R= R'
        fin si
fin para
```

Al finalizar este algoritmo la regla R tendrá el número mínimo de cláusulas necesarias para describir solo datos de su propia clase.

Como puede verse en el ejemplo de la figura Figura 2-26 si al querer simplificar la regla 1) se elimina la cláusula (x≤12), a la regla resultante la satisfacen muchos de los datos de la clase de círculos. Por lo tanto, esa cláusula no puede ser eliminada de la regla. De las tres cláusulas restantes, al ser eliminadas, la regla resultante no satisface ningún dato de la clase círculos y por eso fueron eliminadas una a una dejando como resultado una regla con una única cláusula.

6. Uso del modelo. Predicción

Una vez que el usuario obtiene un modelo de datos y un conjunto de reglas de clasificación es posible utilizarlo para que realice predicciones de clases ante la aparición de nuevos datos.

Por un lado, el conjunto de reglas de clasificación extraídas le da al usuario una idea general de cuáles son las características de cada una de las clases intervinientes en su problema (extracción de conocimiento). Este

conjunto de reglas por lo general son suficientes para clasificar futuras muestras y así predecir a que clase pertenecen nuevos datos.

Por lo tanto, ante la aparición de una nueva muestra y la necesidad de clasificarla, se busca aquella regla que es satisfecha por dicha muestra y la clase de tal regla será la clase de la muestra.

Por otro lado, puede ocurrir que una muestra no cumpla ninguna de las reglas extraídas del modelo de datos. La Figura 2-29 a) muestra el espacio que satisface cada una de las dos reglas simplificadas del ejemplo de la Figura 2-28. Cualquier nuevo dato que caiga en tales zonas es satisfecho por una o por otra regla. Pero como lo muestra la Figura 2-29 b) hay zonas que no son cubiertas por ninguna de las dos reglas y por lo tanto las muestras que caigan en esas zonas no podrán ser clasificadas por las reglas. En estos casos, es necesario tener otro mecanismo de predicción.

Una alternativa que puede ser utilizada para determinar la clase de una muestra, cuando esta no satisface ninguna de las reglas extraídas del modelo de datos, es la de recurrir al propio modelo de datos y su estructura interna. Si bien el usuario por lo general se manejará con el conjunto de reglas extraídas, los hiper-rectángulos de los cuales se extrajo el conjunto de reglas siguen formando parte del modelo de datos.

De esta forma, la manera más directa de determinar a qué clase pertenece una muestra que no cumple ninguna regla es la de encontrar el hiper-rectángulo más cercano a dicha muestra. Este método no solo es propuesto por ser el más coherente con el problema sino que además también es utilizado en otros trabajos (García, y otros, 2009).

La propuesta en esta tesis para encontrar el hiper-rectángulo más cercano a una muestra consiste en medir la distancia euclídea entre la muestra y un hiper-rectángulo. La distancia se realiza a la cara o la arista del hiper-rectángulo que se encuentra más cerca de la muestra. La siguiente ecuación calcula la distancia entre un punto y la cara o arista de un hiper-rectángulo más cercana a la muestra.

Ecuación 2-2:
$$Dist(m,H) = \sqrt[2]{\sum_{i=1}^{D} dif_i}$$

donde

$$dif_i = \begin{cases} (m_i - h_{ni})^2 & if\ m_i < h_{ni} \\ (m_i - h_{xi})^2 & if\ m_i > x \\ 0 & en\ otro\ caso \end{cases}$$

Notar que al usar esta ecuación, si la muestra cae dentro del hiper-rectángulo, entonces la distancia será cero. Y si el hiper-rectángulo es un punto en el espacio entonces esta ecuación es equivalente al cálculo de distancia euclídeo en el espacio de *D* dimensiones.

Por lo tanto, si se presenta una nueva muestra *x* al modelo de datos y se desea determinar a que clase pertenece dicha muestra, se siguen estos pasos:

- Buscar la regla que cumpla la condición para *x* y establecer como clase de *x* la clase de la regla hallada.
- Si no existe ninguna regla que cumpla la condición para *x*, entonces se pasa a buscar el hiper-rectángulo que se encuentre más cerca de *x* utilizando la Ecuación 2-2, estableciendo como clase de *x* a la misma clase del hiper-rectángulo hallado.

Notar que esta función al medir distancias de puntos en el espacio hacia aristas o vértices de los hiper-rectángulo es posible determinar los límites extendidos de cada hiper-rectángulo para cubrir todo el espacio de búsqueda y así modificar las reglas para que siempre den una respuesta y así no sea necesario recurrir a la Ecuación 2-2 (Figura 2-30).

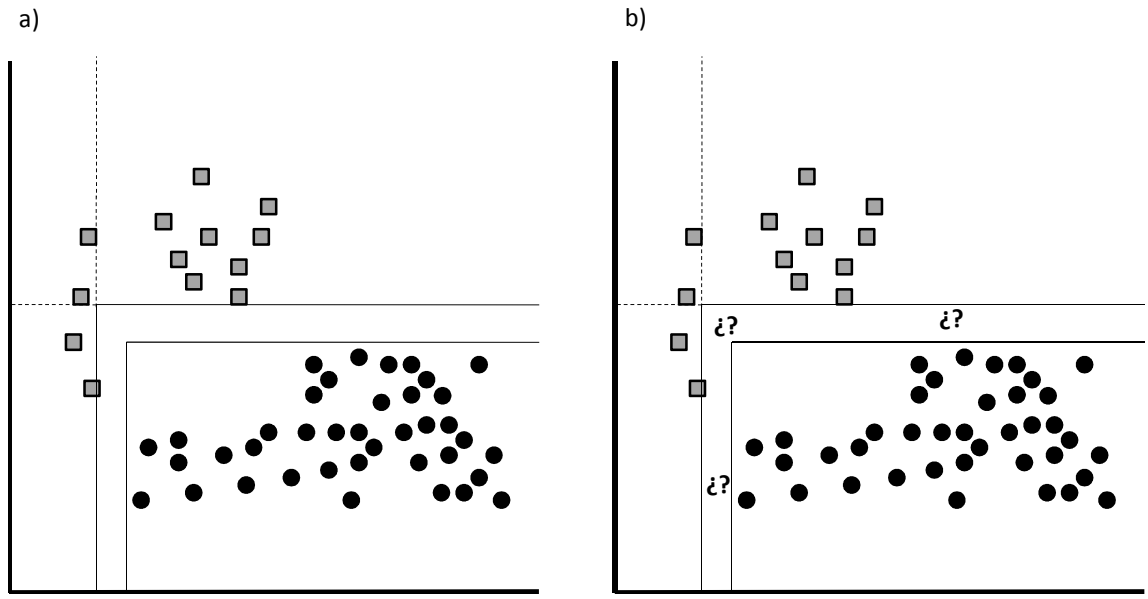

Figura 2-29. Esta figura muestra como queda cubierto el espacio con las reglas de clasificación en el ejemplo de la Figura 2-26 (a) y como quedan zonas del espacio sin cubrir (b).

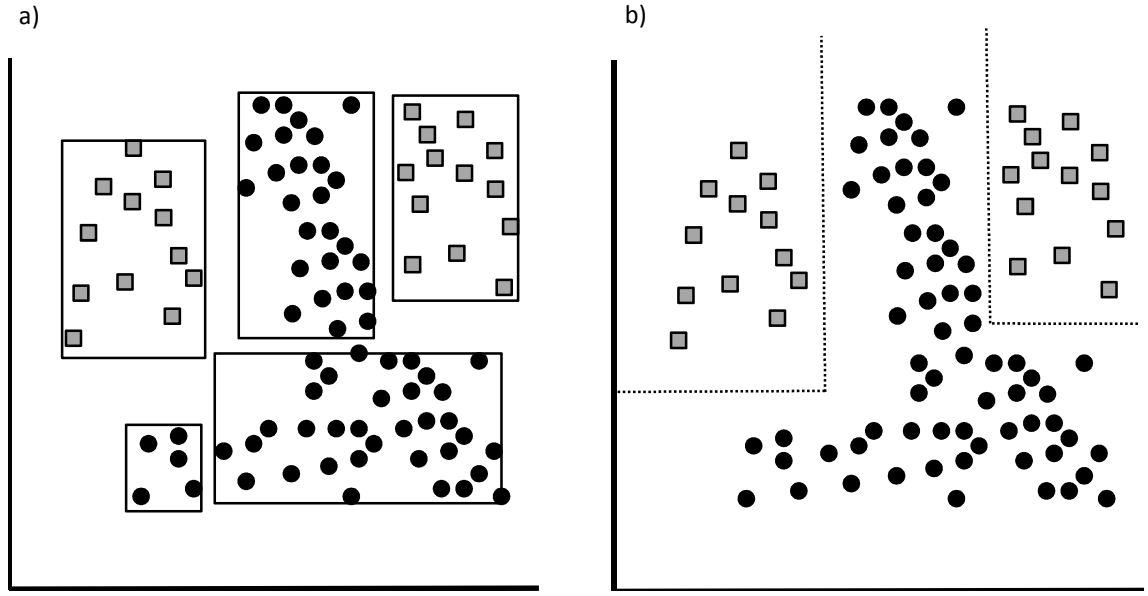

Figura 2-30. Esta figura muestra el resultado de un modelo de datos con sus hiper-rectángulos ajustados (a) y como quedarían determinados los límites extendidos de las reglas extraídas (b).

Esta extensión de los límites obviamente es opcional y se hará solo si el usuario lo desea. De la misma manera, el usuario tiene la posibilidad de cubrir el espacio de búsqueda estirando los límites con otros criterios, como por ejemplo, que sea cubierto por hiper-rectángulos de una clase particular.

7. Intervención del experto

A lo largo de todo el capítulo se observó que el algoritmo de clasificación, para llevar a cabo el armado del modelo de datos, tuvo que tomar muchas decisiones. La decisión de que superposición resolver es decidida

mediante el cálculo de los índices Ω. Aun así, se mencionó que es posible calcular este índice de mucha formas al mismo tiempo que es posible ponderar de diferente manera aquellos elementos utilizados para el cálculo del índice Ω. Por otro lado, una vez que se estableció que superposición eliminar surge la duda de si hay que dividir los hiper-rectángulos, disminuir su volumen, cual dividir o como llevar a cabo la disminución. En esta dirección, la técnica propuesta en esta tesis no determina ninguna decisión en particular, ya que esta depende fuertemente del problema.

De todas formas, es posible tomar todas estas decisiones previo al armado del modelo, configurar el algoritmo para que lleve a cabo el armado y ejecutarlo de manera totalmente automática para lograr el resultado. Aunque la técnica propuesta en esta tesis es completamente flexible en el sentido que un experto, previo a la ejecución de proceso, puede determinar que índices utilizar y como llevar a cabo las divisiones y así personalizar el proceso automático, en problemas reales nunca es posible armar un proceso automático que sea capaz de mejorar a las decisiones "on-line" que tomaría un humano, más cuando este es experto en el domino del problema.

Longbing Cao, en su reciente publicación (Cao, 2010) introduce una nueva metodología denominada "Domain-Driven Data Mining (D^3M)" cuyo objetivo es el de superar los problemas que se han encontrado en el proceso tradicional de data mining. Esta nueva metodología presenta la novedad de incluir en la solución de un problema nuevas características logrando una extracción de conocimiento con la capacidad de sugerir al usuario tomar acciones en concreto para su provecho en el mundo real. Entre estas nuevas características a incluir en la solución de un problema se incluyen un proceso de data mining interactivo, ayuda de expertos, recursos de red, soporte de interacción social y cognitiva, medir conocimiento accionable en problemas multi-objetivos, procesos de data mining operables, ejecutables y confiables que resulten amigables para el usuario, entre otros.

En esta dirección, la estrategia presentada en esta tesis presenta la particularidad de ser utilizada como un proceso totalmente automático o un proceso interactivo mediante la intervención de un experto en el dominio del problema. Esta estrategia tiene las siguientes características:

- Flexibilidad en el uso de los índices.
- Capacidad de darle distintos pesos a cada clase.
- Libertad para decidir cómo llevar a cabo una división o disminución de volumen.
- Posibilidad de ejecutar el proceso de manera totalmente automática, totalmente supervisada o semi-supervisada por un experto.

Por las características mencionadas anteriormente, resulta interesante contar con una herramienta que le permita al experto supervisar el armado del modelo, logrando que este pueda tomar parte en una y cada una de las superposiciones que se deben eliminar.

Si bien esta herramienta no ha sido desarrollada en esta tesis, resulta interesante plantear su implementación como trabajo futuro. Esta herramienta debería poseer las siguientes características:

- Tener la facilidad para incluir o excluir índices Z y sus respectivos pesos V
- Poder ver los resultados del cálculo de los índices Ω y poder decidir eliminar una superposición aunque esta no sea la del Ω de mayor valor, ya que las características de las clases involucradas o la superposición misma resulta más importante de ser eliminada en ciertos casos.
- Poder visualizar los resultados si se hiciera tal o cual división para que el experto pudiera decidir cual llevar a cabo en un cierto momento del algoritmo. Incluso es posible plantear el análisis de todo un árbol de *n* niveles con posibles alternativas.
- Ver cuales superposiciones existen en un mismo espacio de trabajo, que clases están involucradas y cuantos datos intervienen.

- La herramienta podría tener la capacidad de aprender de las decisiones que toma el experto para poder ayudarlo en los siguientes pasos, ya sea porque el experto le da más importancia a una clase o una cierta característica del problema.

- El experto podría contar con ayuda numérica viendo la cantidad de datos participantes, zona del espacio donde está involucrada una superposición, información estadística de los datos participantes, matriz de confusión, etc.

Capítulo 3

Adaptabilidad y actualización del modelo de datos

No basta con alcanzar la sabiduría, es necesario saber utilizarla.
(Marco Tulio Cicerón)

En el capítulo anterior se presentó en detalle el proceso de clasificación de la técnica propuesta en esta tesis, denominada CLUHR. Esta técnica logra armar un modelo de datos que aporta al usuario dos principales contribuciones: la extracción de conocimiento en forma de reglas de clasificación que permite conocer las características de los datos de estudio y, la obtención de un modelo de datos capaz de realizar predicciones sobre nuevas muestras. En este capítulo se verá una tercera contribución del modelo de datos, que consiste en la capacidad de éste para adaptarse a los cambios que sufren los datos sin necesidad de llevar a cabo el armado del modelo comenzando desde cero.

Desde hace varios años se han propuesto varias técnicas en las cuales un modelo de datos ya armado actualiza su estructura interna ante los cambios producidos en la base de datos. Estas técnicas, incluidas en el área de la minería de datos incremental, se han aplicado a distintas tareas entre las que se incluyen clustering, frequent pattern mining y clasificación. La principal ventaja que presentan estas técnicas es que el modelo de datos armado es adaptado a los nuevos cambios sin necesidad de llevar a cabo todo el proceso desde cero, logrando así una técnica mucho más eficiente, ya que, en vez de re-entrenar el modelo de datos con la base de datos completa, sólo se modifica aquella parte de la estructura interna del modelo de datos que es afectada ante el cambio de la base de datos.

En esta dirección, la estrategia propuesta en esta tesis es capaz de actualizar su estructura interna de manera eficiente ante la aparición de nuevos datos y ante la eliminación o modificación de los datos ya representados por el modelo. Se detalla el método de actualización ante la llegada de nuevos datos, eliminación de datos existentes y dos problemas puntuales y menos frecuentes en el mundo real: cambio de clase de datos y sub-clasificación.

Con estas características, CLUHR se transforma en una técnica que es capaz de extraer nuevo conocimiento sin perder el adquirido previamente.

Finalmente, al igual que en el capítulo anterior, se discute como un experto en el dominio del problema puede supervisar la adaptación del modelo de datos ante la aparición de nuevos datos.

1. Adaptabilidad del modelo

Una técnica de data mining para que pueda ser considerada como una estrategia adaptable debe ser capaz de modificar su estructura interna ante la llegada de nuevos datos, debe permitir eliminar datos que se vuelvan innecesarios, corregir datos mediante el cambio de su clase y realizar una posible sub-clasificación de datos sin recurrir a la re-ejecución del algoritmo o proceso que se llevó a cabo para lograr dicho modelo.

Una técnica que es adaptable posee la capacidad de realizar con muy poco esfuerzo computacional las modificaciones pertinentes en su estructura interna para reflejar los cambios de los datos. La capacidad de adaptación logra un proceso de extracción de conocimiento más eficiente y transparente en la tarea de clasificación. Esto hace que resulte una técnica más eficiente y amigable para el usuario por las modificaciones mínimas necesarias que se realizan. Además es transparente, ya que el propio usuario puede ver qué parte del modelo sufre los cambios, y en el caso puntual de CLUHR que reglas de clasificación sufren modificación.

Las estrategias adaptativas se diferencian de las que no lo son por que estas últimas deben ser re-ejecutadas con toda la base de datos completa, esto incluye un gran esfuerzo computacional y, eventualmente, la pérdida de conocimiento adquirido previamente. En el caso particular de CLUHR, si el armado del modelo se realizó de manera supervisada por un experto en el dominio del problema (con todas las decisiones que pudo haber tomado y que se discutieron en el capítulo anterior), no es posible asegurar que una re-ejecución del proceso logre el mismo resultado que se poseía anteriormente. Por eso, es de suma importancia que las estrategias tengan la capacidad de adaptarse a los cambios de los datos.

La adaptabilidad llevada a cabo por CLUHR es "on-line", realiza solo mínimas modificaciones en su estructura interna y, por lo tanto, el conocimiento adquirido previamente "sufre" pequeños cambios. Además, las modificaciones necesarias pueden ser llevadas a cabo de manera totalmente automática, o como se verá más adelante y al igual que sucede durante el armado del modelo, ser supervisada por un experto en el dominio del problema.

1.1. Precondiciones

Para que el modelo armado con esta estrategia sea capaz de adaptarse a los cambios de los datos es necesario que los hiper-rectángulos que conforman el modelo de datos tengan conocimiento del subconjunto de datos a los cuales representa. Como se verá en la siguiente sección, sin esta información disponible es imposible llevar a cabo una adaptación del modelo.

En muchos problemas, por ejemplo en aquellos donde está incluido la seguridad o anonimato de clientes, esta información puede no estar disponible, por lo tanto el modelo para este tipo de problemas será incapaz de llevar a cabo una adaptación.

2. Actualización en línea

Ya sea con datos etiquetados o con datos sin etiquetar, el modelo armado con la estrategia propuesta en esta tesis tiene la capacidad de adaptarse a los cambios de manera totalmente manual o mediante la supervisión del usuario o el experto.

La estructura interna del modelo de datos está conformada por un conjunto de hiper-rectángulos los cuales representan datos de distintas clases. Estos hiper-rectángulos, según la decisión del experto al momento de armar el modelo, pueden presentar superposición en el espacio o no. En el caso de que existan superposiciones, también se habrá determinado qué prioridad tiene cada una de las reglas de clasificación para que el modelo pueda predecir muestras sin contradicciones.

La adaptación del modelo de datos se lleva a cabo mediante la modificación de los límites de los hiper-rectángulos, esta modificación incluye la expansión o disminución del volumen de los mismos y la división en hiper-rectángulos más pequeños.

Para que la adaptación del modelo de datos pueda ser llevada a cabo de manera automática, el proceso debe ser capaz de tomar decisiones, éstas incluyen los mismos criterios del armado del modelo, mencionados en el capítulo anterior: conocer el número mínimo de datos participantes en una superposición para llevar a cabo una re-estructuración (parámetro μ del algoritmo), el criterio a utilizar para la división los hiper-rectángulos o el criterio para ajustar los límites del hiper-rectángulo en caso de ser necesario.

2.1. Agregando nuevos datos

Para que el modelo sea capaz de adaptarse, se necesita que el nuevo dato que se presenta al modelo tenga asignada una clase. Por lo general, el usuario del modelo agregará datos sabiendo a qué clase pertenecen. Pero en el caso de que no se conozca la clase del dato, entonces se puede obtener la clase mediante la propia predicción del modelo y, si el usuario lo desea, agregar dicho dato con la clase predicha por el modelo. Esta aclaración es importante hacerla ya que como se verá más adelante, esta información puede ser utilizada en distintas acciones del proceso de adaptación.

Ya sea que el nuevo dato tenga una clase conocida o si la clase del nuevo dato es predicha por el modelo, éste sólo es capaz de adaptarse si el dato a incorporar tiene una clase asignada. Al momento de incorporar un nuevo dato pueden ocurrir varios escenarios:

- El nuevo dato está dentro de los límites de un hiper-rectángulo.
- El nuevo dato está dentro de los límites de una superposición (este caso puede ocurrir cuando durante el armado del modelo se permitió la existencia de ciertas superposiciones).
- El nuevo dato no está dentro de los límites de ningún hiper-rectángulo del modelo.

Además, hay que tener en cuenta que el dato puede estar dentro de los límites de un hiper-rectángulo que es representante de la misma clase del nuevo dato o de distinta clase. Todas las acciones que se discuten a continuación son válidas tanto para los casos en que las nuevas muestras sean pertenecientes a clases ya representadas por el modelo de datos como en los casos donde, por la naturaleza del problema a resolver, aparezcan nuevas clases.

Por lo tanto, al agregar un nuevo dato en el modelo, la primer tarea que se realiza es la de verificar si el dato está incluido o no en un hiper-rectángulo. Todas las alternativas que se pueden presentar se discuten a continuación.

2.1.1. El nuevo dato está incluido en un único hiper-rectángulo

Si el nuevo dato *m* que se incorpora al modelo está incluido dentro de un hiper-rectángulo *H* y tanto *m* como *H* son de la misma clase, entonces el modelo de datos no sufre modificación alguna ya que la llegada del nuevo dato dentro de un hiper-rectángulo de su misma clase no produce incoherencia en el conocimiento adquirido.

Por otro lado, si *m* está incluido dentro de un hiper-rectángulo *H* cuya clase no es la misma que la clase de *m*, entonces debe llevarse a cabo una re-estructuración en *H*.

Si al momento de incorporar a *m*, *H* ya incluye datos de la misma clase de *m*, entonces este subconjunto de datos conforman un hiper-rectángulo *J* dentro de *H*. Si *J* existe, entonces se ajustan sus límites para que incluya al nuevo dato *m*. Éste paso podría no producir cambios en *J* si *m* también está incluido en *J*. Por otro lado, si *m* es el primer dato de su clase en estar incluido dentro de *H*, entonces se forma un hiper-rectángulo *J* el cual contendrá a *m* como único dato. En ambos casos, el resultado de incorporar un dato a un hiper-rectángulo resulta en un nuevo hiper-rectángulo *J*, el cual tiene la propiedad de estar completamente incluido dentro de *H*.

Si el parámetro μ establecido para el armado del modelo lo permite, entonces se procede a eliminar la superposición entre *H* y *J*. Para hacer esto se procede al mismo método explicado en la sección 4.2 del capítulo 2, en la que se calculan los índices Ω para encontrar la dimensión por la cual llevar a cabo la división de los hiper-rectángulos. Este proceso de división o ajuste de los hiper-rectángulos incluye controlar a posteriori

las nuevas superposiciones que se formen, modifiquen o que desaparezcan entre los nuevos hiper-rectángulos formados y los restantes hiper-rectángulos que conforman el modelo de datos.

El pseudocódigo para la adaptación de nuevos datos cuando ocurren las condiciones mencionadas en esta sección es el siguiente:

```
m = nuevo dato que se incorpora al modelo

H = hiper-rectángulo que incluye a m

si Clase(m) <> Clase(H) entonces

        J = hiper-rectángulo formado por los datos incluidos en H que
            tienen la misma clase que m

        si J es vacío entonces

                J = hiper-rectángulo formado a partir de m

        Eliminar superposición entre H y J

        fin si

sino

        Agregar m a H

fin si
```

2.1.2. El nuevo dato está incluido en una superposición entre dos hiper-rectángulos

Si el nuevo dato *m* está incluido dentro de una superposición *S* que forman dos hiper-rectángulos *H* y *J*, entonces puede ocurrir dos situaciones. En la primera se cumple que *H* o *J* representa a la misma clase que *m*, en este caso se verifica que se cumpla la condición establecida por el parámetro μ y de ser permitido se lleva a cabo una re-estructuración de los hiper-rectángulos participantes tal como se comentó en la sección 4.2 del capítulo 2.

En la segunda situación puede ocurrir que ni *H* ni *J* sean representantes de la misma clase que *m*. En este tipo de situaciones se realiza la verificación de detectar cuantos datos de la clase de *m* existen en *H* y cuantos existen en *J*. Si existe una superposición entre *H* y *J*, significa que, por la razón que fuera, en el armado del modelo se aceptó tal superposición, por lo tanto esta superposición no es de interés para ser eliminada y la adaptación sólo se concentra en eliminar las superposiciones de *H* versus la clase de *m* por un lado, y de *J* versus la clase de *m* por el otro.

Dentro de *H* se detectan todos los datos de la clase de *m* formando un hiper-rectángulo *P*, al mismo tiempo que se detectan dentro de *J* todos los datos de la clase de *m* formando un nuevo hiper-rectángulo *Q*. Estas dos acciones producen dos nuevas superposiciones, por un lado entre *H* y *P* y por el otro entre *J* y *Q*. De esta manera se obtienen dos nuevas superposiciones y se utilizan los cálculos de los índices Ω (sección 4.2 del capítulo 2) para eliminar o minimizar las dos nuevas superposiciones. Al igual que en los casos previos, esta acción implica revisar nuevas superposiciones entre los hiper-rectángulos formados.

Nótese que este último caso es similar al explicado en la sección previa pero en un sentido "doble", ya que hay dos superposiciones para eliminar. En vez de eliminar una superposición primero y luego la otra (lo cual implica decidir cuál se elimina primero), se determina el orden de eliminación mediante el cálculo de los índices Ω. De esta manera la eliminación de las superposiciones se realiza del mismo modo que durante el armado inicial del modelo y se preserva la consistencia en la metodología de trabajo.

Nótese además que a pesar de ser de la misma clase, los datos de la clase de *m* que están dentro del hiper-rectángulo *H* forman un hiper-rectángulo distinto al que forman los datos de la clase de *m* que están incluidos dentro de *J*. Estos dos hiper-rectángulos de la misma clase pueden presentar una superposición, pero justamente porque son de la misma clase su existencia no acarrea problemas. La razón por la cual se crean dos hiper-rectángulos de la misma clase es simple: los dos hiper-rectángulos, *P* y *Q*, están completamente incluidos en otros dos, *H* y *J* respectivamente. Esto implica la ventaja de no generar ninguna superposición con ningún otro hiper-rectángulo del modelo, algo que no se puede garantizar si se genera un único hiper-rectángulo con los datos de la clase de *m* presentes en *H* y *J* (Figura 3-1). Si esta condición se permite y se genera un único hiper-rectángulo, la aparición del nuevo dato *m* causaría la modificación no sólo de los hiper-rectángulos *H* y *J* por caer en su mismo espacio, sino de otros hiper-rectángulos que se verían afectados por la construcción de un hiper-rectángulo que excede los límites de manera innecesaria de *H* y *J*. La principal desventaja de llevar a acabo esta forma de proceder, que es la razón por la que no se aplica en CLUHR, es que el modelo de datos puede perder precisión y conocimiento adquirido previamente al modificar hiper-rectángulos que no deben ser modificados.

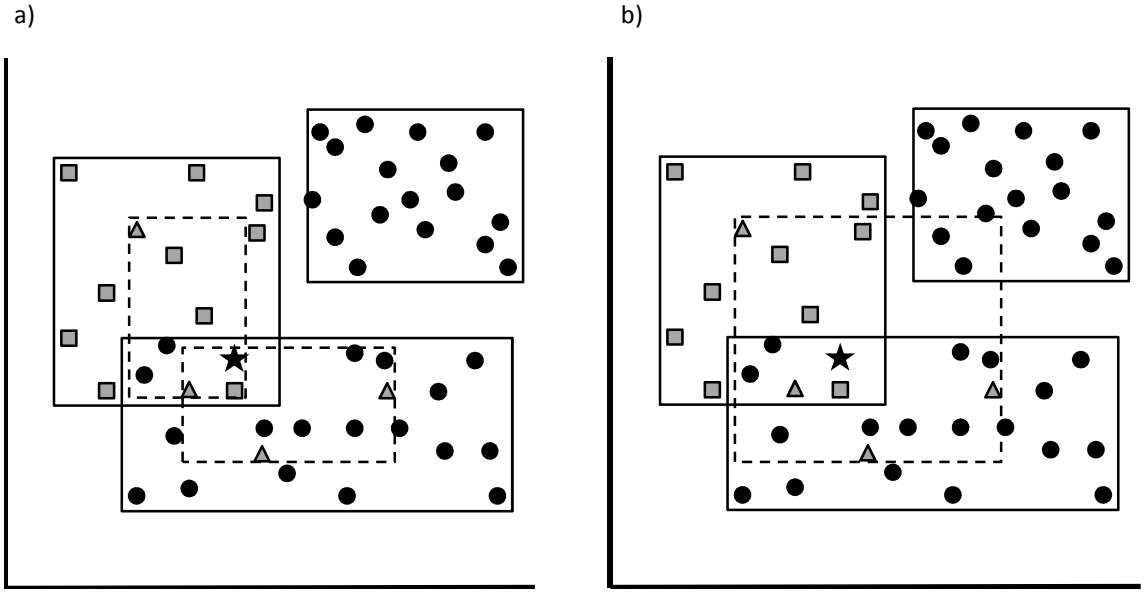

Figura 3-1. Un nuevo dato de la clase triángulo (señalado con una estrella) es introducido al modelo de datos. En a) se ve como quedan formados los dos hiper-rectángulos incluidos completamente en los de las clases de círculos y cuadrados. En b) se observa como el armado de un único hiper-rectángulo de la clase triángulos formado con todos los datos presentes, tanto en H como en J, alcanza a otro hiper-rectángulo que está fuera de la superposición donde cae el nuevo dato.

El pseudocódigo para la adaptación de nuevos datos cuando ocurren las condiciones mencionadas en esta sección es el siguiente:

```
m = nuevo dato que se incorpora al modelo

H = hiper-rectángulo que incluye a m

J = hiper-rectángulo que incluye a m

si (Clase(m) = Clase(H)) entonces

        Proceder como en caso de la sección 2.1.1 eliminando la
        superposición entre H y m

sino

si (Clase(m) = Clase(J)) entonces

        Proceder como en caso de la sección 2.1.1 eliminando la
        superposición entre J y m

sino

// (Clase(m) <> Clase(H)) y (Clase(m) <> Clase(J))

        P = hiper-rectángulo formado por los datos incluidos en H que
            tienen la misma clase que m

        si P es vacío entonces

                P = hiper-rectángulo formado a partir de m

        fin si

        Q = hiper-rectángulo formado por los datos incluidos en J que
            tienen la misma clase que m

        si Q es vacío entonces

                Q = hiper-rectángulo formado a partir de m

        fin si

        Eliminar superposiciones presentes entre H y P

        Eliminar superposiciones presentes entre J y Q

fin si
```

2.1.3. El nuevo dato no está incluido en ningún hiper-rectángulo

Si el nuevo dato *m* no está incluido en ningún hiper-rectángulo del modelo de datos, entonces se procede a forzar su inclusión dentro de un hiper-rectángulo que represente su misma clase. Para lograr esto, se aumenta el volumen de un hiper-rectángulo *H* para que, con sus nuevos límites, incluya a *m*. Este es el único caso donde un hiper-rectángulo aumenta su volumen en vez de reducirlo, particularidad que nunca sucede en el armado inicial del modelo de datos.

Ya que el principal objetivo de CLUHR es lograr un modelo de datos que sea preciso, el hiper-rectángulo que debe aumentar su volumen debe, en su modificación, evitar una intersección con otro hiper-rectángulo que representa a una clase distinta que la de *m*. Por lo tanto, la tarea de encontrar el hiper-rectángulo a ser modificado consiste en hallar aquel hiper-rectángulo *H* que al aumentar su tamaño no presente superposición con ningún otro hiper-rectángulo de otra clase. Si existiera más de un hiper-rectángulo con estas características se elige al que más cerca se encuentre de la muestra, usando la Ecuación 2-2 para medir las distancias (Figura 3-2).

Si ningún hiper-rectángulo puede crecer sin presentar superposición con otro hiper-rectángulo de otra clase, entonces el criterio de elección del hiper-rectángulo resulta distinto a lo explicado anteriormente. Primero se selecciona a todos los hiper-rectángulos que menor cantidad de superposiciones presente, de todos aquellos que cumplan con esa condición se elige al hiper-rectángulo que presente mayor índice Ω.

La razón de elegir el hiper-rectángulo que menos superposiciones presente es que de esa manera el modelo sufre una menor re-estructuración (Figura 3-3). Y en el caso de que existan varios hiper-rectángulos candidatos se elige el que mayor índice Ω obtenga, ya que este índice indica en la magnitud de su valor cuán pequeño es el cambio que se produce en los hiper-rectángulos; a mayor valor, menor es el cambio (ver sección 3 del capítulo 2).

Una vez elegido el hiper-rectángulo *H* que será modificado, se ajustan sus límites para incluir al nuevo dato *m*, mediante la siguiente ecuación:

Ecuación 3-1

$$1.\ Hn_i = \min(Hn_i, m_i)\,, \forall i = 1..D$$

$$2.\ Hx_i = \max(Hx_i, m_i)\,, \forall i = 1..D$$

Una vez que *H* fue modificado para incluir al nuevo dato se procede a eliminar las superposiciones que se hayan formado como se detalla en la sección 4.2 del capítulo 2.

El pseudocódigo para la adaptación de nuevos datos cuando ocurren las condiciones mencionadas en esta sección es el siguiente:

```
m = nuevo dato que se incorpora al modelo

HS = subconjunto de hiper-rectángulos de la misma clase que m y que al
expandirse no presentan superposición con hiper-rectángulos de otras clases

si (HS no es vacío) entonces

        H = hiper-rectángulo de HS que más cerca está de m

sino

        HC = subconjunto de hiper-rectángulos de la misma clase que m
        que al expandirse presentan superposición con hiper-rectángulos
        de otras clases

        H = hiper-rectángulo de HC que menos superposiciones presente y
        con índice Ω más alto

fin si

Ajustar H para que incluya a m

Eliminar superposiciones presentes
```

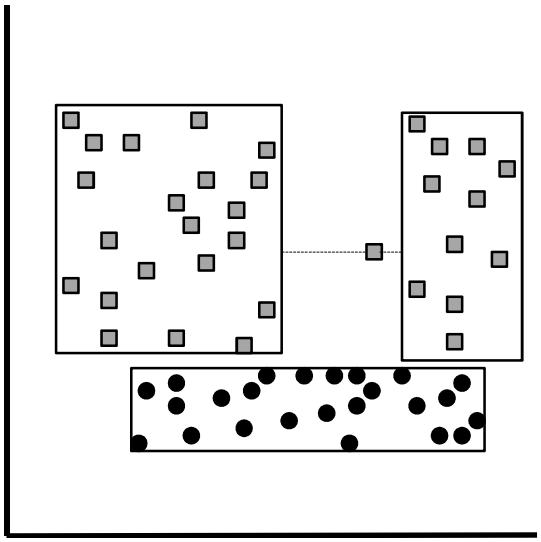

Figura 3-2. En la figura se muestra como un nuevo dato de la clase cuadrado no está incluido por ningún hiper-rectángulo. De los dos hiper-rectángulos que pueden crecer para incluirlo sin presentar superposiciones con otro hiper-rectángulo, el de la derecha se encuentra más cerca de la nueva muestra y por lo tanto será el que se modifique.

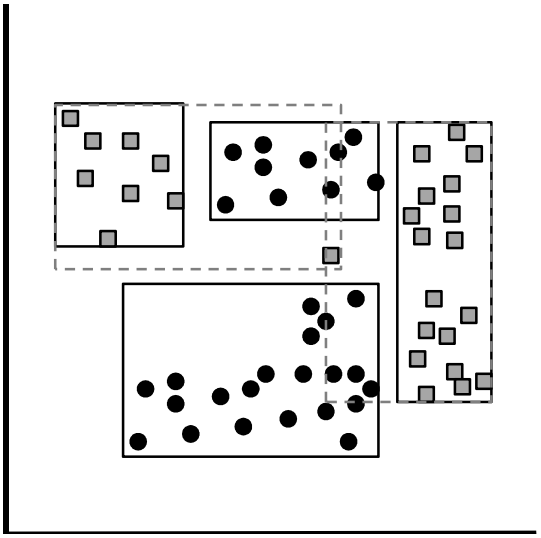

Figura 3-3. En la figura se muestra como un nuevo dato de la clase cuadrado no está incluido por ningún hiper-rectángulo. De los dos hiper-rectángulos que pueden crecer para incluirlo, el de la izquierda presenta una sola superposición, mientras que en el de la derecha presenta dos. En líneas punteadas se muestra cómo quedan formados los hiper-rectángulos luego de la expansión.

2.2. Eliminando datos existentes

Existen problemas en los cuales puede ser necesario eliminar del modelo datos pertenecientes a alguna clase en particular, ya sea porque fue detectado un error y algunos de los datos utilizados para el armado inicial del modelo de datos no debieron ser utilizados, o bien porque la naturaleza del problema permita este tipo de modificaciones, por ejemplo cuando un cliente de tales características realiza la compra de un producto en

particular y a los pocos días devuelve el producto por la razón que sea, en este caso la transacción que se guardó en el modelo ya no tiene sentido seguir almacenándola.

En estos casos puede resultar interesante modificar la estructura interna del modelo de datos ya que de dicha re-estructuración puede ocurrir que desaparezcan superposiciones existentes y al mismo tiempo pueden liberarse zonas del espacio que permiten crecer libremente a otros hiper-rectángulos.

Ante la necesidad de eliminar datos de alguna clase en particular pueden ocurrir distintos escenarios:

- La eliminación del dato no produce ninguna re-estructuración.
- La eliminación del dato produce una disminución del volumen de uno de los hiper-rectángulos.

2.2.1. El dato está incluido en un hiper-rectángulo representante de otra clase

Como se vio repetidamente, durante el armado del modelo pueden quedar datos de una clase dentro de hiper-rectángulos representantes de otra clase (Figura 2-14). En estos casos la eliminación de estos datos no produce ningún tipo de re-estructuración ya que el hiper-rectángulo no se ve afectado al no tener dentro de sus límites un dato que pertenezca a otra clase. Por el contrario, el modelo se ve afectado de manera positiva ya que en estos gana precisión al eliminar falsos positivos.

Por lo tanto la única acción a llevar a cabo en estos casos es la eliminación del dato tanto de la base de datos como del modelo.

2.2.2. El dato está incluido en un hiper-rectángulo representante de su misma clase

La eliminación de un dato *m* de un hiper-rectángulo *H* que representa su misma clase puede producir una modificación de *H*. Si los límites de *H* no estaban determinados por los valores del dato, entonces *H* no sufre modificación. Por otro lado, si al menos en una dimensión, los límites de *H* quedan determinados por los valores de *m* entonces *H* debe ser re-estructurado (Figura 3-4).

La re-estructuración mencionada anteriormente dependerá exclusivamente del problema y de las necesidades del usuario, ya que podría no llevarse a cabo. Ante el ajuste de un hiper-rectángulo surge la pregunta de por qué "perder" zonas del espacio donde había datos de la clase, ya que en un futuro podrían aparecer nuevas, además de que el modelo podría seguir haciendo predicciones sobre los datos que caigan en dicha zona. Por el otro lado, y como se mencionó anteriormente, la liberación de zonas del espacio no solo elimina posibles superposiciones sino que además da a lugar la posibilidad de "crecer" libremente a otros hiper-rectángulos. Por lo tanto y ante estas características, llevar a cabo la re-estructuración del hiper-rectángulo dependerá del problema que se quiera solucionar y de las necesidades del usuario.

En caso de llevar a cabo la re-estructuración, entonces se excluye el dato *m* del modelo y sin dicho dato el hiper-rectángulo *H* es ajustado a los nuevos límites. Estos nuevos límites serán los que queden determinados luego de encontrar el hiper-rectángulo representativo mínimo de los datos restantes. Esta acción puede producir que algunas superposiciones desaparezcan al cambiar el volumen de *H* (Figura 3-4). Si *m* era el único dato de *H*, entonces *H* es eliminado del modelo ya que no representa a ningún dato.

Un dato que se elimina puede llegar a producir que se puedan unir otros hiper-rectángulos como lo muestra la Figura 3-5. Para detectar estos tipos de casos la única manera es la de revisar todos los posibles pares de hiper-rectángulos y ver si al unirlos en un único hiper-rectángulo no presentan superposición con otros hiper-rectángulos de otras clases.

La ventaja de unir hiper-rectángulos tiene como objetivo la reducción de reglas de clasificación, las necesidades del usuario determinarán si se lleva a cabo esta tarea. La técnica propuesta en esta tesis no sugiere ninguna heurística para llevar a cabo esta tarea.

El pseudocódigo para la adaptación cuando ocurren las condiciones mencionadas en esta sección es el siguiente:

```
m = dato que se elimina del modelo

H = hiper-rectángulo que incluye a m

Eliminar de H el dato m

si H es vacío entonces

        Eliminar H del modelo de datos

sino

        Ajustar H a sus nuevos límites

fin si
```

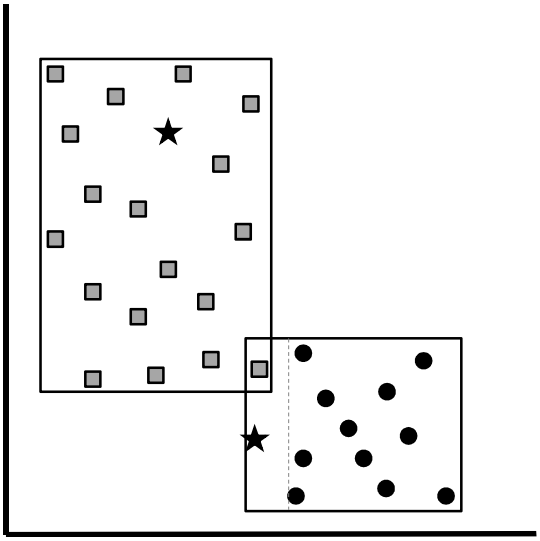

Figura 3-4. La eliminación del dato de la clase cuadrado (señalado con una estrella) no produce modificación en el hiper-rectángulo correspondiente. La eliminación del dato de la clase círculo (señalado con una estrella) produce que se ajuste a nuevo hiper-rectángulo representante mínimo (el nuevo límite inferior sobre el eje X queda señalado por la línea rayada). Esta acción además tiene la particularidad de que elimina la superposición entre ambos hiper-rectángulos.

2.3. Modificación de la clase de los datos

CLUHR es capaz de adaptarse para llevar a cabo la corrección de errores. Si se determina que datos de cierta clase en realidad deben pertenecer a otra clase entonces el modelo de datos lleva a cabo una re-estructuración que reflejará dichos cambios.

Si un dato *m* cambia de clase entonces pueden ocurrir dos posibles escenarios:

- La modificación del dato no produce re-estructuración alguna.
- La modificación del dato produce un ajuste en los hiper-rectángulos.

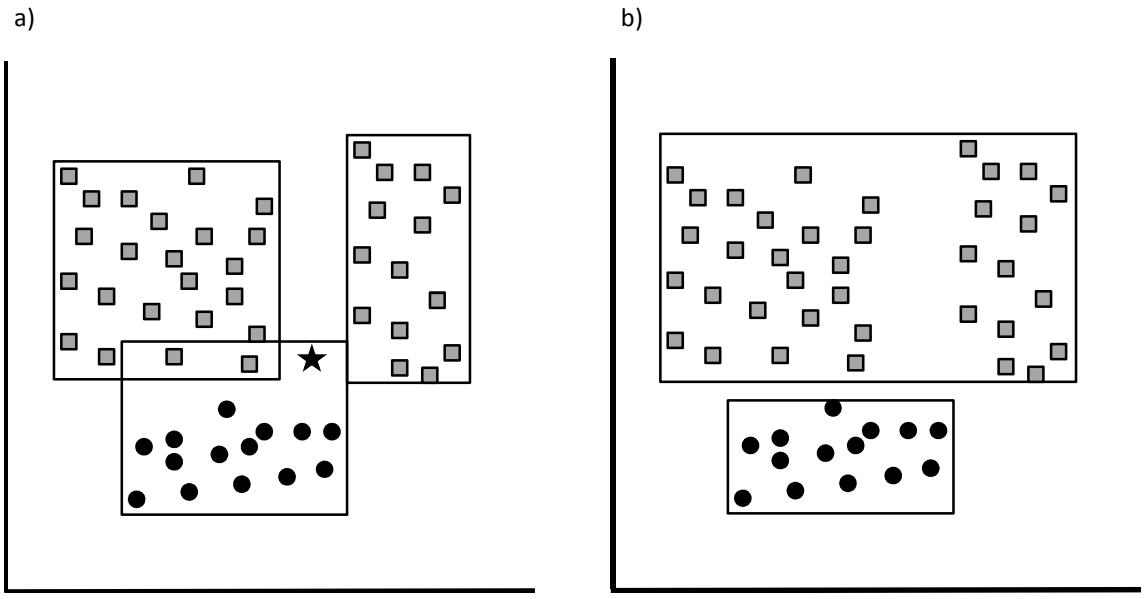

Figura 3-5. En a) se muestra un dato de la clase círculo (señalado con una estrella) que al ser eliminado y al ajustar el hiper-rectángulo de la clase círculo, da lugar a que los dos hiper-rectángulos de la clase cuadrado puedan unirse en un único hiper-rectángulo sin presentar superposición con otros hiper-rectángulos (b).

2.3.1. El dato está incluido en un hiper-rectángulo de la misma clase a la cual cambia el dato

Si un hiper-rectángulo que representa a datos de una clase C1 tiene incluidos datos de una clase C2 y el cambio consiste en cambiar la clase de tales datos a C1, entonces no hay que hacer nada, ya que el dato ahora está siendo representado por un hiper-rectángulo de su propia clase. Además el modelo gana en precisión al ser eliminado un falso positivo.

2.3.2. El dato está incluido en un hiper-rectángulo que representa a otra clase distinta

Si luego del cambio de clase, un dato *m* queda incluido en un hiper-rectángulo de otra clase, entonces se procede a realizar un ajuste del hiper-rectángulo *H*, como en la eliminación de un dato (sección 2.2.1).

Luego de la re-estructuración de *H*, se puede ver a *m* como un nuevo dato y entonces se procede a actuar según como haya resultado la re-estructuración. Si *m* queda incluido dentro de *H*, entonces se procede a realizar las mismas acciones que se detallaron en las secciones 2.1.1 y 2.1.2 según sea el caso. Si *m* no queda incluido dentro de ningún hiper-rectángulo, entonces se procede de la misma manera que la descripta en la sección 2.1.3.

```
m = dato que cambia a la clase C

H = hiper-rectángulo que incluye a m

si Clase(m) <> Clase(H) entonces

        Sacar m de H

        Ajustar H

        Proceder como en la sección 2.1

fin si
```

2.4. Sub-clasificando muestras

A menudo es posible encontrar problemas donde la naturaleza de los datos y su clasificación es un proceso de estudio y de una constante actualización. Por ejemplo, en un sistema financiero donde se desea determinar si otorgar o no préstamos personales según las características del cliente, el sistema puede considerar la existencia de dos tipos de clientes, los que son confiables y lo que no lo son. Después de un cierto tiempo de uso, la entidad financiera por algún motivo puede decidir subdividir la categoría de usuarios confiables a usuarios muy confiables y usuarios no tan confiables. Pasando el sistema a tener datos de tres clases distintas.

Un modelo de datos adaptable debe ser capaz de acomodarse a estos cambios sin la necesidad de realizar grandes re-estructuraciones y mucho menos un re-entrenamiento del modelo. La estrategia propuesta presenta la capacidad de poder sub-clasificar datos de una clase llevando a cabo sólo las re-estructuraciones necesarias.

Para realizar esto, se determina el subconjunto de datos a sub-clasificar y se re-etiquetan todos los datos que pertenecen a tal sub-conjunto. Luego se forman para cada clase el hiper-rectángulo representativo mínimo y se procede a eliminar las superposiciones que se formen entre estos, o con otros hiper-rectángulos de otras clases, de la misma forma que se detalla en la sección 4.2 del capítulo 2.

El pseudocódigo del proceso de subclasificación de muestras es el siguiente:

```
ds = datos de la clase C a sub-clasificar

Hs = hiper-rectángulos que representan a la clase C

eliminar del modelo todos los hiper-rectángulos de Hs

para cada nueva clase Ci hacer

        sc = sub-conjunto de ds formado por los datos de la clase Ci

        armar el hiper-rectángulo representativo de sc

fin para

Proceder con la eliminación de superposiciones (sección 4.2 del capítulo 2)
```

Nótese que en este tipo de acciones se puede aplicar una heurística para no llevar a cabo la re-estructuración explicada anteriormente. Si por la razón que fuera, la decisión del usuario es la de sub-clasificar según las reglas extraídas del modelo de datos, entonces sólo es necesario re-etiquetar los datos y las reglas sin realizar ninguna otra modificación.

En el ejemplo detallado en la sección 5 del capítulo 2 donde había tres reglas, dos para la clase cuadrado y una para la clase círculo, el usuario del modelo podría determinar que la sub-clasificación de la clase cuadrado se realice en base a sus reglas, así los datos que satisfacen la regla 1 serán de la clase "Cuadrado 1" y los que satisfacen la regla 2 serán de la clase "Cuadrado 2". En este caso sólo se re-etiquetan los datos y las propias reglas, dejando la estructura interna del modelo y los límites de los hiper-rectángulos sin modificar.

2.5. Realizando varios cambios simultáneamente

Como se mencionó en el capítulo anterior una de las principales características de CLUHR es que es una técnica determinista, por lo tanto una misma entrada (una base de datos) siempre produce una misma salida (un modelo de datos). Para que el proceso de adaptabilidad presente la misma característica determinista y, al mismo tiempo, la técnica coherente en su forma de trabajar, entonces todos los cambios a realizar deben ser hechos al mismo momento y en una misma operación.

Si bien es posible hacer la adaptación del modelo presentando un dato por vez, el orden en el cual se presentan estos datos puede producir modelos diferentes (Figura 3-6). Para evitar estas ambigüedades se sugiere que todos los cambios sean realizados en una única operación.

Hay que notar que esto es posible siempre que el problema planteado permita la "acumulación" de cambios a realizar. En problemas donde el modelo deba adaptarse de forma on-line a medida que surja un cambio, por ejemplo en un problema de detección de fallas, estos cambios deberán hacerse sin esperar al próximo cambio. Por ende, el modelo que resulte de estos cambios será dependiente de cómo fueron surgiendo dichos cambios. Por otro lado, si el problema lo permite, entonces se recomienda que se "acumulen" varios cambios a realizar para poder efectuarlos juntos en una única operación.

La adaptación del modelo ante varios cambios se realiza siguiendo estas acciones:

- Para cada nuevo dato que se presente en el modelo se busca el hiper-rectángulo *H* que más cerca esté y de ser necesario se amplia *H*.
- Los datos que deben ser eliminados son sacados del modelo y se identifica los hiper-rectángulos que resultaron afectados.
- Se re-etiquetan los datos con cambio de clase y se identifican los hiper-rectángulos que resultaron afectados.
- Se llevan a cabo las sub-clasificaciones necesarias creando nuevos hiper-rectángulos representativos mínimos.

Luego de todos estos cambios se procede a actualizar el modelo de datos como se indica en la sección 4.2 del capítulo 2 hasta disminuir las superposiciones formadas. Cabe aclarar que en este punto sólo es necesario revisar los hiper-rectángulos identificados como modificados, dejando intacto el resto del modelo.

El pseudocódigo para la modificación ante la acumulación de varias modificaciones es el siguiente:

```
dsa = datos a agregar al modelo
para cada dato d de dsa hacer
        H = hiper-rectángulo más cercano a d
        ampliar H para que incluya a d
fin para
dse = datos a eliminar del modelo
para cada dato d de dse hacer
        H = hiper-rectángulo representante de d
        eliminar d de H
fin para
dsc = datos que sufrieron un cambio de clase
para cada dato d de dsc hacer
        H = hiper-rectángulo más cercano a d
        ampliar H para que incluya a d
fin para
realizar sub-clasificaciones (sección 2.4)
Proceder con la eliminación de superposiciones (sección 4.2 del capítulo 2)
```

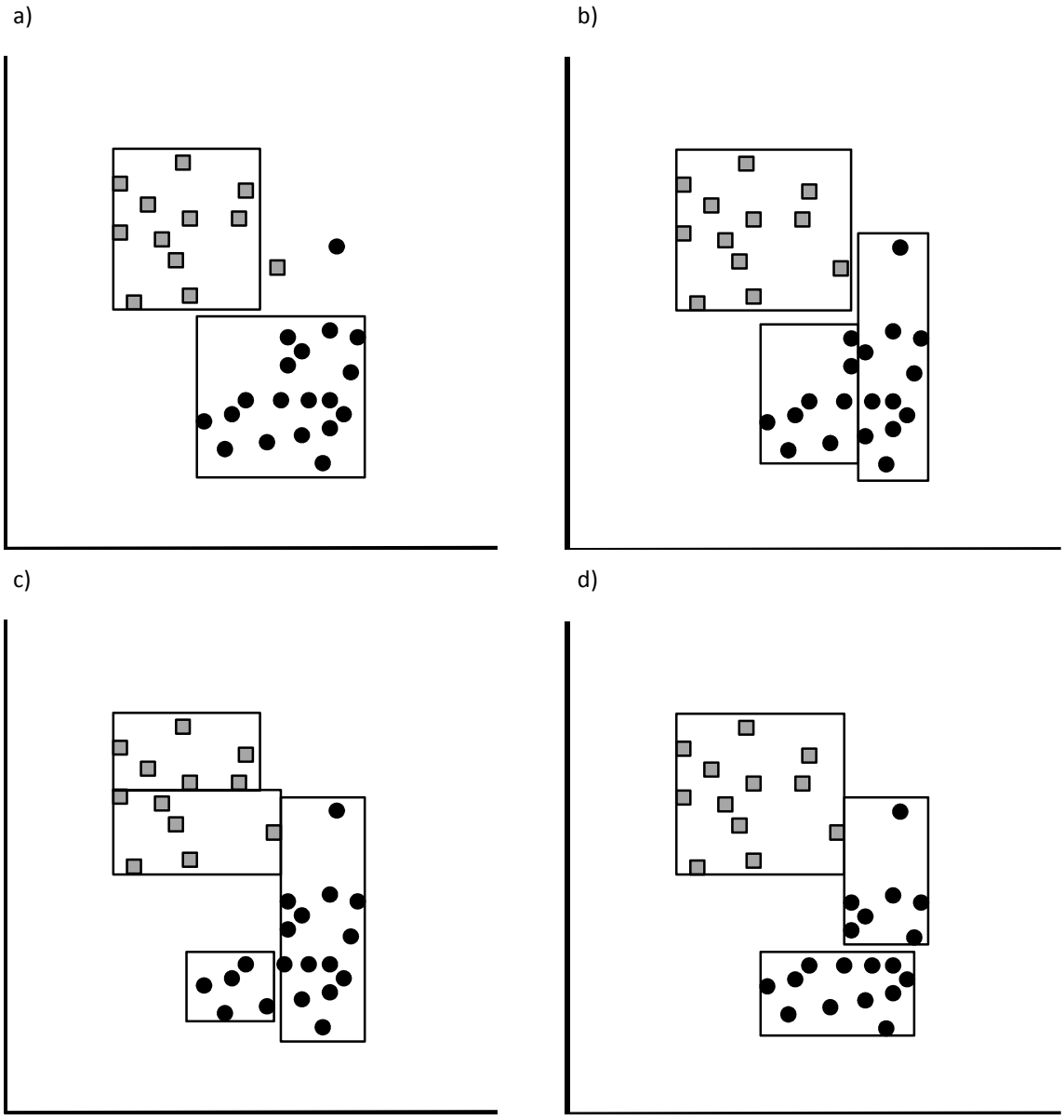

Figura 3-6. Este ejemplo muestra que el agregado de un dato de la clase cuadrado primero y un dato de la clase círculo después, puede producir diferentes modelos de datos según el orden en que se procesen dichos datos. En a) se muestra el modelo actual y donde se incorporan los nuevos datos (ambos fuera de los hiper-rectángulos). En b) se muestra lo que sucede al agregar primero el dato cuadrado y luego el dato círculo. El hiper-rectángulo de la clase cuadrado crece sin inconvenientes y, luego cuando crece el hiper-rectángulo de la clase círculo, esta es dividida en dos hiper-rectángulos. En c) se muestra el resultado de agregar primero el círculo y luego el cuadrado. Primero crece el hiper-rectángulo de la clase círculo obligando a dividir al hiper-rectángulo de la clase cuadrado, luego crece uno de los hiper-rectángulos de la clase cuadrado causando la división en el hiper-rectángulo de la clase círculo. En d) se muestra el resultado de procesar ambos cambios al mismo tiempo, luego de analizar los índices Ω, producto de la superposición.

3. Actualizando reglas de clasificación

La actualización de las reglas ocurre de manera casi directa con la adaptación producida en el modelo. Como la adaptación del modelo consiste en modificar los límites de un hiper-rectángulo, entonces, para la regla

correspondiente al hiper-rectángulo modificado se actualizan las cláusulas correspondientes, y las reglas ya están actualizadas y reflejan de esta manera el nuevo conocimiento extraído.

Cuando se llevan a cabo actualizaciones en las cuales la estructura del modelo sufre importantes modificaciones, entonces las actualizaciones de las reglas no son tan fáciles de realizar junto a la modificación del modelo. Más cuando aparecen nuevos hiper-rectángulos o se eliminan superposiciones formadas.

Como se explicó en el capítulo anterior, no es objetivo de esta tesis realizar una conversión eficiente de hiper-rectángulos a reglas. Ya que es posible realizar el mismo procedimiento detallado en la sección 5.1 del capítulo 2. Por lo tanto y para la implementación de CLUHR, si la adaptación del modelo produce una re-estructuración importante del modelo, es posible reemplazar el conjunto de reglas original por uno nuevo extraído por el mismo procedimiento ya visto, el cual consiste en la extracción del conjunto de reglas simplificadas.

El uso del modelo en la tarea de predicción sigue siendo el mismo y puede utilizarse de la misma manera que se explicó en la sección 6 del capítulo 2.

4. Intervención del experto

Como se vio en todos los procedimientos de adaptación a los cuales puede estar sometido un modelo de datos, algunos son de resolución simple (incluso sin re-estructuración) y otros pueden producir una importante cantidad de cambios.

En este último caso es posible que el experto del problema desee supervisar la adaptación ya que le puede resultar de interés sugerir las modificaciones que deban realizarse ante la necesidad de eliminar una superposición. Como todas las superposiciones se eliminan siguiendo el mismo procedimiento que se detalló en la sección 4.2 del capítulo 2, el experto puede supervisar la adaptación del modelo de la misma manera que si lo estuviera creando por primera vez, dándole más peso o prioridad a los casos que resulten de interés, al mismo tiempo que puede configurar la herramienta de la manera que lo necesite.

Así, la herramienta propuesta como trabajo a futuro en la sección 7 del capítulo 2 puede hacer uso de las mismas acciones para llevar a cabo la tarea de adaptación.

Una de las características que se pueden implementar en esa herramienta es una especie de "memoria" donde la herramienta sea capaz de recordar qué operación llevó a cabo el experto ante un cambio, para poder sugerírselo cuando en un futuro ocurra un caso similar; por ejemplo, si el experto decidió dividir un hiper-rectángulo de una clase dejando intacto otro hiper-rectángulo de otra clase, la herramienta podría sugerir la misma acción en el futuro.

Otra de las características que puede ser de provecho para el experto es que la herramienta recuerde la estructura del modelo para que, en cambios sucesivos en el tiempo, la propia herramienta pueda sugerir llevar a cabo las modificaciones necesarias para mantener la estructura del modelo lo más parecida posible a la estructura original y evitar que esta sea deformada conforme pase el tiempo y se vayan realizando modificaciones.

Por último, como se mencionó en la sección 2.1, un dato puede ser incluido en el modelo sólo si tiene una clase asociada y esta clase bien pudiera estar establecida por el usuario o bien ser la predicha por el modelo con el consentimiento del usuario. La herramienta puede recordar que tal dato fue clasificado por el propio modelo y así, en futuras decisiones a tomar el experto podrá saber si los datos que intervienen en una superposición a eliminar fueron clasificados por el propio modelo o por el usuario. Esta característica puede resultar muy útil cuando ocurren las diferentes modificaciones presentadas en la sección 2. Por ejemplo, en la sección 2.1.3 cuando se busca un hiper-rectángulo para aumentar su volumen, puede considerarse si producto de este aumento se incluyen datos de otra clase y al mismo tiempo la clase de dichos datos fueron predicciones del propio modelo. En estos casos es posible plantearse si la predicción realizada en su momento pudo deberse a una falta de información en el modelo y, al mismo tiempo plantear, un posible cambio de clase para estos datos.

5. Análisis de rendimiento

A medida que la tecnología fue avanzando y la capacidad de almacenamiento para los datos aumentaba, se comenzó a estudiar la posibilidad de introducir nuevas técnicas que sean capaces de adaptar los modelos de datos construidos, sin necesidad de llevar a cabo un re-entrenamiento completo del modelo.

Desde la aparición del algoritmo ID4 (Schlimmer, y otros, 1986) en 1986 varios autores han presentado diferentes técnicas para resolver el problema de la minería de datos incremental (Chao, y otros, 2009) (Qingyun, 2011) (Shaorong, y otros, 2010) (Utgoff, 1994) (Utgoff, 1988) (Utgoff, 1989) (Utgoff, y otros, 1996) (Piao, y otros, 2010), la mayoría de ellos basados en árboles de decisión.

Si bien es muy difícil de poder comparar el esfuerzo computacional de adaptar el modelo de CLUHR comparado contra otros modelos adaptativos, en esta sección se hace un breve repaso sobre algunas de las técnicas aparecidas recientemente para llegar a la conclusión de que CLUHR podría ser más eficiente que las demás técnicas en la tarea de adaptar el modelo.

La técnica presentada en (Chao, y otros, 2009) está basado en el algoritmo ID3 (Quinlan, 1986), presentando el armado inicial del árbol usando el algoritmo ID3 con pequeñas modificaciones mientras que la tarea de adaptar el modelo incluye las siguientes tareas:

- Se actualiza la información de cada nodo, por la cual pasa el nuevo dato presentado, hasta llegar a un nodo hoja.
- Si el dato no pertenece a la misma clase que la hoja entonces se divide el nodo hoja generando un nuevo sub-árbol
- Se evalúa la función discriminante en cada nodo para acomodar la información del nuevo dato, si algún nodo no satisface el criterio de soporte, a partir de ese nodo se crea un nuevo sub-árbol.

En (Piao, y otros, 2010) presentan una técnica que construye un árbol de la misma manera que en (Utgoff, y otros, 1996) con el agregado de un "ensemble" de árboles como presentan en (Li, y otros, 2003). Si bien la precisión lograda por esta estrategia es alta, ya que posee todo un "comité" de árboles de decisión, el mantenimiento es muy costoso ya que se deben mantener *k* árboles de decisión. El algoritmo ITI presentado en (Utgoff, y otros, 1996) presenta las mismas tareas que la técnica enterior, es decir: actualizar los nodos por los que pasa un dato, dividir el nodo si el dato no es de la misma clase y evaluar la función discriminante de cada nodo, lo que ocasionaría una re-estructuración de un sub-árbol.

La técnica presentada en (Qingyun, 2011) es similar a las anteriores con la diferencia que la re-estructuración de un sub-árbol es más eficiente ya que cada nodo guarda una tabla de la frecuencia de valores que poseen los datos que cubre dicho nodo. Las principales dos desventajas de esta técnica son: cada nodo debe actualizar sus correspondientes tablas por cada nuevo dato que se presenta al árbol, y si un valor determinado no está contemplado en una tabla entonces este nodo y todos sus descendientes debe ser re-estructurados.

En (Shaorong, y otros, 2010) presentan una novedosa técnica para minería incremental que no incluye al árbol como estructura de datos. Trabajan con conjuntos de reglas y para cada nuevo dato que se incorpora al modelo, esta técnica evalúa el soporte de cada regla para determinar si eliminar o agregar reglas a los respectivos conjuntos. Si bien esta técnica es muy eficiente al agregar nuevos datos, ya que solo re-evalúa el soporte de las reglas, presenta como desventajas que solo trabaja con problemas de dos clases y además la técnica por si sola no descubre nuevas reglas, estas deben ser establecidas por un humano.

CLUHR al agregar un nuevo dato solo ajusta el hiper-rectángulo en el cual cae el dato, donde solo es necesario evaluar este hiper-rectángulo, en el caso de decidir modificar el hiper-rectángulo solo es necesario evaluar las superposiciones que presenta este con otros hiper-rectángulos.

Ante lo discutido en esta sección es posible determinar las siguientes ventajas y desventajas de CLUHR frente a otras estrategias, cuando un nuevo dato es presentado al modelo armado. En el caso de los árboles de decisión,

se deba hallar el nodo hoja que representa al dato, mientras que en CLUHR se hace lo mismo con los hiper-rectángulos.

5.1. Costo en hallar el hiper-rectángulo (u hoja)

En este aspecto los árboles resultan más eficientes que CLUHR en el hecho que alcanzar una hoja es de orden $O(\log_2 n)$ mientras que en CLUHR, al no tener una estructura de hiper-rectángulo, la búsqueda es lineal y por lo tanto de $O(n)$.

Aunque vale notar que rara vez un modelo en CLUHR tendrá más de 50 hiper-rectángulos (recordar que cada hiper-rectángulo equivale a una regla), por lo que buscar de manera lineal en un conjunto de 50 hiper-rectángulos para las computadoras de hoy en día no representan una carga extremadamente alta.

5.2. Re-estructuración del hiper-rectángulo (u hoja)

Si el dato que se presenta al modelo cae en un hiper-rectángulo (u hoja) donde la clase representada por el hiper-rectángulo (u hoja) es distitna a la clase del dato entonces ocurre lo siguiente:

- En los árboles se llevan a cabo las tareas mencionadas anteriormente. En el peor de los casos se debe re-construir todo un sub-árbol, el cual representa a una mayor parte de la base de datos cuanto más cerca está el nodo a re-estructurar del nodo raíz del árbol. De hecho, si el nodo a re-estructurar es el nodo raíz, entonces el árbol debe ser construido nuevamente.

- Por su parte, CLUHR solo divide el hiper-rectángulo en el cual cae el dato y analiza posibles superposiciones del nuevo hiper-rectángulo creado contra aquellos que presente superposición.

Los nodos hoja de un árbol de decisión pueden verse (salvando las diferencias) como los hiper-rectángulos de CLUHR en el sentido que ambos abarcan un área hiper-rectangular, por lo que un árbol con 16 nodos hojas puede verse como un modelo de CLUHR con 16 hiper-rectángulos.

Si un sub-árbol que contiene 16 nodos hojas debe ser re-construido, entonces implica recorrer los datos que eran representados por dicho sub-árbol para llevar a cabo la re-estructuración. Mientras que en CLUHR eso no ocurre nunca. En el peor de los casos (y es un caso hipotético e improbable), el hiper-rectángulo que se genera con la llegada del nuevo dato se intersecta contra otros 15 hiper-rectángulos (de clases distintas), por lo que el costo computacional es igual que el caso planteado para el árbol.

En casos más frecuentes un árbol necesita hacer re-estructuraciones esporádicas y estás siempre involucran sub-árboles de no más de cuatro o seis nodos hojas, mientras que CLUHR por lo general siempre chequea superposiciones con tres o cuatro hiper-rectángulos.

Cómo se verá en la sección 4 del capítulo 4, comparando el esfuerzo computacional para llevar a cabo sucesivas agregados de datos, CLUHR supera ampliamente a la técnica ITI (Utgoff, y otros, 1996).

5.3. Conclusiones

No es posible determinar que CLUHR requiera menos costo computacional que el resto de las técnicas aunque por lo planteado anteriormente parecería llevar una pequeña ventaja.

La técnica presentada en (Chao, y otros, 2009) utiliza una variante del algoritmo ITI para llevar a cabo la tarea de re-estructuración. En (Qingyun, 2011) cada nodo del árbol guarda una tabla de frecuencia de valores que debe ser actualizada con cada dato, incluso si este dato resulta ser de la misma clase que el nodo hoja alcanzado. En este sentido, la técnica presentada en (Qingyun, 2011) presenta las mismas desventajas que el algoritmo ITI.

En este sentido CLUHR resulta mejor que estas técnicas ya que, no mantiene estructuras auxiliares que deban ser actualizadas y por los resultados presentados en la sección 4 del capítulo 4, CLUHR requiere menos esfuerzo computacional que la técnica ITI.

La comparación contra el trabajo de (Piao, y otros, 2010) resulta ser favorable para CLUHR en el sentido que en (Piao, y otros, 2010) utilizan un "ensemble" de *k* árboles y por lo tanto, la frecuencia de re-estructuración en uno de ellos es más alta.

Por último, el trabajo de (Shaorong, y otros, 2010) resulta mucho más eficiente que CLUHR ya que trabaja directamente con el soporte de las reglas y no requiere que la base de datos sea recorrida para llevar a cabo una re-estructuración. Las desventajas de la técnica presentada en (Shaorong, y otros, 2010) es que solo es posible utilizarla en problemas de dos clases y además, la propia técnica no introduce nuevas reglas, siendo un humano el encargado de esta tarea.

Por lo comentado en estos últimos párrafos, ante la llegada de nuevos datos, CLUHR supera a las técnicas basadas en árboles de decisión.

Las ventajas adicionales que presenta CLUHR ante la modificación de la base de datos, son los demás tipos de re-estructuracion presentados en este capítulo: modificación de un dato, eliminación de un dato y sub-clasificación. Estos tipos de modificaciones en una base de datos no han sido contemplados por el resto de las técnicas estudiadas.

Capítulo 4

Resultados y comparaciones

Todas las verdades son fáciles de entender, una vez descubiertas. El caso es descubrirlas.
(Galileo Galilei)

En este capítulo se realiza un estudio detallado del funcionamiento de la técnica propuesta en esta tesis en cuanto el armado del modelo de datos. De igual modo se lleva a cabo ensayos sobre bases de datos reales y se compara los resultados obtenidos con otros métodos de clasificación que pueden ser encontrados en la literatura.

En primer lugar, se analizan distintos problemas con bases de datos de dos dimensiones. Estos problemas permiten observar en detalle el funcionamiento de la estrategia propuesta según la distribución en el espacio de los datos usados para el armado del modelo. Estos problemas usados como ejemplos son bases de datos armadas artificialmente, las cuales permiten observar en una figura la distribución de los datos en el espacio. También se analizan una serie de problemas donde están presentes dos, tres y hasta cuatro clases de datos, distintas distribuciones en el espacio y problemas de fácil resolución y problemas donde son necesarias varias divisiones de hiper-rectángulos para el armado del modelo de datos. Al mismo tiempo se lleva a cabo un análisis de cuántos recursos son consumidos por la estrategia para la resolución del armado de un modelo. El recurso que se mide es la cantidad de veces que se recorre la base de datos durante el proceso de armado del modelo.

En una segunda etapa se realizan pruebas con bases de datos descargadas del repositorio UCI, las cuales presentan problemas en espacios de más de dos dimensiones. Por otra parte, son comparados los resultados obtenidos con otras estrategias de clasificación. Además, se hace una comparación estimada de cuántos recursos son consumidos por la estrategia y cuántos por cada uno de los métodos contra los cuales se hace la comparación.

1. Ejemplos ficticios en 2D

En esta sección, se estudia en detalle el funcionamiento de la estrategia utilizando problemas en el espacio de dos dimensiones. Este tipo de problemas resultan prácticos para observar la distribución de los datos de entrada, cómo quedan conformados los hiper-rectángulos que resultan del modelo de datos y cómo abarcan el espacio las reglas extraídas del modelo de datos.

El objetivo de estos ensayos es el de poder observar cómo la técnica resuelve distintos problemas según la distribución que presenten los datos. En estos ensayos no se dividen los datos de entrada en conjunto de datos de entrenamiento y conjunto de datos de testeo. Por lo tanto, la precisión del modelo se mide utilizando los

propios datos usados para el armado del modelo. Por esto, en la mayoría de los problemas estudiados se logra una precisión igual a uno.

El ensayo es realizado de esta manera ya que, como se comentó antes, el objetivo de estos ensayos es el de observar cómo la técnica presentada en esta tesis resuelve los distintos problemas presentados y cómo va eliminando las superposiciones que se presentan durante el armado del modelo. En la próxima sección se analizan problemas con bases de datos reales, separando estas en conjuntos de entrenamiento y de testeo.

Este estudio consta de 11 ejemplos que fueron armados artificialmente para presentar distintos problemas a ser resueltos por la estrategia propuesta y ver cómo CLUHR se comporta en cada uno de los problemas. Estos problemas están conformados por la cantidad de clases distintas, la distribución que tienen los datos en el espacio y el grado de superposición que presentan cada uno de éstos.

Los ejemplos que se compilan en esta sección presentan distintos grados de dificultad para el armado del modelo. Se analiza desde un caso fácil de resolver donde dos clases no presentan superposición alguna en el espacio de datos, hasta un caso donde la "mezcla" en el espacio de los datos de dos clases es tal que prácticamente se genera un hiper-rectángulo por cada dato presente.

Por cada ejemplo se presenta una figura con la distribución de los datos en el espacio y una figura con los hiper-rectángulos formados una vez que ha finalizado el armado del modelo. En la misma figura, también se muestra como queda representado el espacio de los datos por cada una de las reglas extraídas.

Del mismo modo, para cada ejemplo se realiza un estudio detallado sobre los recursos consumidos por la estrategia, la matriz de confusión que detalla la precisión del modelo de datos cuando la precisión es distinta de uno y el conjunto de reglas simplificadas que se extrae del modelo de datos.

Hacia el final de la sección, en una tabla resumen, se visualiza la cantidad de veces que se recorrió la base de datos, la cantidad de hiper-rectángulos del modelo armado, la cantidad de reglas extraídas, el promedio de cláusulas por regla, la cantidad de clases y la cantidad de superposiciones que fueron analizadas con los índices Ω.

1.1. Configuración de la estrategia

Para llevar a cabo todos los estudios que pertenecen a este capítulo, la estrategia fue implementada tal y como se describió en el capítulo 2. Para la tarea de eliminar las superposiciones se lleva a cabo el cálculo del índice Ω usando los seis índices Z y sus respectivos pesos vistos en la sección 3 del capítulo 2.

El criterio para la división de hiper-rectángulos se llevó a cabo con las siguientes consideraciones:

- Si en una intersección *S* un hiper-rectángulo *H* no presenta datos en *S*, entonces se divide *H* según la dimensión determinada por el índice Ω más alto, dejando al otro hiper-rectángulo sin modificar (ver secciones 2.2.1 y 2.2.2 del capítulo 2).

- Si en una intersección *S* hay datos presentes de ambos hiper-rectángulos, entonces la división no se hace en favor de ninguna de las clases y se divide de manera arbitraria ambos hiper-rectángulos por el punto medio de la intersección.

El aspecto único que se modifica en cada uno de los casos estudiados es el parámetro μ del algoritmo de CLUHR. Este parámetro siempre vale cero (son eliminadas todas las superposiciones) a menos que se indique lo contrario.

La simplificación de reglas se realizó utilizando el método greedy descripto en la sección 5.1 del capítulo 2, uniendo mediante el operador lógico OR en una misma regla aquellas que tienen una única cláusula.

Finalizado el armado del modelo, los hiper-rectángulos que tienen menos datos que el indicado por el parámetro μ se eliminan a los efectos de reducir la cantidad de reglas extraídas. Y para los casos donde μ es

mayor que uno, y aún existen superposiciones de datos, se eliminan de dicha superposición los hiper-rectángulos con menor cantidad de datos en proporción a la cantidad de datos de su clase.

Para medir la exactitud del modelo sobre aquellos casos donde quedan datos sin hiper-rectángulos representantes se utilizó la Ecuación 2-2 para predecir la clase del dato.

1.2. Dos clases separadas

1.2.1. Descripción del ejemplo

En este primer ejemplo de prueba se creó un conjunto de datos donde hay dos clases presentes, 33 datos de la clase Cuadrado y 24 datos de la clase Círculo. Como se puede observar en la Figura 4-1, las dos clases presentan en su distribución una gran separación en el espacio de los datos, lo que le permite a la estrategia hallar para cada clase su correspondiente hiper-rectángulo representativo sin presentar ninguna superposición.

1.2.2. Resultado

Este caso es el más simple de resolver, ya que no requiere ninguna intervención de la estrategia en cuanto a eliminar superposiciones de hiper-rectángulos de distintas clases. Luego de formar los hiper-rectángulos representativos mínimos, se detecta que no hay superposición y, de esta manera, finaliza el armado del modelo y se pasa a la etapa de extracción de reglas.

En este caso, el modelo de datos logra una exactitud igual a uno con respecto a los propios datos utilizados para el armado. Así, cada clase queda representada por un único hiper-rectángulo y, por lo tanto, por una única regla. A su vez, por el método de simplificación de reglas, cada una de éstas queda con una sola cláusula que hace referencia a un valor del eje X.

Las reglas simplificadas extraídas del modelo de datos son las siguientes:

IF (X ≤ 212) THEN Cuadrado

IF (X ≥ 326) THEN Círculo

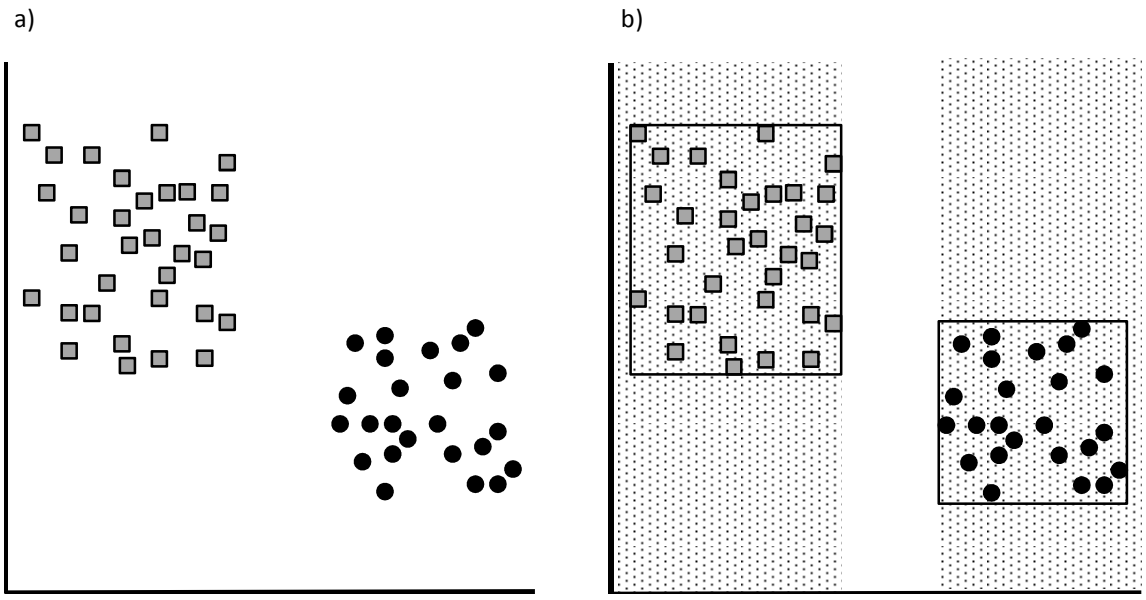

Figura 4-1. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.3. Una clase entremedio de otra

1.3.1. Descripción del ejemplo

En este ejemplo, la distribución de la clase Cuadrado (33 datos) a lo largo del espacio de los datos logra separar dos cúmulos de datos de la clase Círculo (24 datos) (Figura 4-2). En este ejemplo, ya no es posible la separación directa entre clases, dado que los hiper-rectángulos representativos mínimos iniciales de ambas clases presentan una superposición total.

1.3.2. Resultado

En este ejemplo, la estrategia tiene una primera superposición para eliminar. Como CLUHR es incapaz de detectar que un cúmulo alargado de datos de una clase separa dos cúmulos de la otra clase, se procede a calcular los índices Ω. Los resultados de calcular los índices Ω determinan la división de ambos hiper-rectángulos por un valor del eje Y. Luego de los reajustes de los hiper-rectángulos representativos mínimos se forman cuatro hiper-rectángulos (dos para clase) eliminando así las superposiciones de ambas clases.

En este mismo ejemplo, la precisión lograda también resulta ser igual a uno con respecto a los propios datos utilizados para el armado del modelo de datos.

Por otro lado, a cada una de las cuatro reglas extraídas se les pueden eliminar dos cláusulas, quedando las siguientes cuatro reglas simplificadas:

IF (X ≤ 208) AND (Y ≥ 222) THEN Cuadrado

IF (X ≥ 171) AND (Y ≤199) THEN Cuadrado

IF (X ≤111) AND (Y ≤ 116) THEN Círculo

IF (X ≥281) AND (Y ≥ 213) THEN Círculo

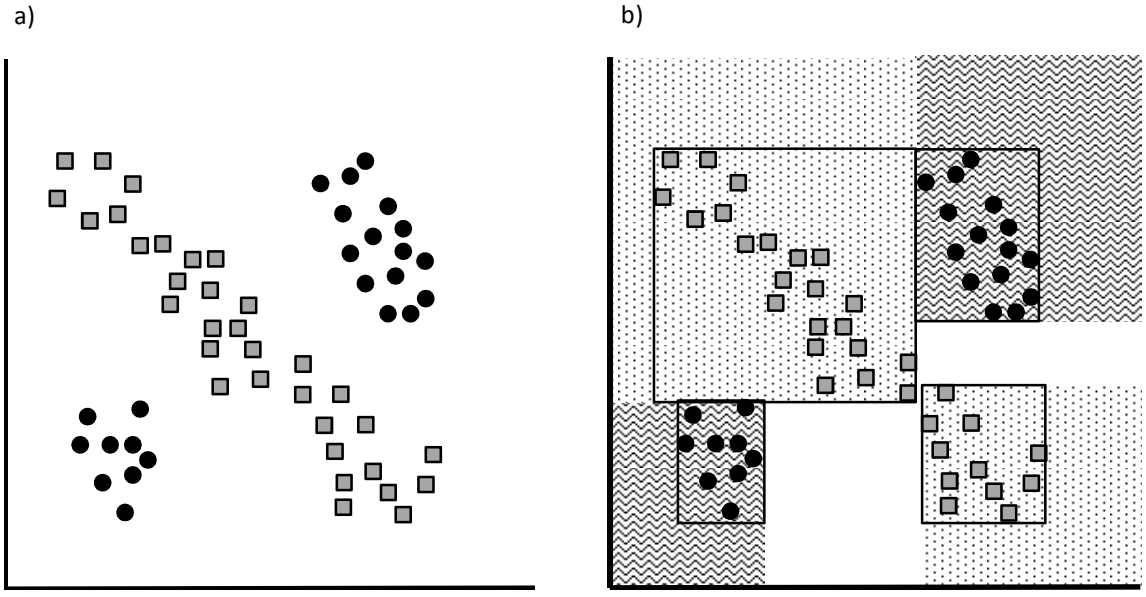

Figura 4-2. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.4. Una clase envolviendo parcialmente a otras dos

1.4.1. Descripción del ejemplo

En este caso se puede observar que la distribución de los 65 datos de la clase Cuadrado "envuelve" a los datos de las clases Círculo (9 datos) y la clase Triángulo (15 datos). En este ejemplo, el hiper-rectángulo de la clase Cuadrado, cuya distribución de los datos es la que más espacio abarca, presenta una superposición con los hiper-rectángulos de cada una de las otras dos clases (Figura 4-3). Mientras que los hiper-rectángulos representativos mínimos de las clases Círculo y Triángulo no presentan superposición entre sí.

1.4.2. Resultado

La estrategia, al comienzo, analiza las dos superposiciones iniciales y por los resultados obtenidos de los índices Ω resulta que la superposición a eliminar es la que presentan la clase Cuadrado y la clase Triángulo. Al realizar la división del hiper-rectángulo de la clase Cuadrado, se elimina la superposición con la clase Triángulo. De los dos nuevos hiper-rectángulos que son generados, uno de ellos presenta una superposición con la clase Círculo. En una segunda división del hiper-rectángulo de la clase Cuadrado, se elimina la superposición entre éste y el hiper-rectángulo de la clase Círculo, formando así tres hiper-rectángulos para la clase cuadrado.

Así, se observa que con dos divisiones del hiper-rectángulo de la clase Cuadrado, se obtiene un modelo de datos cuya precisión es igual a uno para los datos utilizados para el armado. De este ejemplo, se extraen cuatro reglas. Las tres reglas de la clase Cuadrado pueden ser unificadas en una única regla mediante el operador lógico OR. De esta manera las reglas, quedan conformadas de la siguiente manera:

IF (X ≤ 191) OR (Y ≥ 340) OR (Y ≥ 278) THEN Cuadrado

IF (X ≥ 213) AND (X ≤ 285) AND (Y ≤ 215) THEN Círculo

IF (X ≥ 346) AND (Y ≤ 137) THEN Triángulo

Notar que de la regla de la clase Cuadrado, la cláusula (Y ≥ 340) puede ser eliminada por ser redundante con la cláusula (Y ≥ 278).

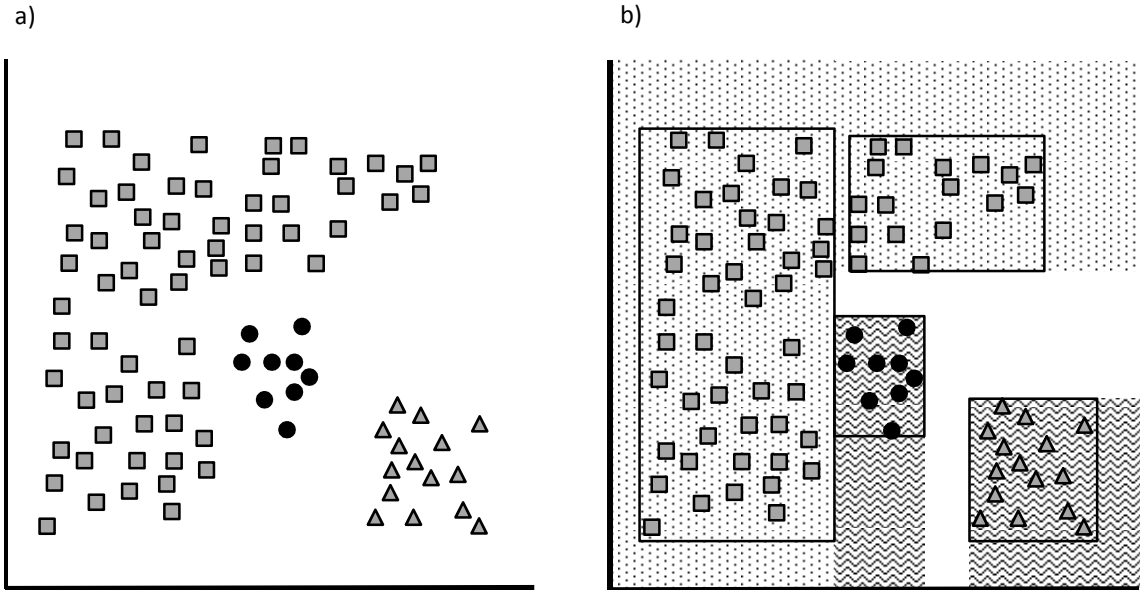

Figura 4-3. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.5. Envolturas sucesivas

1.5.1. Descripción del ejemplo

Este ejemplo presenta una distribución de los datos similar al anterior. La base de datos está formada por cuatro clases, donde la distribución de los datos de una clase "envuelve" a la otra. De esta manera, la distribución de los 49 datos de la clase Cuadrado envuelve a las otras tres clases, formando un hiper-rectángulo representativo mínimo inicial que presenta superposición con las tres clases. De manera similar, los 45 datos de la clase Círculo envuelven a los datos de las otras dos, incluyendo la superposición con los hiper-rectángulos representativos mínimos. Finalmente, los 30 datos de la clase Triángulo envuelven a los datos de la clase Cruz (12 datos). En el caso de esta última clase, también es incluida por el hiper-rectángulo representativo mínimo inicial de la clase Triángulo (Figura 4-4).

Este ejemplo muestra un caso con seis superposiciones iniciales entre las cuatro clases, por lo que el índice Ω en este tipo de problemas comienza a jugar un papel fundamental para lograr armar un modelo de datos óptimo.

1.5.2. Resultado

La estrategia comienza analizando las superposiciones iniciales y el cálculo de los índices Ω. Luego determina que la primera superposición a eliminar sea la que forman las clases Cuadrado y Cruz, lo que ocasiona la división del hiper-rectángulo de la clase Cuadrado. Uno de estos nuevos hiper-rectángulos presenta superposición con las clases Círculo y Triángulo. En una segunda etapa, se vuelve a dividir el hiper-rectángulo de la clase Cuadrado para eliminar la superposición entre éste y la clase Triángulo. Finalmente queda una nueva división del hiper-rectángulo de la clase Cuadrado para eliminar la superposición con la clase Círculo.

En una segunda etapa, se procede a eliminar la superposición entre las clases Círculo y Cruz con una división del hiper-rectángulo de la clase Círculo una primera vez. Luego una segunda división de este hiper-rectángulo elimina la superposición con la clase Triángulo. En la etapa final, se elimina la superposición entre las clases Triángulo y Cruz mediante la división del hiper-rectángulo de la clase Triángulo.

El modelo obtenido finaliza con una exactitud igual a uno para los datos utilizados en el armado. Como en el ejemplo anterior, las reglas de la clase Cuadrado se pueden unificar en una única regla. De esta manera el conjunto de reglas simplificadas quedan formadas así:

IF (X ≤ 139) OR (Y ≥ 387) OR (Y ≥ 397) OR (Y ≥ 399) THEN Cuadrado

IF (X ≥ 377) AND (Y≥ 239) AND (Y ≤ 348) THEN Círculo

IF (X ≥ 149) AND (X ≤ 259) AND (Y ≤ 343) THEN Círculo

IF (X ≥ 285) AND (Y ≥ 294) AND (Y ≤ 347) THEN Círculo

IF (X ≥ 373) AND (Y ≥ 215) AND (Y ≤ 257) THEN Triángulo

IF (X ≥260) AND (X ≤360) AND (Y ≤ 257) THEN Triángulo

IF (X ≥ 364) AND (Y ≤ 162) THEN Cruz

Notar que como sucede en el ejemplo anterior, una cláusula de la regla de la clase Cuadrado puede ser eliminada por presentar redundancia a la regla. Y en la clase Círculo, también es posible unificar dos reglas en una, ya que entre ambas representan al mismo espacio.

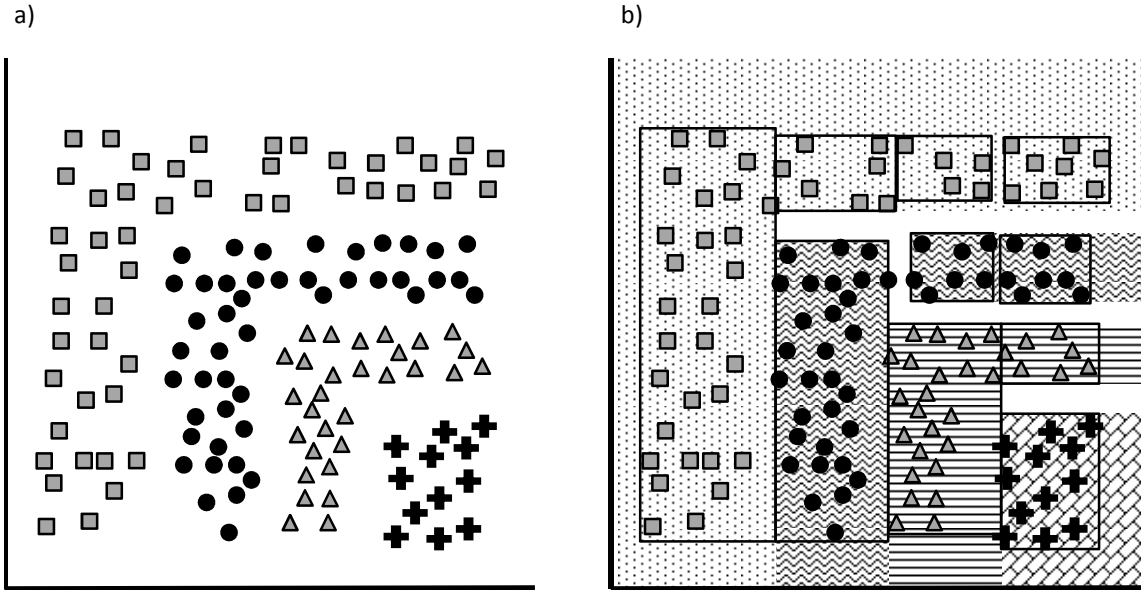

Figura 4-4. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.6. Tres clases con varias zonas de superposición

1.6.1. Descripción del ejemplo

Este ejemplo muestra tres clases cuyas distribuciones de los datos se entremezclan en el espacio unas con otras. La clase Cuadrado con 52 datos "envuelve" a las otras dos clases en el espacio y, al mismo tiempo, forma un hiper-rectángulo representativo mínimo inicial que incluye completamente a los hiper-rectángulos de las otras dos clases. Entre las clases Círculo (45 datos) y la clase Triángulo (32 datos) se presenta una superposición parcial en sus respectivos hiper-rectángulos representativos mínimos iniciales y una gran mezcla en el espacio de los datos (Figura 4-5).

1.6.2. Resultado

Este ejemplo presenta varias complicaciones al momento de dividir los hiper-rectángulos dada la distribución de los datos de las clases Círculo y Triángulo. La clase Cuadrado es relativamente fácil de resolver, ya que con su división se eliminan las superposiciones con las otras dos clases. Entre las clases Círculo y Triángulo se deben realizar varias divisiones hasta lograr un modelo de datos aceptable.

Si el parámetro μ se establece en cero, entonces la división de los distintos hiper-rectángulos se realiza hasta que no haya superposiciones, formando 15 hiper-rectángulos para las clases Círculo y Triángulo, donde cinco hiper-rectángulos de la clase Círculo representan a menos de cuatro datos y dos de la clase Triángulo representan a dos datos cada uno. Si bien este modelo tiene una exactitud igual a uno, midiéndolo con los datos usados para el armado, tener 20 reglas (incluidas las cinco de la clase Cuadrado) con siete reglas que solo representan a 13 datos en total resulta en un modelo bastante ineficiente.

En la Figura 4-5 se observa el resultado de usar la estrategia propuesta con un valor de cuatro para el parámetro μ. De esa manera, el modelo presenta falsos positivos al momento de medir su exactitud pero tiene la ventaja de extraer solo ocho reglas. En el modelo de datos final, queda una superposición entre las clases Triángulo y Círculo, donde el hiper-rectángulo de la clase Círculo es eliminado por tener menos datos que el otro hiper-rectángulo.

La precisión del modelo de datos usando el valor cuatro para el parámetro μ es de 0,9380 y la matriz de confusión correspondiente es la siguiente:

	Cuadrado	Círculo	Triángulo
Cuadrado	52	0	0
Círculo	0	42	3
Triángulo	0	2	30

El conjunto de reglas simplificadas quedan formadas de la siguiente manera:

IF (X ≤ 62) OR (X ≤ 76) OR (X ≥ 485) OR (X ≥ 487) OR (Y ≥ 349) THEN Cuadrado

IF (X ≥ 113) AND (X ≤ 177) AND (Y ≥ 151) AND (Y ≤ 334) THEN Círculo

IF (X ≥ 282) AND (X ≤ 452) AND (Y ≥ 209) AND (Y ≤ 346) THEN Círculo

IF (X ≥ 199) AND (X ≤ 256) AND (Y ≥ 290) AND (Y ≤ 341) THEN Círculo

IF (X ≥ 210) AND (X ≤ 279) AND (Y ≤ 237) THEN Triángulo

IF (X ≥ 310) AND (X ≤ 440) AND (Y ≤ 201) THEN Triángulo

IF (X ≥ 138) AND (X ≤ 169) AND (Y ≤ 109) THEN Triángulo

Hay que notar que las dos reglas de la clase Círculo no pueden ser simplificadas por quedar "encerradas" dentro del espacio de los datos y, por lo tanto, ambas quedan con cuatro cláusulas. Mientras que, de la regla de la clase Cuadrado, pueden excluirse dos cláusulas ya que proveen información redundante. Por ejemplo, la cláusula (X ≤ 76) abarca la cláusula (X ≤ 62).

a) b)

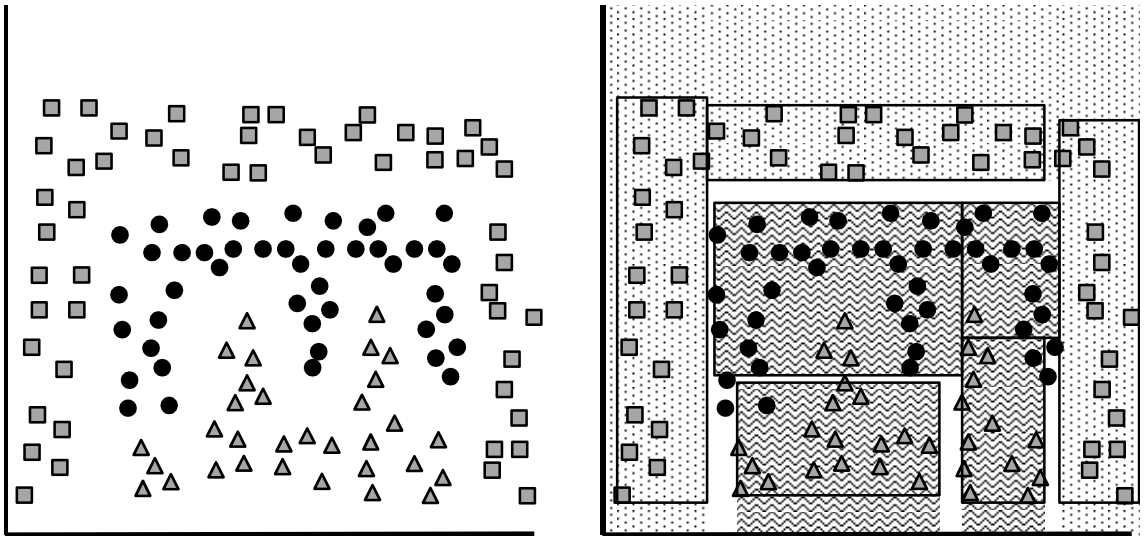

Figura 4-5. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.7. Doble espiral

1.7.1. Descripción del ejemplo

Este ejemplo consiste en dos clases, Cuadrado (70 datos) y Círculo (61 datos), cuya distribución en el espacio forman una espiral dentro de otra (Figura 4-6). Al igual que el ejemplo anterior este problema también presenta dificultades al tratar de eliminar las superposiciones presentes.

1.7.2. Resultado

En un caso donde la distribución de los datos tiene forma de espiral es difícil encontrar reglas que las expliquen de manera clara, más si éstas solo pueden hacer referencia a dos valores de cada una de las dimensiones del espacio. Una posible solución podría ser aquella en que se generen pequeños hiper-rectángulos de tal forma que no presenten superposición unos con otros, pero la gran desventaja de esta solución es que el número de reglas extraídas puede resultar elevado.

En este ejemplo, si se usa un valor cero para el parámetro μ, se generan 16 hiper-rectángulos donde cinco de ellos solo representan a tres datos o menos. Usando un valor de seis para μ se generan 10 hiper-rectángulos que, si bien resulta en un número importante de reglas para un problema de dos clases, es un resultado aceptable. Sobre todo si se tiene en cuenta la naturaleza de la distribución de los datos, la cual impide obtener mejores resultados.

La precisión del modelo de datos usando el valor seis para el parámetro μ es de 0,9237 y la matriz de confusión correspondiente es la siguiente:

	Cuadrado	Círculo
Cuadrado	69	1
Círculo	9	52

Las reglas simplificadas para este ejemplo quedan formadas de la siguiente manera:

IF (Y ≥ 445) OR (Y ≥ 437) THEN Cuadrado

IF (X ≥ 395) AND (Y ≥ 313) THEN Cuadrado

IF (X ≥ 123) AND (X ≤ 226) AND (Y ≥ 46) AND (Y ≤ 334) THEN Cuadrado

IF (X ≥ 243) AND (Y ≥ 230) AND (Y ≤ 303) THEN Cuadrado

IF (X ≥ 242) AND (Y ≤ 135) THEN Cuadrado

IF (X ≥ 262) AND (X ≤ 394) AND (Y ≥ 309) AND (Y ≤ 426) THEN Círculo

IF (X ≤ 231) AND (Y ≥ 344) AND (Y ≤ 417) THEN Círculo

IF (X ≤ 109) AND (Y ≤ 322) THEN Círculo

IF (X ≥ 246) AND (X ≤ 364) AND (Y ≥ 145) AND (Y ≤ 208) THEN Círculo

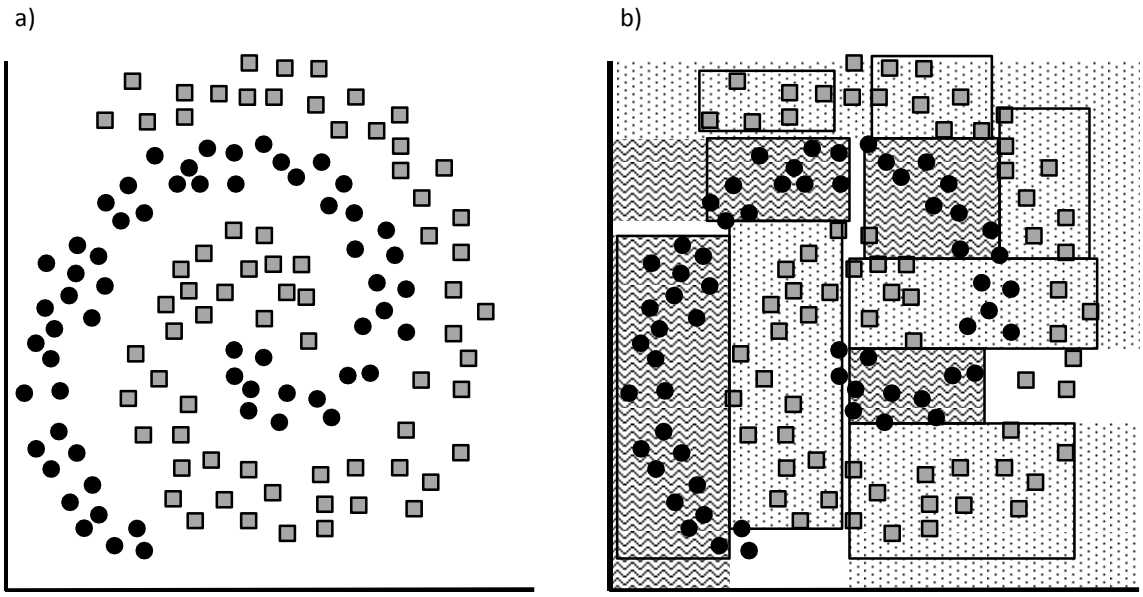

Figura 4-6. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.8. Una clase que encierra a otra

1.8.1. Descripción del ejemplo

Este ejemplo consiste en una clase Círculo con 29 datos, los cuales forman una distribución que se encuentra completamente encerrada por la distribución en forma de "anillo" de la clase Cuadrado (67 datos) (Figura 4-7). Ambas clases se encuentran separadas por una gran distancia, lo que permite una clara separación de los hiper-rectángulos.

1.8.2. Resultado

Este ejemplo es de fácil resolución ya que la primera superposición entre ambas clases se elimina dividiendo el hiper-rectángulo inicial de la clase Cuadrado en cuatro hiper-rectángulos.

La precisión del modelo lograda en relación a los datos usados en el armado igual a uno, y el conjunto de reglas simplificadas en este ejemplo queda formado así:

IF (X ≤213) OR (X ≥379) OR (Y ≤ 166) OR (Y ≥461) THEN Cuadrado

IF (X ≥ 216) AND (X ≤ 369) AND (Y≥228) AND (Y ≤363) THEN Círculo

1.9. Una clase que encierra a otra de manera más ajustada

1.9.1. Descripción del ejemplo

Este ejemplo es similar al anterior ya que se encuentran presentes las mismas clases con los mismos datos, pero la separación en el espacio entre las dos clases es mucho menor (Figura 4-8). En este ejemplo, ya no es posible describir con cinco hiper-rectángulos las dos clases manteniendo una precisión del modelo igual a uno.

1.9.2. Resultado

Usando el valor cero para el parámetro μ la estrategia genera 13 hiper-rectángulos, donde siete representan a cuatro datos o menos. Utilizando un valor de cinco para el parámetro μ la estrategia produce seis hiper-rectángulos con una precisión de 0,9375.

a) b)

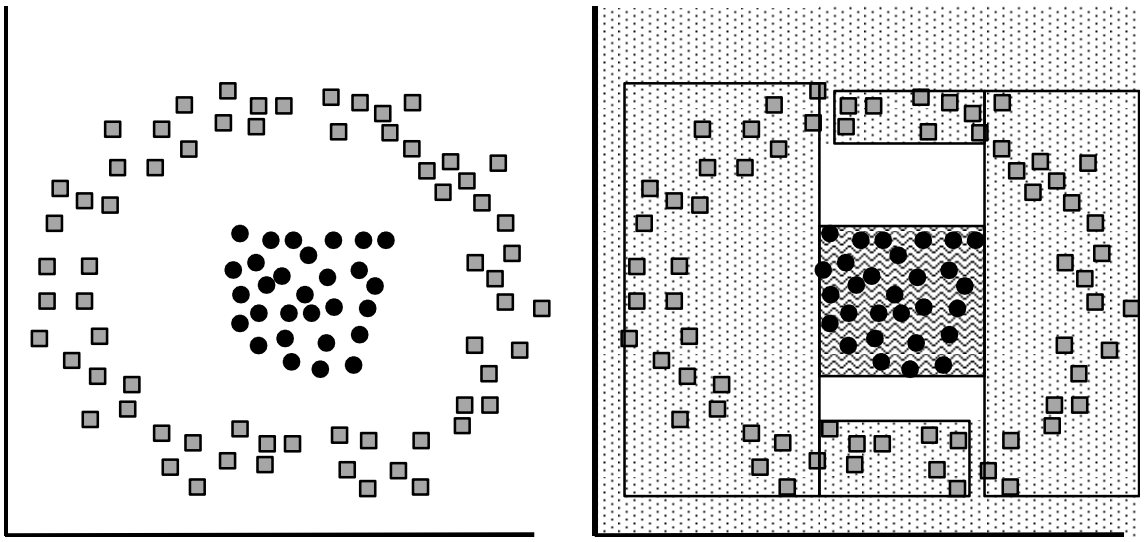

Figura 4-7. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

La matriz de confusión es la siguiente:

	Cuadrado	Círculo
Cuadrado	65	2
Círculo	4	25

El conjunto de reglas simplificadas para este ejemplo quedan formadas de la siguiente manera:

IF (X ≤97) OR (X ≥ 258) OR (Y ≤ 226) OR (Y ≥429) THEN Cuadrado

IF (X ≥ 106) AND (X ≤136) AND (Y ≥248) AND (Y ≤ 361) THEN Círculo

IF (X ≥ 138) AND (X ≤251) AND (Y ≥ 232) AND (Y ≤ 422) THEN Círculo

1.10. División en diagonal

1.10.1. Descripción del ejemplo

Este ejemplo presenta dos clases, Cuadrado con 53 datos y Círculo con 29 datos, las cuales presentan una distribución de los datos tal que se separan en el espacio claramente. Pero, la brecha que las separa es en diagonal con respecto a los ejes del propio espacio (Figura 4-9). Esto hace que los hiper-rectángulos de ambas clases deban ser divididos varias veces.

1.10.2. Resultado

Utilizando un valor de cero para el parámetro μ se generan ocho hiper-rectángulos, cuatro de ellos representando a menos de cuatro datos. Utilizando un valor de cuatro para el parámetro μ, la estrategia genera solo tres hiper-rectángulos y por lo tanto tres reglas.

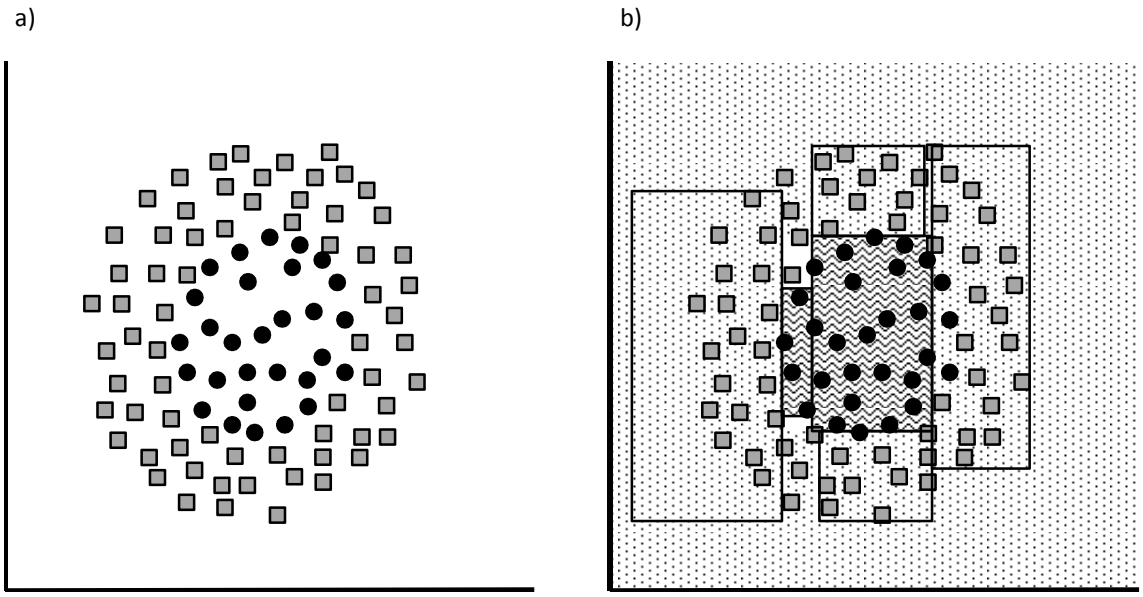

Figura 4-8. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

La precisión lograda en el modelo armado es del 95% y la correspondiente matriz de confusión es la siguiente:

	Cuadrado	Círculo
Cuadrado	51	2
Círculo	2	27

El conjunto de reglas simplificadas queda formado por las siguientes reglas:

IF (X ≤ 160) THEN Cuadrado

IF (X ≤ 310) AND (Y ≥ 368) THEN Cuadrado

IF (X ≥ 169) AND (Y ≤ 362) THEN Círculo

1.11. Dos clases compartiendo un sector del espacio

1.11.1. Descripción del ejemplo

Este ejemplo muestra dos clases, Cuadrado con 54 datos y Círculo con 71 datos, donde las distribuciones de los datos presentan un importante grado de mezcla (Figura 4-10). Es fácil ver que en este tipo de problemas es imposible que se generen pocos hiper-rectángulos, y que los que se formen no representen a datos de la otra clase.

En este tipo de problemas comienza a participar la naturaleza del problema y, si una de las clases es más importante que la otra, lo que determina que la división de hiper-rectángulos se realice en favor de una u otra clase.

1.11.2. Resultado

Si se utiliza el valor cero para el parámetro μ, se generan 20 hiper-rectángulos, donde solo cuatro de ellos representan un número importante de datos. Al utilizar el valor cuatro para el parámetro μ, se generan cuatro hiper-rectángulos, tres para la clase Cuadrado y uno para la clase Círculo.

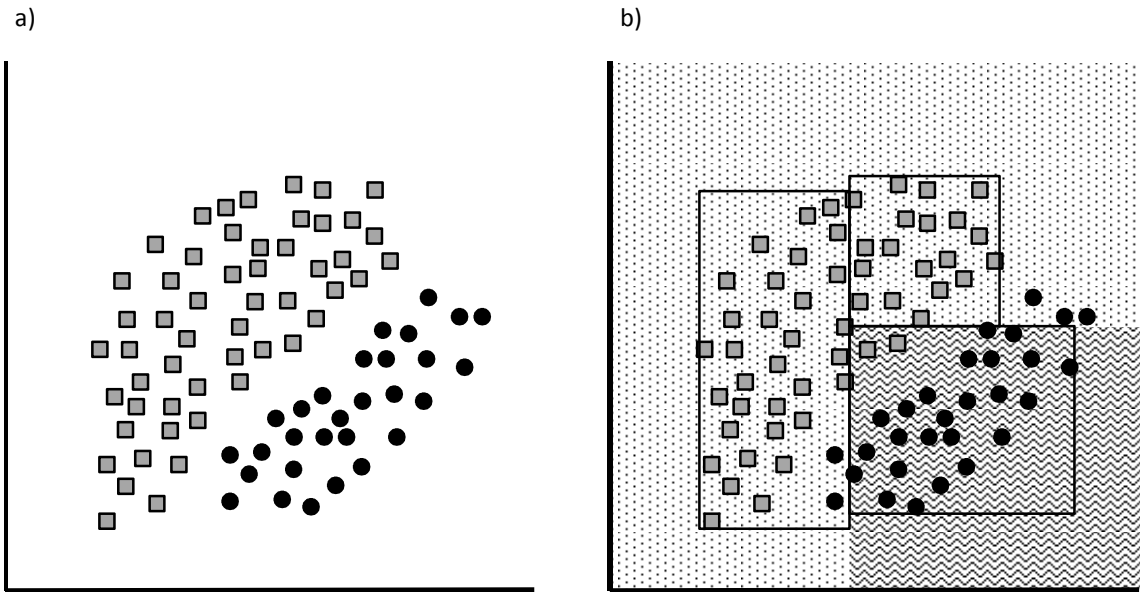

Figura 4-9. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

La precisión lograda por el modelo de datos en este ejemplo es de 0,9040 y la correspondiente matriz de confusión queda formada así:

	Cuadrado	Círculo
Cuadrado	52	2
Círculo	10	61

El conjunto de reglas simplificadas en este ejemplo quedan formadas de la siguiente manera:

IF (X ≤140) AND(Y ≥ 122) AND (Y ≤222) THEN Cuadrado

IF (X≥153) AND (X ≤ 161) AND (Y ≤ 252) THEN Cuadrado

IF (X ≤130) AND (Y ≥250) THEN Cuadrado

IF (X ≥ 164) THEN Círculo

1.12. Mezcla total de dos clases

1.12.1. Descripción del ejemplo

Este ejemplo presenta un problema que no puede ser resuelto de manera eficiente por CLUHR. En el problema se presentan dos clases, Cuadrado con 36 datos y Círculo también con 36 datos, donde se puede observar que su distribución es muy peculiar ya que los datos están agrupados de a pares en una grilla cuadrangular. En dicha grilla cada "celda" presenta dos datos de una clase y las celdas contiguas a cada celda, en las cuatro direcciones, presentan datos de la otra clase (Figura 4-11).

1.12.2. Resultado

El utilizar el valor cero para el parámetro μ causa que la estrategia consiga extraer 31 reglas. Casi el mismo número que celdas en la grilla en la cual están ubicados los datos.

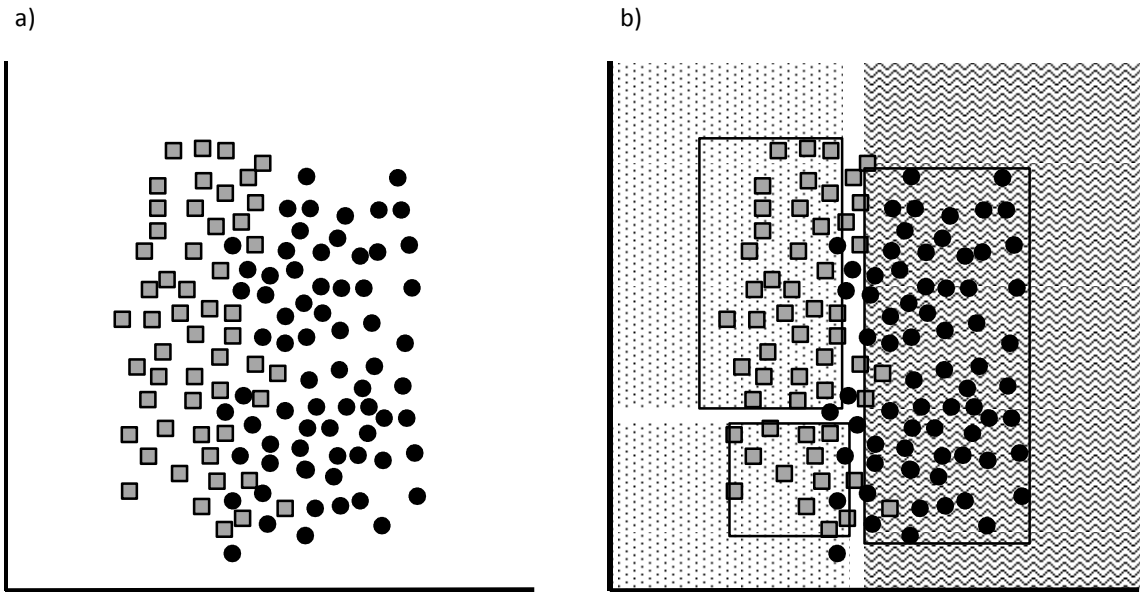

Figura 4-10. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

Usando el valor cuatro para el parámetro μ, se logra un modelo de datos con ocho reglas obteniendo una precisión de 0,6111 para los propios datos usados en el armado y generando la siguiente matriz de confusión:

	Cuadrado	Círculo
Cuadrado	24	12
Círculo	16	20

El conjunto de reglas simplificadas es el siguiente:

IF (X ≥ 367) AND (Y ≤ 457) THEN Cuadrado

IF (X ≤ 36) AND (Y ≥ 272) THEN Cuadrado

IF (X ≥ 252) AND (X ≤ 307) THEN Cuadrado

IF (X ≥ 169) AND (X ≤ 169) THEN Cuadrado

IF (X ≥ 38) AND (X ≤ 96) AND (Y ≤ 389) THEN Círculo

IF (X ≥ 325) AND (X ≤ 362) THEN Círculo

IF (X ≥ 171) AND (X ≤ 230) THEN Círculo

IF (X ≥ 162) AND (X ≤ 168) THEN Círculo

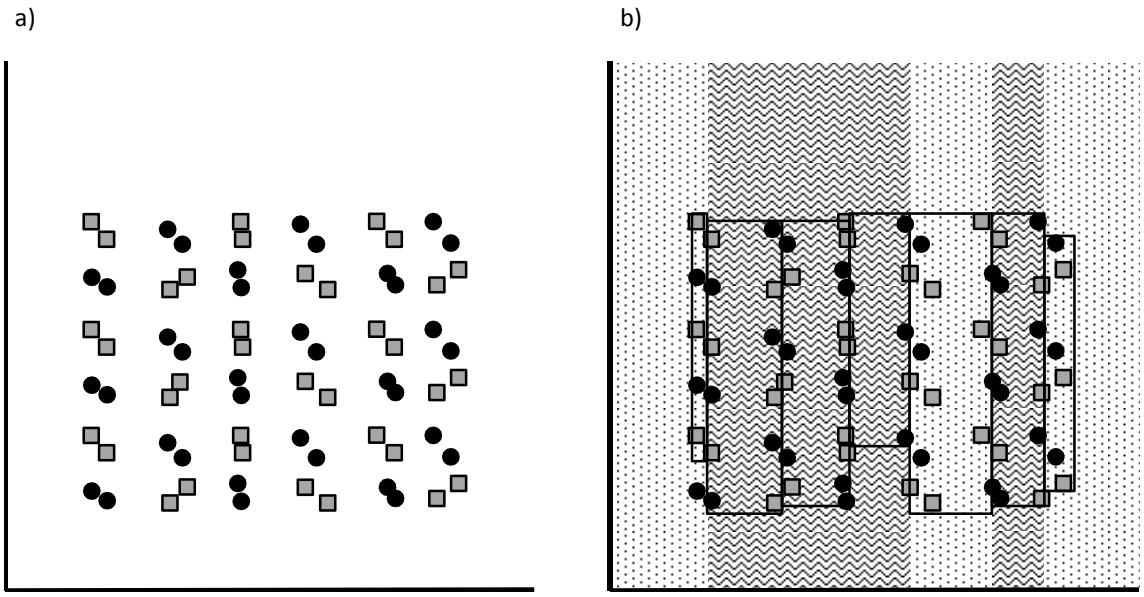

Figura 4-11. En a) se muestra la distribución de los datos. En b) se muestra cómo quedaron establecidos los hiper-rectángulos y las regiones del espacio que cubren las reglas correspondientes.

1.13. Resumen

La Tabla 4-1 muestra el resumen de los 11 ejemplos estudiados con anterioridad. En la tabla se puede ver para cada ejemplo la cantidad de veces que se recorre la base de datos para formar el modelo de datos, la cantidad de superposiciones que son analizadas y tratadas, el número de clases de datos en la base de datos, la cantidad de hiper-rectángulos con los que finaliza el modelo armado, el número de reglas simplificadas que son extraídas del modelo, el promedio de cláusulas por regla del conjunto simplificado de reglas y la precisión lograda por el modelo de datos. La precisión es medida utilizando como datos de prueba los mismos que se usaron durante el armado del modelo.

Tabla 4-1. Resumen de los recursos consumidos y resultados obtenidos por la estrategia ante cada uno de los ejemplos estudiados en dos dimensiones.

	Caso 1.2	Caso 1.3	Caso 1.4	Caso 1.5	Caso 1.6	Caso 1.7	Caso 1.8	Caso 1.9	Caso 1.10	Caso 1.11	Caso 1.12
Veces que se recorrió la base de datos	1	2	2,40	2,91	2,88	3,78	1,70	3,38	2,59	3,86	5,21
Superposiciones analizadas	0	1	3	11	6	7	1	4	2	5	11
Número de clases en la base de datos	2	2	3	4	3	2	2	2	2	2	2
Número de hiper-rectángulos del modelo	2	4	5	10	11	10	5	6	3	4	8
Número de reglas simplificadas extraídas	2	4	3	7	7	9	2	3	3	4	8
Promedio de cláusulas por regla	1	2	2,67	3	3,71	2,89	4	4	1,67	2,25	2,12
Exactitud del modelo de datos	1	1	1	1	0,9612	0,9237	1	0,9375	0,9512	0,9040	0,6111

2. Bases de datos del repositorio UCI

En esta sección, se lleva a cabo el armado de un modelo de datos utilizando 14 bases de datos del repositorio UCI (Frank, y otros, 2010). Estas bases de datos presentan una dimensionalidad alta en el espacio de entrada. Las bases de datos presentadas en estos ensayos fueron elegidas por cumplir con los requisitos impuestos por CLUHR, la cual consiste en la necesidad de que todos los atributos de la base de datos deben ser numéricos. Para cada una de las bases de datos se realiza el test conocido como 10-fold cross-validation (sección 2.4.1.2 del capítulo 1).

Por otra parte, se realiza una breve descripción de las bases de datos utilizadas y luego se presentan los resultados del test 10-fold cross-validation obtenidos para cada una de ellas. Como resultado, además de la precisión promedio obtenida, se presentan los promedios del número de hiper-rectángulos del modelo, del número de reglas extraídas, del promedio de cláusulas por regla, de la cantidad de superposiciones analizadas y de la cantidad de veces que se recorrió la base de datos para el armado del modelo.

Cada uno de estos test se ejecutó de manera independiente 10 veces, para poder tener un promedio y un desvío estándar de cada uno de los aspectos que se quiere medir: número de veces que se recorre la base de datos para armar el modelo de datos, cantidad de superposiciones analizadas y resueltas, cantidad de hiper-rectángulos con los que el modelo finaliza, cantidad de reglas simplificadas, promedio de cláusulas por regla y la exactitud del modelo de datos.

La configuración del funcionamiento de la estrategia para realizar estos ensayos son los mismos que se establecieron en la sección 1.1, excepto la decisión de eliminar todas las superposiciones que quedan al finalizar el armado del modelo. Como se verá más detalladamente en la sección 2.2, este cambio en el procedimiento de extracción de reglas se hace para poder realizar una mejor comparación con los resultados de la estrategia PSO/ACO2.

2.1. Bases de datos usadas

Se utilizan 14 bases de datos del repositorio UCI (Frank, y otros, 2010). Estas bases de datos tienen la particularidad de tener todos los atributos numéricos, algunos de los cuales se encuentran en el dominio de los números enteros y otros en el dominio de los números reales. Otra particularidad que presentan estas bases de datos es que las mismas no presentan datos faltantes.

Con estas condiciones, CLUHR es capaz de llevar a cabo el armado del modelo de datos sin ningún tipo de problemas.

2.1.1. Ecoli data set

Esta base de datos está conformada por datos provenientes del estudio de la célula biológica. Y se emplea para predecir el lugar donde puede ser encontrada una proteína dentro de la misma. Existen ocho clases de datos, donde cada una representa un lugar determinado dentro de una célula, desde el citoplasma hasta la membrana exterior. Así, se busca predecir, dados los resultados de una célula, en qué lugar puede ser encontrada una proteína.

Consta de 336 datos y siete atributos, dos de los cuales son binarios. Por lo comentado en la sección 4.7 del capítulo 2, los dos atributos binarios fueron excluidos de la base de datos llevando a cabo el armado y el testeo en un espacio de cinco dimensiones.

2.1.2. Glass data set

Esta base de datos contiene información de análisis de distintos tipos de vidrio. La misma que consiste en el porcentaje de presencia de ciertos óxidos, además de otra característica de los vidrios; el índice de refracción. El objetivo es predecir, dados el análisis de una muestra, a qué tipo de vidrio pertenece dicha muestra. La base de datos, además, contiene información de muestras de seis clases, 214 datos y nueve atributos.

2.1.3. Haberman's Survival data set

Esta base de datos contiene casos de estudio que se llevaron a cabo sobre pacientes que fueron sometidos a cirugía por padecer cáncer de mama. Mediante la edad del paciente al momento de la operación, año de la operación y número de nodos auxiliares positivos se busca predecir si el paciente vivió más de cinco años después de la operación o no. Consta de 306 datos y tres atributos.

2.1.4. Image segmentation data set

Esta base de datos está formada por información de matrices de píxeles de 3x3 que contienen diversos cálculos sobre el análisis de una imagen. Las imágenes corresponden a fotos al aire libre, habiendo siete tipos de imágenes distintas. Se busca predecir, dado un patrón de imagen de 3x3 píxeles a qué tipo de imagen corresponde.

Consta de 210 datos y 19 atributos, donde uno de ellos (REGION-PIXEL-COUNT) fue eliminado ya que presentaba el mismo dato en todas las muestras.

2.1.5. Ionosphere data set

Esta base de datos contiene señales de radar enviadas a la ionósfera en búsqueda de electrones libres. Se pretende predecir si en la ionósfera hay estructuras de electrones presentes o no.

Consta de 351 datos y 34 atributos, dos de los cuales fueron eliminados de la base de datos, uno por ser binario (ver sección 4.7 del capítulo 2) y otro por tener el mismo valor para todos los datos.

2.1.6. Iris data set

Esta base de datos contiene las medidas del ancho y el largo del pétalo y sépalo de distintas plantas iris. Dada las características de una planta, se busca predecir a qué especie pertenece: *Iris setosa*, *Iris versicolor* o *Iris virginica*. Consta de 150 datos y cuatro atributos.

2.1.7. Liver disorders data set

Esta base de datos contiene información sobre muestras de sangre de personas del sexo masculino y el número de vasos de alcohol que beben por día. Con estas características, se busca predecir con estas características si una persona sufre trastornos hepáticos o no. Consta de 345 datos y seis atributos.

2.1.8. Pima indians diabetes data set

Esta base de datos contiene información médica de pacientes femeninos menores de 21 años con antecedentes familiares de diabetes. Se busca predecir si los pacientes presentan diabetes o no. Consta de 768 datos y ocho atributos.

2.1.9. Connectionist bench (Sonar, mines vs. rocks) data set

Esta base de datos contiene patrones obtenidos por rebote de una señal de sonar en diferentes ángulos y bajo condiciones distintas. Las señales de sonar se hacen rebotar sobre cilindros de metal y sobre rocas, buscando predecir si un determinado patrón está rebotando sobre metal o sobre roca. La clase contiene 208 datos y 60 atributos, donde cada uno de éstos representa la energía dentro de una determinada banda de frecuencia.

2.1.10. Statlog (Vehicle silhouettes) data set

Esta base de datos contiene información extraída de imágenes en dos dimensiones de cuatro tipos de vehículos distintos. Se busca predecir, dadas las características de una imagen, que tipo de vehículo representa. Consta de 842 datos y 18 atributos.

2.1.11. Connectionist bench (Vowel recognition – Deterding data) data set

Esta base de datos contiene información de análisis extraída de señales de audio, donde diferentes personas leen un texto. Este texto consiste en palabras donde se destacan 11 pronunciaciones diferentes de vocales en el

idioma inglés. La predicción que se busca en este ejemplo consiste en determinar qué vocal pronuncia un interlocutor dado el análisis de la señal de audio. Consta de 990 datos y 10 atributos.

2.1.12. Wine data set

Esta base de datos contiene resultados de análisis químicos de distintos vinos los cuales fueron hechos con uvas de tres viñedos diferentes. Se busca predecir el viñedo (clase) al cual pertenece una muestra dada. Consta de 178 datos y 13 atributos.

2.1.13. Breast cancer Wisconsin (Original) data set

Esta base de datos contiene información médica sobre pacientes con cáncer de mama y se busca predecir si un paciente determinado presenta tumores benignos o malignos. Consta de 699 datos y nueve atributos.

2.1.14. Forest Covertype data set

Esta base de datos contiene información cartográfica y diferentes mediciones realizadas sobre siete tipos de bosques distintos. Dada la información cartográfica de un bosque en particular se busca predecir a que tipo de bosque pertenece. Consta de 581.012 datos y 54 atributos, de éstos atributos sólo 10 son atributos numéricos, el resto son atributos binarios. Por lo comentado en la sección 4.7 del capítulo 2, se eliminaron los atributos binarios armando el modelo de datos con los 10 atributos numéricos.

2.2. Resultados

Los resultados obtenidos en cada una de las bases de datos se muestran en la Tabla 4-2 y en la Tabla 4-3. En estas dos tablas se muestran el valor del parámetro μ utilizado en 10 corridas independientes del algoritmo de CLUHR. Para estas mismas 10 corridas se muestra la media y el desvío estándar del número de veces que es necesario recorrer la base de datos para lograr el modelo de datos, la cantidad de superposiciones analizadas, el número de hiper-rectángulos con los que termina el modelo de datos, el número de reglas extraídas del modelo de datos (excepto para la base de datos de Wisconsin, donde el número de reglas coincide con el número de hiper-rectángulos al no poder simplificar más de una regla en una única regla), el promedio de cláusulas usadas en las reglas extraídas y la precisión lograda por el modelo de datos.

3. Comparaciones con otros métodos

Los resultados obtenidos en la sección anterior son comparados contra los resultados obtenidos con otros métodos de clasificación. Para ello, se utilizó el método clásico de clasificación C4.5 (Quinlan, 1993), una estrategia que, al igual que la propuesta, utiliza hiper-rectángulos junto con un algoritmo evolutivo para obtener las reglas de clasificación (García, y otros, 2009) y una estrategia que utiliza PSO junto con ACO para obtener el conjunto de reglas de clasificación (Holden, y otros, 2008).

Estos métodos fueron elegidos para llevar a cabo una comparación, ya que todos ellos producen como resultado del modelo un conjunto de reglas de clasificación. A continuación, se hacen unos breves comentarios sobre cada algoritmo y la manera en que fue medido.

Asimismo, se lleva a cabo una comparación entre la exactitud del modelo de datos, el número de reglas extraídas y el promedio de cláusulas por regla.

3.1. C4.5

El método C4.5 (Quinlan, 1993), algoritmo clásico para resolver problemas de clasificación, es un algoritmo que construye un árbol de decisión. Este algoritmo comienza analizando toda la base de datos buscando un valor en un determinado atributo que le permita dividir el conjunto de datos en dos subconjuntos. Cada uno de estos dos subconjuntos es analizado en forma independiente y para cada uno de ellos también se busca un valor en un determinado atributo que divida cada subconjunto en dos nuevos subconjuntos. De esta manera, se va construyendo un árbol binario y se va armando de tal forma que en las hojas del árbol queden subconjuntos de datos formados por una misma clase.

Tabla 4-2. Para las primeras siete bases de datos probadas se muestran el valor del parámetro μ utilizado en el algoritmo y los resultados con media y desvío estándar de 10 corridas independientes obtenidos con el test 10-fold cross-validation.

	Ecoli	Glass	Haberman	Segmen-tation	Ionos-phere	Iris	Liver
Valor del parámetro μ	6	4	12	3	20	16	6
Número de veces que se recorrió la base de datos	3,53 (0,33)	3,97 (0,37)	5,28 (0,32)	1,67 (0,06)	2,47 (0,14)	1,5 (0,06)	5,20 (0,50)
Superposiciones analizadas	17,56 (1,93)	28,09 (3,35)	10,08 (0,97)	4,95 (0,51)	3,16 (0,50)	0,75 (0,10)	17,56 (2,14)
Número de hiper-rectángulos del modelo	12,62 (1,44)	15,17 (1,30)	4,29 (0,33)	10,93 (0,47)	3,98 (0,37)	3,21 (0,12)	17,79 (2,21)
Número de reglas simplificadas extraídas	12,62 (1,44)	15,17 (1,30)	4,29 (0,33)	10,93 (0,47)	3,98 (0,37)	3,21 (0,12)	17,79 (2,21)
Promedio de cláusulas por regla	4,65 (0,15)	5,37 (0,18)	2,54 (0,06)	3,74 (0,10)	5,17 (0,18)	2,08 (0,05)	5,01 (0,06)
Precisión del modelo de datos	0,7891 (0,0160)	0,6215 (0,0360)	0,7356 (0,0064)	0,8538 (0,0134)	0,8777 (0,0169)	0,9300 (0,0079)	0,5919 (0,0211)

Tabla 4-3. Para las últimas seis bases de datos probadas se muestran el valor del parámetro μ utilizado en el algoritmo y los resultados con media y desvío estándar de 10 corridas independientes obtenidos con el test 10-fold cross-validation.

	Pima	Sonar	Silhou-ettes	Vowel	Wine	Wiscon-sin	Forest Covertype
Valor del parámetro μ	6	16	8	10	6	4	3000
Número de veces que se recorrió la base de datos	4,97 (0,39)	2,41 (0,17)	5,06 (2,56)	2,99 (0,11)	1,20 (0,01)	3,02 (0,32)	5,24 (0,45)
Superposiciones analizadas	17,66 (1,47)	1,36 (0,28)	67,03 (4,64)	74,4 (4,72)	0,51 (0,03)	5,63 (0,80)	16,14 (2,34)
Número de hiper-rectángulos del modelo	10,45 (0,91)	4,14 (0,20)	32,35 (2,03)	31,74 (0,78)	3,18 (0,11)	9,63 (1,39)	41,25 (2,05)
Número de reglas simplificadas extraídas	10,45 (0,91)	4,14 (0,20)	32,35 (2,03)	31,74 (0,78)	3,18 (0,11)	9,62 (1,38)	41,25 (2,05)
Promedio de cláusulas por regla	5,27 (0,12)	16,27 (0,72)	7,38 (0,33)	8,13 (0,27)	4,08 (0,09)	3,59 (0,11)	6,49 (0,48)
Precisión del modelo de datos	0,5595 (0,0191)	0,6666 (0,0283)	0,6819 (0,0171)	0,7120 (0,0132)	0,9529 (0,0113)	0,9251 (0,0102)	0,6928 (0,0149)

Se utilizó la implementación provista en la herramienta de data mining Tanagra (Rakotomalala, 2005). Con esta herramienta se realizó para cada conjunto de datos el test 10-fold cross-validation obteniendo así un promedio y desvío para la precisión, el número de reglas promedio y el promedio de cláusulas por regla.

3.2. EHS-CHC

El método propuesto en (García, y otros, 2009) al igual que CLUHR trabaja con hiper-rectángulos cuyas caras son paralelas a los ejes. Esta capacidad le permite trabajar con hiper-rectángulos para luego, con los límites de éstos, extraer las reglas de clasificación.

EHS-CHC comienza formando un conjunto *HS* de posibles hiper-rectángulos, donde cada uno de ellos representa a una clase. Luego, mediante la ejecución de un algoritmo evolutivo basado en CHC, termina encontrando un subconjunto de *HS* tal que los hiper-rectángulos de ese subconjunto proveen la mejor exactitud al modelo de datos.

Se utilizaron los resultados publicados en (García, y otros, 2009) para llevar a cabo las comparaciones.

3.3. PSO/ACO2

El método propuesto en (Holden, y otros, 2008) utiliza una combinación de las estrategias de optimización PSO y ACO para encontrar el conjunto mínimo de reglas de clasificación que logran una alta exactitud en el modelo de datos. En realidad, esta estrategia utiliza ACO para analizar los atributos nominales, mientras que usan PSO para los atributos de datos continuos. Por lo que para llevar a cabo las comparaciones con la técnica propuesta solo se tomará en cuenta este último.

El método completo trabaja descubriendo un conjunto de reglas para cada clase por separado, llevando al final un método de simplificación de reglas. Una particularidad de este método es que el conjunto de reglas obtenido es ordenado. Esto sucede porque una muestra dada comienza a evaluarse por cada una de las reglas en el orden determinado y, de este modo, se establece como clase de la muestra aquella en la cual se encontró la primera regla que satisface a la muestra.

Se utilizaron los resultados publicados en (Holden, y otros, 2008) para llevar a cabo las comparaciones.

3.4. Resultados

En esta sección se muestran los resultados comparativos entre lo obtenido por CLUHR y las estrategias C4.5, EHS-CHC y PSO/ACO2. Se compara también la precisión promedio alcanzada por el modelo de datos, la cantidad de reglas extraídas, la cantidad promedio de cláusulas de cada una de las reglas y el número de veces que se recorre la base de datos durante el armado del modelo.

En la estrategia propuesta se cuenta como número de reglas a la cantidad de hiper-rectángulos del modelo de datos y no a las reglas simplificadas para poder así comparar estos datos con los resultados obtenidos en (García, y otros, 2009), ya que CLUHR utiliza un procedimiento de simplificación de reglas (sección 5.1 del capítulo 2). Hay que notar que el mismo procedimiento de simplificación de reglas puede aplicarse a los resultados obtenidos por la técnica EHS-CHC, ya que ésta también produce hiper-rectángulos. Como en (García, y otros, 2009) no están publicadas las reglas obtenidas, no es posible encontrar el conjunto de reglas simplificadas para EHS-CHC. Por este motivo, solamente, se comparan la cantidad de hiper-rectángulos formados en el modelo de datos, siendo esta cantidad el número de reglas a comparar. De todas formas, en los ejemplos utilizados para estas pruebas, el número de reglas simplificadas es el mismo que el número de hiper-rectángulos, excepto para la base de datos Wisconsin (Tabla 4-2 y Tabla 4-3). Además, como en (García, y otros, 2009) los autores no publican la cantidad de cláusulas promedio, este valor no puede ser comparado.

En el algoritmo C4.5 de cada rama del árbol se extrae una regla de clasificación y la longitud de la rama determina el número de cláusulas.

El método de PSO/ACO2 arroja como resultado el conjunto de reglas de clasificación por lo que el propio resultado ya sirve para realizar las comparaciones. En (Holden, y otros, 2008) se publican la precisión del modelo, la cantidad de reglas extraídas y la cantidad de cláusulas promedio por regla por lo que es posible hacer todas las comparaciones deseadas. Hay que notar que el conjunto de reglas obtenido por PSO/ACO2 es un conjunto ordenado, esto implica que al momento de hacer una predicción se recorren en un orden determinado todas las reglas hasta encontrar aquella que satisface el dato a predecir. Esta particularidad logra un conjunto de reglas más reducido y al mismo tiempo con menos cláusulas por reglas. Por este motivo y para poder presentar resultados más comparables, las superposiciones que quedan al finalizar el modelo en la estrategia propuesta no se eliminan y al momento de realizar una predicción acerca de si una muestra cae en dos o más hiper-rectángulos, se eligen como clase de predicción la del hiper-rectángulo con menor volumen. Este cambio en la forma de extraer reglas en CLUHR permite elevar el valor del parámetro μ para logar una buena precisión del modelo, pero con menor cantidad de reglas. De esta manera, los resultados obtenidos por CLUHR son más comparables a los ofrecidos por la técnica PSO/ACO2.

Dado que la precisión de un modelo de datos y la cantidad de reglas son resultados inversamente proporcionales entre sí, ya que se puede lograr una muy buena precisión pero con una gran cantidad de reglas, y al mismo tiempo se puede obtener un número muy reducido de reglas pero con una mala precisión, en los ensayos realizados con CLUHR y con C4.5 se eligieron los parámetros correspondientes para lograr un equilibrio

razonable entre precisión y cantidad de reglas y así poder comparar contra los resultados publicados en (Holden, y otros, 2008).

Debido a que las ejecuciones del algoritmo C4.5 fueron llevadas a cabo para elaborar esta tesis y que los autores de (Holden, y otros, 2008) publican en sus resultados la media, desvío estándar y n para cada base de datos, se realiza una prueba t-student de doble cola con nivel de confianza del 95% para determinar si las diferencias logradas entre CLUHR y C4.5 y PSO/ACO2 son estadísticamente significativas o no. Como en (García, y otros, 2009) no publican ni media, ni desvío estándar, no puede usarse EHS-CHC para estas comparaciones. La Tabla 4-7 muestra los resultados obtenidos. En dicha tabla se marca con un signo "+" cuando CLUHR es mejor estadísticamente, con un signo "-" cuando CLUHR es peor estadísticamente y con un signo "=" cuando no hay significancias estadísticas.

Resumiendo los resultados mostrados en la Tabla 4-7 se discute uno por uno los resultados obtenidos en las distintas bases de datos estudiadas.

- Ecoli. La exactitud lograda y el número de reglas obtenida en C4.5 y en CLUHR resultan no ser estadísticamente significativas, mientras que si lo es el promedio de cláusulas por regla. Aunque viendo los resultados de la Tabla 4-6 la diferencia no llega a ser de una cláusula por regla.

- Glass. La exactitud lograda por CLUHR en esta base de datos es mucho menor que PSO/ACO2, aunque este último utiliza un mayor número de reglas para describir al modelo y al utilizar más reglas le es posible utilizar menos cláusulas. CLUHR alcanzó el mismo número de reglas y la misma cantidad de cláusulas por regla que C4.5 aunque no logrando la misma exactitud en el modelo de datos.

- Haberman's survival: En esta base de datos CLUHR fue estadísticamente mejor que C4.5 en todos los aspectos.

- Image segmentation: Comparado con C4.5, CLUHR solo logró usar menos cláusulas por regla aunque esta diferencia no llega a ser de una cláusula. PSO/ACO2 logra una muy buena exactitud pero utiliza el doble de reglas que CLUHR para hacerlo, y este número elevado de reglas le permite usar menos cláusulas por regla.

- Ionosphere. Comparado con C4.5, CLUHR logra una mejor exactitud en el modelo de datos usando menos reglas y un número similar de cláusulas por regla. PSO/ACO2 solo logra sacar diferencia en el número de cláusulas por regla.

- Iris. C4.5 logra una mejor exactitud con el costo de usar más reglas para la representación del modelo. PSO/ACO2 logra la misma exactitud que CLUHR usando menos reglas y menos cláusulas por regla.

- Liver disorders. C4.5 logra una mejor exactitud en el modelo de datos pero utiliza más de seis reglas que CLUHR y estas tienen una cláusula más que las logradas por CLUHR.

- Pima indians diabetes. Como en el caso anterior, C4.5 logra una muy buena exactitud con un número mayor de reglas, aunque estas tienen menos cláusulas que las extraídas por CLUHR.

- Sonar. CLUHR no alcanza la exactitud que logran C4.5 y PSO/ACO2, aunque CLUHR usa mucho menos reglas que C4.5 y un número levemente inferior comparado con PSO/ACO2. En número de cláusulas CLUHR utiliza en esta base de datos un número muy elevado en comparación a los otros dos métodos.

- Vehicle silhouettes. Utilizando el mismo número de reglas tanto C4.5 como PSO/ACO2 logran una mejor exactitud y un menor número de cláusulas por regla.

- Vowel recognition. Comparado con C4.5, CLUHR logra mejor exactitud con menor número de reglas aunque con dos cláusulas más por regla. Comparado con PSO/ACO2, este resulta mejor en todos los sentidos.

- Wine. CLUHR logra mejor exactitud con menor número de reglas que C4.5, aunque este último usa menos cláusulas por regla.

- Wisconsin. CLUHR no logra la exactitud de C4.5 y PSO/ACO2 pero utiliza un menor número de reglas para representar el modelo. CLUHR usa menos cláusulas por regla que C4.5 pero más que PSO/ACO2.

- Forest covertype. CLUHR logra la misma precisión que C4.5 pero con un número ligeramente más alto de reglas.

Haciendo un estudio de los distintos resultados obtenidos no es posible determinar que CLUHR se destaque sobre el resto, tampoco lo contrario, que CLUHR sea una técnica mala comparada contra el resto. Comparado con C4.5 los resultados han sido muy similares mientras que PSO/ACO2 parecería lograr un número reducido no solo de reglas, sino también de cláusulas por regla. Esto último se debe a dos factores claves que presenta la propia estrategia, el primero es que al tener ordenado el conjunto de reglas permite eliminar muchas cláusulas de las mismas, y segundo que al ser una estrategia de optimización, las partículas de PSO recorren todo el espacio de búsqueda encontrando una solución óptima.

El hecho que CLUHR no alcance buenas exactitudes en algunas bases de datos con distribuciones complicadas como la mostrada en la sección 1.12 se debe al hecho que, si se reduce el valor del parámetro μ para lograr una mejor precisión, aumenta significativamente el número de reglas. Y esta relación se da con mucha más brusquedad cuando los valores de μ son bajos, ya que si se elimina una superposición con pocos datos, al dividir estos hiper-rectángulos se generan unos más chicos aún y con menos datos, lo que incrementa considerablemente el número de hiper-rectángulos.

Comparando la precisión, el número de reglas y el promedio de cláusulas por regla podemos afirmar que CLUHR no se destaca (ni para bien ni para mal) sobre el resto de las técnicas estudiadas produciendo resultados similares a los que arrojan C4.5 y PSO/ACO2. El punto fuerte de CLUHR es el tema del costo computacional necesario para alcanzar este resultado. Este aspecto se verá con más detalle en la próxima sección.

Tabla 4-4. Esta tabla muestra los resultados de exactitud del modelo logrados por cada una de las estrategias medidas y para cada una de las bases de datos ensayadas. Se presentan las medias y entre paréntesis el desvío estándar de 10 corridas independientes.

	C4.5	EHS-CHC	PSO/ACO2	CLUHR
Ecoli	0,7964 (0,0141)	0,7948	-	0,7891 (0,0160)
Glass	0,6576 (0,0302)	0,6287	0,7095 (0,075)	0,6215 (0,0360)
Haberman	0,7103 (0,0202)	0,7122	-	0,7356 (0,0064)
Image	0,8586 (0,0155)	-	0,9667 (0,0117)	0,8538 (0,0135)
Ionosphere	0,9054 (0,0151)	-	0,8806 (0,0491)	0,8777 (0,0169)
Iris	0,9420 (0,0077)	0,9267	0,9467 (0,0526)	0,9300 (0,0079)
Liver	0,6418 (0,0300)	0,6167	-	0,5918 (0,0211)
Pima	0,7434 (0,0093)	0,7384	-	0,5595 (0,0191)
Sonar	0,7235 (0,0247)	0,7650	0,7505 (0,0911)	0,6666 (0,0283)
Vehicle	0,7111 (0,0099)	-	0,7305 (0,0445)	0,6819 (0,0171)
Vowel	0,6008 (0,0158)	-	0,8616 (0,0347)	0,7120 (0,0132)
Wine	0,9141 (0,0145)	0,9490	-	0,9530 (0,0113)
Wisconsin	0,9446 (0,0047)	0,9599	0,9487 (0,0253)	0,9251 (0,0102)
Forest covertype	0,7063 (0,0187)	-	-	0,6928 (0,0149)

Tabla 4-5. Esta tabla muestra los resultados de la cantidad de reglas extraídas por cada una de las estrategias medidas y para cada una de las bases de datos ensayadas. Se presentan las medias y entre paréntesis el desvío estándar de 10 corridas independientes.

	C4.5	EHS-CHC	PSO/ACO2	CLUHR
Ecoli	12,1 (1,45)	11,1	-	12,62 (1,44)
Glass	14,8 (0,79)	12,2	20,4 (1,35)	15,17 (1,30)
Haberman	10,7 (3,62)	4,4	-	4,29 (0,33)
Image	10,6 (0,70)	-	21,9 (0,99)	10,93 (0,47)
Ionosphere	10,2 (2,04)	-	3,6 (0,97)	3,98 (0,37)
Iris	4,0 (0,47)	3,4	3,0 (0,00)	3,21 (0,12)
Liver	23,9 (4,46)	9,8	-	17,79 (2,21)
Pima	13,2 (1,40)	11	-	10,45 (0,91)
Sonar	10,9 (1,60)	10,3	4,4 (1,58)	4,14 (0,20)
Vehicle	31,0 (2,31)	-	37,8 (1,2)	32,35 (2,03)
Vowel	32,8 (2,20)	-	29,0 (0,82)	31,74 (0,78)
Wine	5,1 (0,57)	3,6	-	3,18 (0,11)
Wisconsin	11,9 (1,79)	3,8	10,2 (1,87)	9,63 (1,39)
Forest covertype	39,7 (2,35)	-	-	41,25 (2,05)

Tabla 4-6. Esta tabla muestra los resultados del número promedio de cláusula por regla por cada una de las estrategias medidas y para cada una de las bases de datos ensayadas. Se presentan las medias y entre paréntesis el desvío estándar de 10 corridas independientes.

	C4.5	PSO/ACO2	CLUHR
Ecoli	4,32 (0,30)	-	4,65 (0,15)
Glass	5,68 (0,75)	3,11 (0,18)	5,37 (0,18)
Haberman	4,54 (1,27)	-	2,54 (0,06)
Image	4,31 (0,58)	2,8 (0,27)	3,74 (0,10)
Ionosphere	5,36 (0,89)	3,33 (0,79)	5,17 (0,18)
Iris	2,25 (0,27)	0,93 (0,14)	2,08 (0,05)
Liver	6,80 (1,30)	-	5,01 (0,06)
Pima	4,55 (0,27)	-	5,27 (0,12)
Sonar	3,99 (0,43)	2,6 (0,63)	16,27 (0,72)
Vehicle	7,10 (0,34)	3,85 (0,18)	7,38 (0,33)
Vowel	5,69 (0,18)	4,2 (0,25)	8,13 (0,27)
Wine	2,46 (0,17)	-	4,08 (0,09)
Wisconsin	4,31 (0,39)	1,21 (0,07)	3,59 (0,11)
Forest covertype	6,67 (0,82)	-	6,49 (0,48)

3.5. Análisis de rendimiento

En esta sección se intenta medir los recursos consumidos por cada estrategia evaluada en función del número de veces que se recorre la base de datos para armar el modelo.

Solo se mide el proceso de armado del modelo y no el de la extracción de reglas, ya que este último procedimiento es lineal para las cuatro estrategias. En C4.5 se debe recorrer todo el árbol, en PSO/ACO2 es el resultado colectado por los mejores individuos obtenidos para cada una de las clases y en EHS-CHC y CLURH las reglas se arman recorriendo los hiper-rectángulos formados en el modelo de datos. Hay que notar que al momento de pulir las reglas, el que se lleva todos los logros es el método C4.5 ya que las ramas del árbol ofrecen las reglas de clasificación pulidas, mientras que las otras tres estrategias hacen un pulido de regla usando un método greedy, el cual puede resultar muy oneroso, más si el número de reglas y el tamaño de la base de datos es grande.

A continuación se detalla el modo de proceder de cada estrategia y se estima cuantas veces debe ser recorrida la base de datos para armar el modelo de datos.

Tabla 4-7. Resultados de la prueba t-student para determinar si hay diferencias significativas entre los resultados obtenidos por CLUHR y los obtenidos por C4.5 y PSO/ACO2.

	Exactitud		Número de reglas		Promedio de cláusulas por regla	
	C4.5	PSO/ACO2	C4.5	PSO/ACO2	C4.5	PSO/ACO2
Ecoli	=		=		-	
Glass	-	-	=	+	=	-
Haberman	+		+		+	
Image	=	-	=	+	+	-
Ionosphere	+	=	+	=	=	-
Iris	-	=	+	-	=	-
Liver	-		+		+	
Pima	-		+		-	
Sonar	-	-	+	+	-	-
Vehicle	-	-	=	=	-	-
Vowel	+	-	+	-	-	-
Wine	+		+		-	
Wisconsin	-	-	+	+	+	-
Forest covertype	=		-		=	
Total	-3	-6	+8	+2	-2	-8

3.5.1. C4.5

El algoritmo C4.5 comienza analizando toda la base de datos para crear la primera división del nodo raíz del árbol. Luego, con cada subconjunto formado realiza un proceso iterativo que consiste en volver a analizar el subconjunto de datos buscando un nuevo atributo y su correspondiente valor donde hacer una nueva división de datos. Este proceso iterativo continúa hasta llegar a un subconjunto donde todos los datos son de la misma clase y donde ya no se requiere una nueva división. O hasta que la cantidad de datos de una clase es tan pequeña que no se justifica una nueva división. Esta cantidad de datos por la cual no se debe llevar a cabo una nueva división es un parámetro del algoritmo C4.5.

A priori es difícil determinar cuantas veces se recorre la base de datos en el algoritmo C4.5, ya que depende de cómo queden formados los nuevos subconjuntos de datos. Veamos con dos ejemplos sencillos cómo resulta difícil estimar la cantidad de veces que se recorre la base de datos, dada una base de datos.

- Ejemplo 1. Una base de datos tiene 80 datos de la clase A y 20 datos de la clase B. Al recorrer por primera vez la base de datos se determina un cierto valor para un atributo y se divide el nodo, 70 datos de la clase A van a parar al nodo 1, y los otros 10 datos junto con los 20 de la clase B van a parar al nodo 2. El nodo 1 como tiene todos los datos de una misma clase ya no requiere más atención. El nodo 2 se analiza (30 datos) y se determina un nuevo corte donde los 10 datos de la clase A van a parar al nodo 3 y los 20 datos de la clase B a nodo 4. Ya no se requiere más atención y la base de datos fue recorrida 1.3 veces.

- Ejemplo 2. Al igual que el ejemplo anterior una base de datos tiene 80 datos de la clase A y 20 datos de la clase B. Se determina una primera división y 10 datos de la clase A van a parar al nodo 1 y 70 datos de la clase A con los 20 datos de la clase B van a parar al nodo 2. El nodo 1 ya no requiere atención y en el nodo dos se vuelven a revisar los datos (90 datos) para determinar que los 70 datos de la clase A van a parar al nodo 3 y los 20 datos de la clase B van a parar al nodo 4. Ya no se necesita hacer más divisiones y la base de datos en este ejemplo fue recorrida 1.9 veces.

Como puede verse, la base de datos tiene la misma cantidad de clases, la misma cantidad de datos en cada clase y la estructura del árbol final es la misma en ambos ejemplos. Por como se determinaron las divisiones en el ejemplo 1, la base de datos se recorrió 1,3 veces, mientras que en el ejemplo 2 se recorrió 1,9 veces.

Como los resultados presentados en la sección 3.4 fueron realizados especialmente para estas comparaciones fue posible medir, gracias a la información que ofrece la herramienta Tanagra (Rakotomalala, 2005), cuantas veces fue recorrida la base de datos en cada uno de los ejemplos.

3.5.2. EHS-CHC

Esta estrategia comienza armando un conjunto inicial de posibles hiper-rectángulos. Se crea un hiper-rectángulo para cada dato y luego para cada uno de estos se encuentran los *K*-1 datos más cercanos de la misma clase, donde el dato *K* es un dato de otra clase. Con los *K*-1 datos se forman los límites del hiper-rectángulo. De esa manera, se obtiene el conjunto *HS* de todos los posibles hiper-rectángulos que no presentan superposición alguna.

Los individuos del proceso evolutivo son una representación binaria con tantos bits como hiper-rectángulos de *HS*, donde el bit *h* guarda como un 1 o como un 0, la información de si el hiper-rectángulo *h* está representado por el individuo o no.

La evaluación del fitness de un individuo consiste en determinar la exactitud del modelo con los hiper-rectángulos representados por el individuo. Con los hiper-rectángulos representados por el individuo se arma el modelo de datos y la evaluación del fitness consiste en recorrer toda la base de datos buscando la predicción del modelo con cada uno de los datos. La función de fitness además de contar la cantidad de predicciones positivas, tiene un factor de ponderación para brindar más fitness a los individuos con menos cantidad de hiper-rectángulos.

De esta manera, esta estrategia busca encontrar de manera evolutiva el subconjunto mínimo de hiper-rectángulos de *HS* y que al mismo tiempo ofrezca una buena exactitud en el modelo.

El armado inicial del conjunto HS, si bien en (García, y otros, 2009) no describen el pseudocódigo, es de esperar que sea de orden $O(n^2)$, ya que para cada dato se deben buscar los más cercanos de su misma clase.

Luego en el proceso evolutivo se recorre la base de datos completa en cada evaluación de fitness. En (García, y otros, 2009) se menciona que se utiliza una población de 50 individuos y se realizan 200 generaciones, por lo que se realizan 10000 evaluaciones de fitness y la base de datos es recorrida esta misma cantidad de veces. Aún cuando no se realicen todas las generaciones, ya en la primer generación la base de datos se recorre 50 veces (una por cada individuo).

3.5.3. PSO/ACO2

El algoritmo de extracción de reglas comienza con un conjunto de reglas *RS* vacío y luego para cada clase encuentra sus correspondientes reglas de clasificación. En un conjunto *TS* se almacenan todos los datos de la clase *C* a tratar.

Para obtener las reglas de la clase *C* el algoritmo realiza un proceso iterativo, donde en cada iteración se ejecuta una vez el algoritmo PSO/ACO2 el cual devuelve la mejor regla que describe los datos usando solo atributos nominales. Esta regla es usada como base para por ser completada con los atributos continuos.

Para encontrar las cláusulas de los atributos continuos se utiliza un PSO convencional con algunas modificaciones. El vector de las partículas contiene dos valores para cada atributo continuo, uno para el límite inferior y otro para el superior. Para evaluar una partícula, su vector es convertido en regla, la cual se agrega a la devuelta por PSO/ACO2 para llevar a cabo su evaluación. De esta manera, la mejor partícula de PSO será el complemento de la regla devuelta por PSO/ACO2.

Esta regla, luego de ser simplificada, es agregada al conjunto *RS*, y todos los datos que satisfacen a la regla son eliminados del conjunto *TS*. De esa manera, el algoritmo de PSO/ACO2 termina un lazo continuando con el proceso iterativo de volver a aplicar PSO/ACO2 para los datos restantes y completando la regla devuelta con las cláusulas de atributos continuos. Este proceso iterativo continúa hasta que los datos que queden en *TS* sean menores que un umbral.

Al finalizar de extraer un conjunto de reglas para cada una de las clases, se ordena las reglas de todos estos conjuntos de acuerdo a un criterio de calidad explicado con detalle en (Holden, y otros, 2008).

Como los ejemplos estudiados en este capítulo son bases de datos con atributos continuos, no interesa evaluar la parte de la ejecución de PSO/ACO2. Por lo tanto, para cada clase se encuentran las reglas de clasificación utilizando el algoritmo de PSO.

La primera vez que se intenta encontrar una regla, el algoritmo de PSO trabaja con todos los datos de una clase. La evaluación del fitness de una partícula implica recorrer toda la base de datos midiendo la exactitud de la regla representada por el vector de la partícula. Como resultado de esta operación se obtiene una regla y todos los datos que logran satisfacer a la regla no se vuelven a analizar para el hallazgo de una segunda regla. Como no es posible determinar qué subconjunto de datos es evaluado una y otra vez a medida que se van buscando reglas, se medirá como cota inferior el hecho de que para cada clase se extrae una única regla. Por lo tanto determinamos que para cada clase se ejecuta una vez el algoritmo de PSO con los datos de dicha clase.

En (Holden, y otros, 2008) se detalla que las corridas de PSO son realizadas con un cúmulo de 100 partículas y que se ejecuta un máximo de 100 iteraciones. En el mejor de los casos (que resulta prácticamente imposible) PSO ejecuta una única iteración para obtener el resultado óptimo, para lo cual tuvo que evaluar el fitness de 100 partículas y donde cada una de estas evaluaciones recorrió los datos de una clase. Si para cada clase se ejecuta un único PSO (caso hipotético prácticamente imposible) donde la corrida solo ejecutó una iteración (caso hipotético prácticamente imposible), entonces la base de datos tuvo que ser recorrida de manera completa al menos 100 veces.

Esta es la cota mínima de veces que la base de datos se recorre de manera completa. En un ejemplo más real donde, supongamos que, para cada clase se extraen dos reglas, donde la segunda regla fue armada solo con el 50% de los datos de cada clase y cada PSO ejecutó 20 iteraciones entonces se obtiene que la base de datos se debe recorrer 2000 veces para la primer regla y 1000 veces para la segunda (ya que trabaja con la mitad de los datos) dando un total de 3000 veces.

3.5.4. Resultados

La Tabla 4-8 muestra las comparaciones de la cantidad de recursos utilizados por CLUHR y C4.5 en cada una de las bases de datos analizadas. Las estrategias EHS-CHC y PSO/ACO2 no aparecen en la tabla por desconocer la verdadera cantidad de veces que se recorre. Aunque, como se comentó en las descripciones de cada estrategia en EHS-CHC, se recorre 10000 veces (o 50 si se evalúa una única generación, la cual hipotéticamente podría dar un resultado óptimo) y en PSO/ACO2 se estima un promedio de 3000 veces (o 100 si se evalúa el cúmulo una vez al obtener un hipotético resultado óptimo). Aun así, con un pensamiento optimista, ambas estrategias están lejos de la cantidad de veces que se recorre la base de datos en CLUHR.

Como puede observarse en la Tabla 4-8, CLUHR recorre menos veces la base de datos comparado con C4.5. Se realizó un test t-student de doble cola al 95% y en la tabla figura un símbolo "+" cuando CLUHR es mejor estadísticamente, un símbolo "-" cuando CLUHR es peor estadísticamente y con un símbolo "=" cuando la diferencia no resulta estadísticamente significante. Hay casos donde el número de veces es similar, pero en otras se recorre una, dos y hasta tres veces menos que las recorridas por C4.5. Solo en dos bases de datos CLUHR necesitó recorrer más veces la base de datos que C4.5.

Obviamente, al ser EHS-CHC y PSO/ACO2 estrategias que utilizan optimización, la cual requiere la evaluación del fitness de los individuos una y otra vez y que por cada evaluación de fitness se recorre la base de datos, produce estrategias onerosas en cantidad de esfuerzo computacional. El punto fuerte de estas estrategias es que al ser estrategias de optimización aseguran un resultado óptimo para el problema que se quiera resolver.

Por otra parte, EHS-CHC y PSO/ACO2, si bien es de esperar que logren ante una misma entrada siempre la misma salida, como las estrategias de optimización utilizan el azar no se garantiza que se cumpla la condición mencionada. Al igual que C4.5, CLUHR es una estrategia determinista lo que implica que siempre que se presente la misma entrada de cómo resultado la misma salida y solo cambia esta si cambian los parámetros.

También es cierto que cuando se presenta un problema uno deba probar con varios valores para los parámetros del algoritmo y así probar los distintos resultados que se obtienen. Los ejemplos presentados en esta sección fueron probados con tres, cuatro o cinco valores de μ, hasta encontrar el valor que produce el mejor resultado. Si se probara con 10 valores distintos de μ para asegurarse un buen resultado, la base de datos se recorre 10 veces más, una por cada valor del parámetro μ y luego quedarse con el mejor resultado. Aun así, si a la cantidad de veces mostrada en la Tabla 4-8 la multiplicamos por 10, resulta en un número mucho menor que las veces que recorren la base las estrategias EHS-CHC y PSO/ACO2.

De todas formas, se verá en el capítulo siguiente una forma de trabajo para probar distintos valores de μ que no presenta carga extra en el consumo de recursos permitiendo probar distintos valores de μ con un número de veces levemente superior a los mostrados en la Tabla 4-8.

En resumen, CLUHR demostró resolver todos los problemas estudiados ofreciendo resultados similares a las estrategias comparadas, pero con una utilización de recursos similar a la que presenta C4.5 y mucho menor a los utilizados por EHS-CHC y PSO/ACO2.

En el siguiente capítulo se discuten todas las mejoras que presenta la estrategia propuesta en esta tesis en cuanto a recursos consumidos con múltiples corridas y la adaptabilidad del modelo.

Tabla 4-8. Tabla con las comparaciones de los recursos utilizados (número de veces que se recorre la base de datos) por CLUHR y C4.5 en cada una de las bases de datos.

	C4.5	CLUHR	Significancia
Ecoli	4,19 (0,39)	3,53 (0,33)	+
Glass	5,64 (1,10)	3,97 (0,37)	+
Haberman	3,61 (1,26)	5,28 (0,32)	-
Image	3,84 (0,35)	1,67 (0,06)	+
Ionosphere	5,78 (0,73)	2,47 (0,14)	+
Iris	2,02 (0,13)	1,5 (0,06)	+
Liver	6,59 (1,48)	5,20 (0,50)	+
Pima	3,74 (0,24)	4,97 (0,39)	-
Sonar	4,03 (0,49)	2,41 (0,17)	+
Vehicle	5,98 (0,24)	5,06 (2,56)	=
Vowel	5,54 (0,13)	2,99 (0,11)	+
Wine	2,34 (0,10)	1,20 (0,01)	+
Wisconsin	3,19 (0,35)	3,02 (0,32)	=
Forest covertype	5,71 (0,72)	5,24 (0,45)	=
Total			+7

4. Minería incremental

Como se mencionó en el capítulo 3, CLUHR posee la capacidad de adaptarse a los cambios que sufren los datos sin necesidad de armar el modelo nuevamente, utilizando para ello la base de datos completa. En esta sección se mide el rendimiento de CLUHR comparado contra la técnica ITI (Utgoff, y otros, 1996).

El principal problema que presenta la técnica ITI, como cualquier técnica basada en árboles de decisión es que la acumulación de datos y la re-evaluación de la función de decisión de los nodos provoca que tarde o temprano se requiera una re-estructuración de un sub-árbol. Cuando el sub-árbol a rehacer tiene como raíz un nodo de los primeros niveles, entonces esta re-estructuración es importante, ya que la re-construcción de uno de estos sub-árboles representa recorrer un importante porcentaje de la base de datos. El pero caso, claro está, sucede cuando el nodo a re-estructurar es el nodo raíz del árbol, lo que provoca una re-estructuración completa y, por lo tanto, un recorrido completo de la base de datos.

En CLUHR, la aparición de nuevos datos, sólo causa que se modifique un hiper-rectángulo. Y si este tiene superposiciones con otros hiper-rectángulos entonces, se modifican los hiper-rectángulos involucrados.

Como se realizó en las secciones anteriores, se mide la cantidad de veces que se recorre la base de datos, esta vez para la modificación del modelo de datos. Se realiza un ensayo para cada una de las bases de datos del repositorio UCI presentadas en la sección 2.

El ensayo mostrado en esta sección consiste en 100 ejecuciones independientes para ambas técnicas, CLUHR e ITI. En cada ejecución se divide la base de datos en dos conjuntos al azar, el primero con el 70% de los datos y el segundo con el 30% restante. Con el primer sub-conjunto de datos se arma un modelo, tanto para CLUHR como para ITI y, con los datos del segundo-subconjunto se presentan uno por uno al modelo armado. Ante cada dato presentado se cuenta la cantidad de veces que se recorre la base de datos. Por lo tanto, para una ejecución, la cantidad total de veces que se recorre la base de datos es la suma de las veces que se recorre para cada dato. Luego se promedia los resultados de las 100 ejecuciones.

La Tabla 4-9 muestra los resultados comparativos entre el esfuerzo computacional que necesita CLUHR y el que necesita la técnica ITI. Se realizó un test t-student de doble cola al 95% y en la tabla figura un símbolo "+" cuando CLUHR es mejor estadísticamente, un símbolo "-" cuando CLUHR es peor estadísticamente y con un símbolo "=" cuando la diferencia no resulta estadísticamente significante.

Como puede observarse en la Tabla 4-9, CLUHR requiere mucho menos esfuerzo computacional que el que necesitan los árboles de decisión basados en el algoritmo ITI. CLUHR fue muy superior en 10 bases de datos, no sacó ventajas en la base de datos de Ionosphere, mientras que fue inferior solamente en dos (Iris y Wisconsin).

Tabla 4-9. Tabla con las comparaciones de los recursos utilizados (cantidad de veces que se recorre la base de datos) por CLUHR e ITI en cada una de las bases de datos.

	ITI	CLUHR	Significancia
Ecoli	5,19 (0,95)	0,59 (0,45)	+
Glass	14,50 (2,56)	0,44 (0,11)	+
Haberman	12,84 (2,24)	0,99 (0,25)	+
Image	2,37 (0,44)	0,19 (0,15)	+
Ionosphere	1,71 (0,30)	1,59 (0,43)	=
Iris	0,25 (0,05)	0,90 (0,50)	-
Liver	14,58 (2,70)	0,61 (0,16)	+
Pima	21,53 (3,69)	1,31 (0,37)	+
Sonar	5,11 (0,90)	0,06 (0,05)	+
Vehicle	14,42 (2,48)	1,07 (0,32)	+
Vowel	31,20 (5,30)	0,11 (0,05)	+
Wine	1,32 (0,26)	0,26 (0,41)	+
Wisconsin	1,65 (0,30)	8,31 (2,11)	-
Total			+8

Capítulo 5

Discusión y trabajo a futuro

La única posibilidad de descubrir los límites de lo posible es aventurarse un poco más allá de ellos, hacia lo imposible.
(Arthur C. Clarke)

Se ha presentado una técnica de clasificación, denominada CLUHR, la cual consiste en formar hiper-rectángulos en el espacio de los datos y, mediante contracción y división de los mismos, se van eliminando las superposiciones que presentan hiper-rectángulos de distintas clases. Este proceso de contracción y división continúa hasta eliminar todas las superposiciones o bien hasta reducir el número de las mismas y alcanzar un resultado aceptable. Producto de los hiper-rectángulos que queden formados se extraen las reglas de clasificación que resumen, de manera clara para el usuario, las características de los datos estudiados.

El uso de esta estrategia puede ser completamente automático, definiendo nada más que un único parámetro. O ser completamente supervisado por un experto en el dominio del problema, quien interviene en todas las decisiones que toma la estrategia para armar el modelo de datos.

Se ha discutido la capacidad de esta técnica para adaptar su estructura interna a los cambios en la base de datos, ya sea ante inserción de nuevos datos, eliminación o incluso de modificación y subclasificación de los mismos.

Se ha presentado con ejemplos ficticios de dos dimensiones cómo se comporta la estrategia dependiendo del problema, de la distribución de los datos y del grado de "mezcla" que tienen los datos de cada clase. Del mismo modo, se estudió en detalle cómo CLUHR fue resolviendo las distintas superposiciones hasta alcanzar el modelo de datos final.

Para este estudio, se utilizaron varias bases de datos del repositorio UCI (Frank, y otros, 2010) para probar la estrategia propuesta midiendo mediante el test 10-fold cross-validation la exactitud lograda por el modelo, el número de reglas extraídas y el número de cláusulas promedio por regla. Asimismo, se compararon estos resultados contra los obtenidos por el método clásico de clasificación C4.5 y dos métodos de extracción de reglas publicados recientemente, uno de los cuales también utiliza hiper-rectángulos como estructura de representación de los datos.

1. CLUHR

Se expuso una técnica de clasificación basada en extracción de reglas por cobertura, denominada CLUHR. Dicha técnica trabaja con hiper-rectángulos en el espacio *D*-dimensional de los datos para buscar los límites de cada

clase y, de esa manera, extraer un conjunto de reglas de clasificación que explique de manera clara las características de los datos y que sirva como conocimiento útil al usuario.

La filosofía del procedimiento para armar el modelo de datos consiste en formar un gran hiper-rectángulo inicial para cada clase, donde este hiper-rectángulo contiene a todos los datos de la clase. Es de esperar que los hiper-rectángulos iniciales presenten cierta superposición en el espacio. Es entonces cuando el algoritmo de CLUHR comienza con la tarea de dividir uno o más hiper-rectángulos en otros más pequeños para eliminar dichas superposiciones. La formación de nuevos hiper-rectángulos produce nuevas superposiciones, las cuales son analizadas y eliminadas mediante nuevas divisiones. Este proceso continua hasta eliminar todas las superposiciones o bien hasta alcanzar un modelo aceptable.

1.1. Índices de separabilidad

Para llevar a cabo las divisiones de los hiper-rectángulos cuando hay más de una superposición en el espacio, en esta tesis se propone un análisis de dichas superposiciones. Se analiza tanto el volumen superpuesto de ambos hiper-rectángulos, como así también la cantidad de datos de cada clase presente en la superposición y cómo se distribuyen estos datos en el espacio superpuesto. Estos análisis se realizan en cada una de las dimensiones del espacio de trabajo mediante el cálculo de unos índices denominados índices de separabilidad Z_i, los cuales son ponderados y promediados luego, para obtener el valor final del índice Ω (el cual arroja un valor entre cero y uno). Dicho índice determina el grado de modificación que se produce en el modelo si se modificaran los hiper-rectángulos que participan en tal superposición. De esta manera, se calcula primero un índice Ω para cada superposición. Aquel índice que obtiene el valor más alto, posteriormente determina la superposición a eliminar.

Lo novedoso de analizar las superposiciones es que es posible seguir con esta línea de investigación tratando de descubrir nuevos índices que midan otras características de una superposición no estudiadas en esta tesis y que puedan servir para mejorar la calidad del modelo de datos obtenido. Esto último es posible ya que la estrategia propuesta es completamente personalizable y permite elegir de una batería de índices Z cuál usar y cuál no usar para llevar a cabo el cálculo final del índice Ω. La elección de qué índices utilizar para armar un modelo de datos se puede realizar dependiendo del problema a resolver.

Por lo tanto, la capacidad de personalizar a CLUHR con determinados índices Z es una gran ventaja que la convierte en una técnica poderosa y flexible para la tarea de clasificación y extracción de conocimiento en minería de datos.

1.2. Supervisión de un experto en el dominio del problema

En un reciente trabajo (Cao, 2010) se introduce un nuevo paradigma denominado domain-driven data mining (D^3M), donde se intenta dirigir el pensamiento actual de "extracción de conocimiento centrado en los datos" a otro que sea "entrega de conocimiento accionable impulsado por el dominio". En esta dirección, aparece la necesidad de utilizar la minería de datos y la extracción de conocimiento para el nacimiento de una nueva generación de técnicas y estrategias que ayuden al usuario en la toma de decisiones para un problema dado.

(Cao, 2010) menciona en su trabajo algunos aspectos que debería tener una estrategia para poder ser utilizada en el paradigma D^3M. En esta misma línea CLUHR cumple algunos de ellos, por lo que es factible perfeccionar a futuro la técnica presentada en esta tesis para que pueda ser utilizada en la "extracción y entrega de conocimiento accionable".

Entre estos aspectos de D^3M están los que involucran a la inteligencia de los datos, del dominio y del humano. CLUHR es capaz de armar un modelo de los datos formando hiper-rectángulos, los cuales se convierten en representantes de un subconjunto de datos de una clase en particular. Estos hiper-rectángulos describen por sí mismos las características de los datos, por lo que es posible extraer información útil en forma de reglas. La inteligencia del dominio y del humano es una característica que puede ser explotada en CLUHR, ya que como se comentó en la sección 7 del capítulo 2 es posible que un experto en el dominio del problema supervise y participe durante el proceso de armado del modelo de datos, decidiendo qué superposición eliminar y cómo llevar a cabo la división de los hiper-rectángulos.

CLUHR es un modelo que es capaz de adaptarse a los cambios de los datos sin necesidad de volver a armar desde el principio el propio modelo, usando nuevamente la base de datos completa. En este sentido, la adaptación también puede ser supervisada por un humano y, tal como se discute en la sección 7 del capítulo 2, es factible implementar CLUHR en una herramienta que aprenda de las decisiones tomadas por el experto para luego sugerirlas a futuro cuando ocurran eventos similares.

1.3. Adaptabilidad

En el capítulo 3 se describió cómo es posible adaptar la estructura interna del modelo de datos logrado por CLUHR sin necesidad de volver a ejecutar el proceso de armado completo. Se presentaron las acciones que deben realizarse sobre los hiper-rectángulos ante la llegada de nuevos datos, la eliminación o modificación de los existentes e incluso la subclasificación de datos generando nuevas clases.

Este es otro gran beneficio de la técnica CLUHR, ya que ahorra tiempo computacional y, por sobre todo, permite retener el conocimiento adquirido con anterioridad. Dado que para otros casos una nueva ejecución completa del armado del modelo no garantiza que se extraigan las mismas reglas y, por lo tanto, que se obtenga el mismo conocimiento que se poseía anteriormente.

Como se puede apreciar en esta tesis, las otras estrategias estudiadas no presentan técnicas de adaptación. Aun así, como la estrategia EHS-CHC (García, y otros, 2009) modela los datos con hiper-rectángulos al igual que CLUHR, es posible aplicar la misma metodología que se utiliza en CLUHR y que se describe en el capítulo 3. Aunque esta adaptación "destruye" la filosofía planteada por EHS-CHC de ser una estrategia evolutiva.

Si EHS-CHC usara su filosofía para adaptarse a los nuevos cambios, entonces deberían formarse los hiper-rectángulos iniciales y volver a ejecutar el proceso evolutivo. Esto no solo involucra un excesivo consumo de recursos computacionales, sino que además provocaría la pérdida de conocimiento que podría no surgir del nuevo modelo.

PSO/ACO2 (Holden, y otros, 2008) no plantea cómo llevar a cabo una adaptación, pero se sabe que al ser una estrategia de optimización la misma necesita volver a ejecutar el procedimiento completo para encontrar una solución óptima, presentando así las mismas desventajas que se mencionaron anteriormente para las otras técnicas.

El método C4.5 tampoco presenta procedimientos para adaptar la estructura interna y debería ejecutarse el algoritmo de clasificación de manera completa para poder reflejar los cambios que sufren los datos. En esta misma dirección se ha investigado durante mucho tiempo y se han propuesto diferentes soluciones a la minería de datos incremental utilizando árboles dinámicos (Chao, y otros, 2009) (Piao, y otros, 2010) (Qingyun, 2011) (Schlimmer, y otros, 1986) (Shaorong, y otros, 2010) (Utgoff, 1994) (Utgoff, 1988) (Utgoff, 1989) (Utgoff, y otros, 1996).

1.4. Comparaciones

CLUHR fue testeado con 13 bases de datos de UCI. Este ensayo se realizó para observar la comparación de los resultados obtenidos con los métodos C4.5 (Quinlan, 1993), EHS-CHC (García, y otros, 2009) y PSO/ACO2 (Holden, y otros, 2008). Utilizando el test 10-fold cross-validation fueron medidas la precisión lograda por el modelo de datos armado, el número de reglas extraídas, el promedio de cláusulas por regla y la cantidad de veces que se recorre la base de datos para lograr armar el modelo de datos.

En cuanto a la precisión obtenida por el modelo de datos, el número de reglas extraídas y el promedio de cláusulas por regla, por lo visto en el capítulo anterior no se puede concluir que CLUHR sea mejor o peor que el resto de las estrategias comparadas. Quizás es levemente inferior a PSO/ACO2, ya que como se comentó previamente, este método permite una "ambigüedad" en el conjunto de reglas. Éstas reglas, por otro lado, presentan un cierto orden de ejecución. Al mismo tiempo, al ser una estrategia de optimización es de esperar que consiga encontrar una partícula que represente un resultado óptimo.

A pesar de que CLUHR no se destaca sobre el resto de las técnicas estudiadas presentando resultados similares en cuanto a precisión, número de reglas extraídas y número promedio de cláusulas por regla, ella presenta dos grandes ventajas:

- Al igual que C4.5, CLUHR es una estrategia determinista, por lo que una misma entrada siempre produce la misma salida, y este es un aspecto que no pueden garantizar EHS-CHC y PSO/ACO2 ni ninguna otra estrategia de optimización.

- La cantidad de veces que CLUHR recorre la base de datos para armar el modelo de datos es mucho menor que las utilizadas por las estrategias de optimización y ligeramente más bajo que la cantidad de recorridas llevadas a cabo por C4.5, llegando a presentar un promedio de dos veces menos que C4.5.

Ante esta última ventaja CLUHR se ha revelado superior incluso sobre C4.5, al probar el armado del modelo con distintos juegos de valores para los parámetros del algoritmo. Esta cuestión se discute en la próxima sección.

1.5. Trabajando con valores decrecientes para μ

El algoritmo de clasificación de CLUHR presenta como ventaja el hecho de que solo cuenta con un parámetro para ajustar el algoritmo y lograr, cambiando el valor de este parámetro, distintos resultados. Como sucede siempre que se presenta un nuevo problema, no es posible saber a priori qué valor usar para los parámetros del algoritmo, y entonces se ejecuta el algoritmo con distintos valores de los parámetros para luego de todos los resultados obtenidos elegir aquel que produjo el mejor resultado. En el capítulo 4 se vio que aún corriendo 10 veces el algoritmo de CLUHR, probándolo con 10 valores distintos para el parámetro μ, la base de datos se recorre un número menor de veces que las que recorren las técnicas de EHS-CHC y PSO/ACO2 para obtener un único resultado con un único juego de parámetros, sacando algo más de ventaja sobre C4.5 (Tabla 4-8). Por tal motivo, existe evidencia en afirmar que CLUHR presenta una gran ventaja por la capacidad de obtener distintos resultados usando diferentes valores del parámetro μ pero en una única ejecución del algoritmo.

El parámetro μ determina el número mínimo de datos que debe presentar una superposición para que esta sea analizada y, por lo tanto, llevar a cabo una re-estructuración de la herramienta. Así, cuando el proceso de armado finaliza puede ocurrir que no existan superposiciones o bien que existan pero con una cantidad de datos menor al especificado por μ.

Por lo tanto, una forma práctica de usar CLUHR es la de comenzar con un valor alto para μ. Cuando el algoritmo finalice es de suponer que existan distintas superposiciones con una cantidad de datos menor a la especificada por el parámetro μ. Ante esta situación la técnica propuesta ofrece dos ventajas. La primera es que permite ver cuántas superposiciones continúan existiendo en el modelo de datos y cuántos datos hay en cada superposición, y la segunda ventaja es que permite continuar desde el punto en que finalizó el armado anterior, usando un valor menor para μ, sin necesidad de comenzar nuevamente desde el principio. Esto último es posible ya que el parámetro μ no interviene en los cálculos de los índices Ω y el utilizar valores distintos de μ hace que se llegue al mismo estado logrado con un valor de μ mayor.

El siguiente ejemplo ayuda a entender este concepto. Llamemos M_1 al estado intermedio alcanzado por el modelo usando el valor m_1 para μ. Este estado se alcanzó dividiendo s superposiciones en un cierto orden, donde la cantidad de datos presentes en cada una de las s superposiciones es mayor o igual que m_1. Si se comenzara a armar el modelo de datos desde el principio, con un valor $m_2 < m_1$ para μ, entonces la estrategia terminaría dividiendo las mismas s superposiciones que antes, ya que los datos participantes en cada una de las s superposiciones es mayor que m_2, por ser éste menor a m_1. Obviamente, el poder llegar a este estado con el valor m_2 presenta la posibilidad de continuar eliminando nuevas superposiciones si las mismas tienen una cantidad de datos mayor o igual que m_2.

Por consecuencia, una buena práctica para usar esta estrategia cuando se desea probar con diferentes valores del parámetro μ, es comenzar con un valor alto para μ, ejecutar el algoritmo para obtener un primer modelo de datos, observar cuantas superposiciones tiene el modelo y cuantos datos contiene cada una y, en función a esta información, elegir un nuevo valor para μ. De esta manera es posible reanudar el proceso desde el estado que

quedó con el anterior valor de μ y continuar así eliminando las superposiciones correspondientes para lograr un segundo modelo de datos. Este procedimiento repetitivo permite obtener distintos modelos utilizando distintos valores de μ, con el objetivo de encontrar un modelo óptimo con pocos recursos computacionales.

Esta característica de trabajo de CLUHR de una importancia significativa si se la compara con el resto de las estrategias, incluso con C4.5. Ejecutar el método C4.5 con distintos valores de los parámetros implica recorrer de nuevo la base de datos una y otra vez para obtener como resultado un nuevo árbol. Si bien por lo general en los procedimientos de optimización por lo general no es frecuente ejecutar para probar otro juego de parámetros, el hecho de hacerlo implica una re-ejecución completa y por lo tanto recorrer la base de datos millares de veces más.

La Tabla 5-1 muestra para las bases de datos estudiadas, la cantidad de veces que recorre la base de datos CLUHR y C4.5 al probar con distintos conjuntos de valores para los parámetros de ambos algoritmos. Se puede observar que la diferencia de la cantidad de veces que se recorre la base de datos usando el procedimiento de valores decrecientes comentado en esta sección es mucho menor en CLUHR. Y al mismo tiempo, solo resultan ligeramente mayores que la cantidad de veces promedio que figura en la Tabla 4-8.

Obviamente, el mayor esfuerzo se encuentra en el primer armado ya que deben eliminarse los hiper-rectángulos iniciales que son los que contienen todos los datos de cada clase. A medida que se va disminuyendo el valor del parámetro μ, los hiper-rectángulos que se modifican contienen menor cantidad de datos y por ende el esfuerzo computacional al generar nuevos hiper-rectángulos representativos mínimos es mucho menor. En la Figura 5-1 se muestra como progresa el número de veces que se recorre la base de datos para las bases de datos de Liver disorders y Pima indians diabetes, ya que éstas fueron las dos bases de datos donde más valores de μ se han probado hasta alcanzar un resultado óptimo.

Tabla 5-1. Tabla con las comparaciones de los recursos utilizados (número de veces que se recorre la base de datos) por las estrategias CLUHR y C4.5 en cada una de las bases de datos cuando se desean conocer distintos resultados al probar con distintos valores de los parámetros del algoritmo.

	Número de ejecuciones	C4.5	CLUHR
Ecoli	3	12,44	3,57
Glass	4	24,14	4,33
Haberman	3	10,61	10,31
Image	2	8,29	2,03
Ionosphere	2	12,60	4,48
Iris	2	4,04	3,47
Liver	5	30,98	6,41
Pima	5	18,70	8,63
Sonar	2	8,30	3,21
Vehicle	3	17,40	5,96
Vowel	3	15,46	3,57
Wine	1	2,46	1,39
Wisconsin	3	9,28	3,76

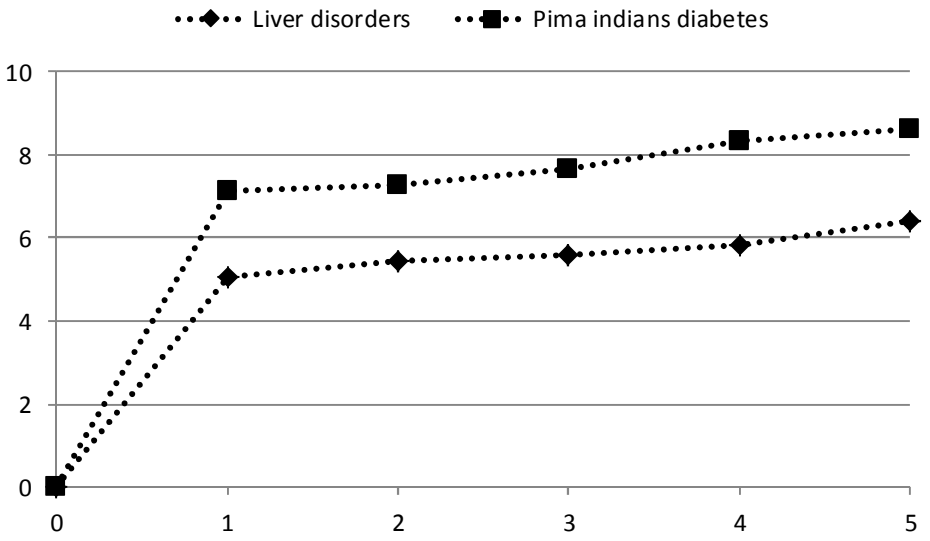

Figura 5-1. Progreso de la cantidad de veces que se recorre la base de datos a medida que se va probando con diferentes valores para μ.

2. Trabajo a futuro

CLUHR ha demostrado ser una poderosa estrategia para clasificación y extracción de conocimiento en donde se destacan las siguientes características:

- Ofrece similares resultados en cuanto a precisión, cantidad de reglas y cantidad de cláusulas por regla que se obtienen con otras estrategias de clasificación.
- Es una estrategia determinista que permite obtener el mismo resultado con la misma entrada.
- Consume mucho menos recursos que las estrategias de optimización y un número ligeramente menor a otras estrategias deterministas como C4.5.
- El algoritmo puede ser ejecutado de manera totalmente automática o totalmente supervisada pudiendo un experto en el dominio del problema participar en todas las decisiones que debe tomar la estrategia para lograr un modelo de datos.
- La estrategia es completamente personalizable pudiéndose elegir que índices de superposición (ver sección 3 del capítulo 2) utilizar a lo largo del procedimiento de armado del modelo.
- El algoritmo tiene un único parámetro lo que lo hace sencillo de utilizar. Además, mediante sus cálculos es posible obtener diferentes resultados con diferentes valores de este parámetro con el mismo costo computacional de una sola ejecución del algoritmo (ver sección 1.5).
- Es una estrategia adaptativa, esto permite modificar su estructura interna con poco esfuerzo sin llevar a cabo una ejecución completa del armado del modelo. Esta adaptación puede ser usada de manera automática o supervisada por un experto.

Aún con todas las ventajas que presenta la estrategia propuesta quedan varios aspectos en los cuales se puede seguir investigando. Los mismos son detallados a continuación.

2.1. CLUHR mejorado

2.1.1. Índices

En cuanto a los índices de superposición utilizados en la decisión de cuál superposición eliminar, los seis índices propuestos en esta tesis demostraron ser capaces de resolver con éxito numerosos problemas.

En esta dirección, es posible seguir investigando el desarrollo de nuevos índices que midan otros aspectos de una superposición, como por ejemplo índices que midan características de más de dos clases al mismo tiempo o que midan aspectos de más de una dimensión a la vez.

Por ejemplo, veamos el problema de dos dimensiones que se vio en la sección 1.5 del capítulo 4 donde se presentan cuatro clases con la particularidad que la distribución de las mismas "envuelven" unas a otras. Por la naturaleza del problema, los índices Z determinan que la primera superposición a eliminar sea la que forma la clase Cuadrado (que resulta ser la más grande en volumen y al mismo tiempo la más poblada) y la clase Cruz (que es la más pequeña en volumen y la que presenta menos datos). Esta superposición es elegida ya que tiene la particularidad que en proporciones, el hiper-rectángulo de la clase Cruz representa poco espacio con pocos datos comparado con el hiper-rectángulo de la clase Cuadrado, y así se convierte en la superposición candidata a eliminar ya que para eliminar a esta superposición solo basta con modificar ligeramente el hiper-rectángulo de la clase Cuadrado. Esta modificación del hiper-rectángulo de la clase Cuadrado causa que un nuevo hiper-rectángulo de esta misma clase presente una nueva superposición con las clases Círculo y Triángulo. Por el mismo motivo explicado anteriormente, la siguiente superposición a eliminar es la que forman las clases Cuadrado y la clase Triángulo causando una nueva división del hiper-rectángulo de la clase Cuadrado y una nueva superposición entre este hiper-rectángulo y la clase Círculo. Finalmente, en una tercera división se trata la superposición entre la clase Cuadrado y la clase Círculo eliminando así todas las superposiciones entre la clase Cuadrado y el resto de las clases (Figura 5-2).

El mismo problema detallado anteriormente ocurre con el hiper-rectángulo de la clase Círculo contra las clases Triángulo y Cruz.

Sería interesante encontrar un índice que se anticipe a estos problemas y que el cálculo del índice Ω determine que la primera superposición a eliminar sea la que forman las clases Cuadrado y Círculo. Ya que realizando esa acción se eliminan por medio de una sola acción las superposiciones que presenta el hiper-rectángulo de la clase Cuadrado con el resto de las clases. De esta manera, no solo se tratan menos superposiciones (tres, una para cada uno de los hiper-rectángulos de las clases Cuadrado, Círculo y Triángulo) sino que además se generan menos hiper-rectángulos finales (siete) y, al mismo tiempo, se recorre la base de datos una menor cantidad de veces (menos de dos) (Figura 5-3).

2.1.2. Unión de hiper-rectángulos

En el ejemplo visto en la sección 1.5.2 del capítulo 2 se ve claramente cómo es posible pulir las reglas de la clase Cuadrado y unificarlas en una única regla. Esto es posible ya que se pueden unir los hiper-rectángulos más pequeños de la clase Cuadrado en un único hiper-rectángulo, logrando así menos reglas en el modelo de datos.

Es interesante investigar si las uniones de los hiper-rectángulos de una misma clase pueden ser llevadas a cabo durante el proceso de armado del modelo o es un tratamiento extra que debería hacerse en la etapa final de extracción de reglas simplificadas. El lograr mejorar este aspecto favorecería al procedimiento de extracción de reglas simplificadas, ya que este procedimiento podrá trabajar con una menor cantidad de reglas.

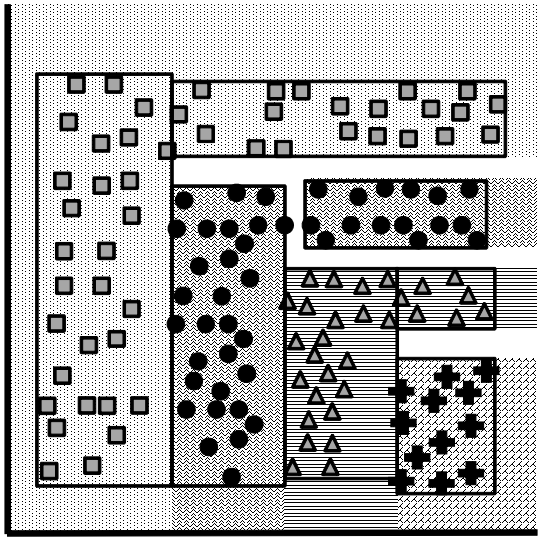

Figura 5-2. a) Ejemplo de distribuciones de cuatro clases de datos que "envuelven" a otras. b) El resultado que ofrece CLUHR con los índices de superposición presentados en esta tesis.

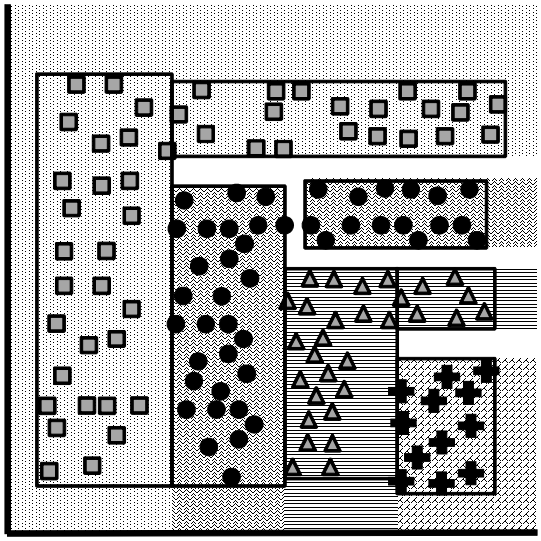

Figura 5-3. Resultado hipotético del modelo realizando solo tres divisiones de hiper-rectángulos.

2.1.3. Simplificación de reglas

En relación al aspecto anterior, uno de los puntos débiles de CLUHR es que el proceso final de extracción del conjunto de reglas simplificadas utiliza un procedimiento greedy del orden $O(n^2)$. Si bien es cierto que se han propuesto diferentes aproximaciones para realizar este trabajado de una manera más eficiente (Darrah, y otros, 2004) (Ma, y otros, 2009) (Shehzad, 2011), resulta interesante investigar de qué manera "marcar" aquellas caras de los hiper-rectángulos que representen un límite en el espacio de los datos. Como se mencionó en la sección 5 del capítulo 2, de cada una de las caras de un hiper-rectángulo se extrae una cláusula para la regla correspondiente a ese hiper-rectángulo. Por lo tanto, si es posible identificar aquellas caras que representan los

límites del espacio de datos se producirá la exclusión directa de la cláusula para la correspondiente regla sin necesidad de llevar a cabo el método de simplificación descripto en la sección 5.1 del capítulo 2.

Por ejemplo, en la Figura 5-4 se muestra un problema donde es posible determinar al momento de armar los hiper-rectángulos representativos mínimos iniciales de cada clase, cuáles son las caras que determinan los límites de los datos dentro del espacio. Por ejemplo, en la Figura 5-4 b se puede observar que en el hiper-rectángulo de la clase Cuadrado, las caras que están formadas por valor inferior del eje X y valor superior del eje Y pueden ser "marcadas" como límites del espacio de datos, puesto que más allá de esos valores no existe ningún dato en la base de datos. Lo mismo sucede con el valor inferior del eje Y y el valor superior del eje X para el hiper-rectángulo de la clase Triángulo.

En cada una de las divisiones que se realicen sobre los hiper-rectángulos durante el proceso del armado del modelo se debería poder seguir identificando a las caras de los nuevos hiper-rectángulos que se vayan generando para que dichas caras puedan ser "marcadas" como límites de los datos. De esta manera, al momento de hacer la extracción de reglas las caras "marcadas" como límites no serían agregadas como cláusulas a las reglas correspondientes, logrando de cada hiper-rectángulo la regla simplificada sin necesidad de ejecutar el método greedy de simplificación explicado en la sección 5.1 del capítulo 2. En este caso, el pulido de reglas es lineal y resulta directo. De cada hiper-rectángulo se podrá conocer si una cara es o no límite del espacio y por lo tanto así se logrará saber si agregarla o no a la regla de ese hiper-rectángulo.

Además, habría que investigar cómo hacer posible estas "marcas" de manera eficiente, identificando las caras de los hiper-rectángulos representativos mínimos iniciales y luego identificando las caras límites de los hiper-rectángulos que se vayan generando durante el armado del modelo, incluyendo los casos que podrían ser complicados de resolver. En estos casos, si bien la distribución de los datos de una clase no está en el límite del espacio de los datos, una de las caras de estos hiper-rectángulos podría ser marcada como límite de los datos en el espacio. Esta particularidad se puede ver en el ejemplo mostrado en la Figura 5-4 c donde la clase Círculo está en el centro de la distribución general de todos los datos de todas las clases, pero la cara que representa el valor inferior del eje Y puede ser "marcada" como límite de los datos.

En la Figura 5-4 c puede verse cómo quedan los hiper-rectángulos finales y a tal efecto se remarcaron en negrita aquellas caras que representan los límites de los datos. Estas caras no deberían ser utilizadas en el armado de las reglas simplificadas ya que no agregan riqueza a dichas reglas.

2.1.4. Operaciones con otros dominios de datos

Otro aspecto débil en la estrategia propuesta es que solo es capaz de trabajar en dominios donde todos los atributos son numéricos, e incluso como se vio en la sección 4.7 del capítulo 2, aún siendo numérico si este atributo tiene aspecto de categórico puede traer ruido al armado del modelo.

En esta dirección sería interesante poder trabajar de manera más confiable con atributos numéricos de pocos valores y con la posibilidad de adaptar la estructura de los hiper-rectángulos al domino de datos nominales. Esto podría obtenerse quizás trabajando con conjuntos de datos nominales asociados a los hiper-rectángulos.

Aunque esta resulta en una línea nueva de investigación podrían utilizarse la intersección de conjuntos de manera equivalente al de la superposición de hiper-rectángulos definiendo índices que midan aspectos de la superposición (intersección) y por sobre todo observando la forma de trabajar de manera conjunta con ambos dominios a la vez.

De igual manera, es posible pensar en incorporar a la herramienta la capacidad de trabajar con incertidumbre. En esta dirección, la teoría de los conjuntos aproximados (Rough sets theory) puede ser un buen punto de partida (Pawlak, 2002) (Pawlak, 1991). También es posible pensar en la posibilidad de utilizar hiper-rectángulos difusos para ampliar el dominio de trabajo actual de CLUHR.

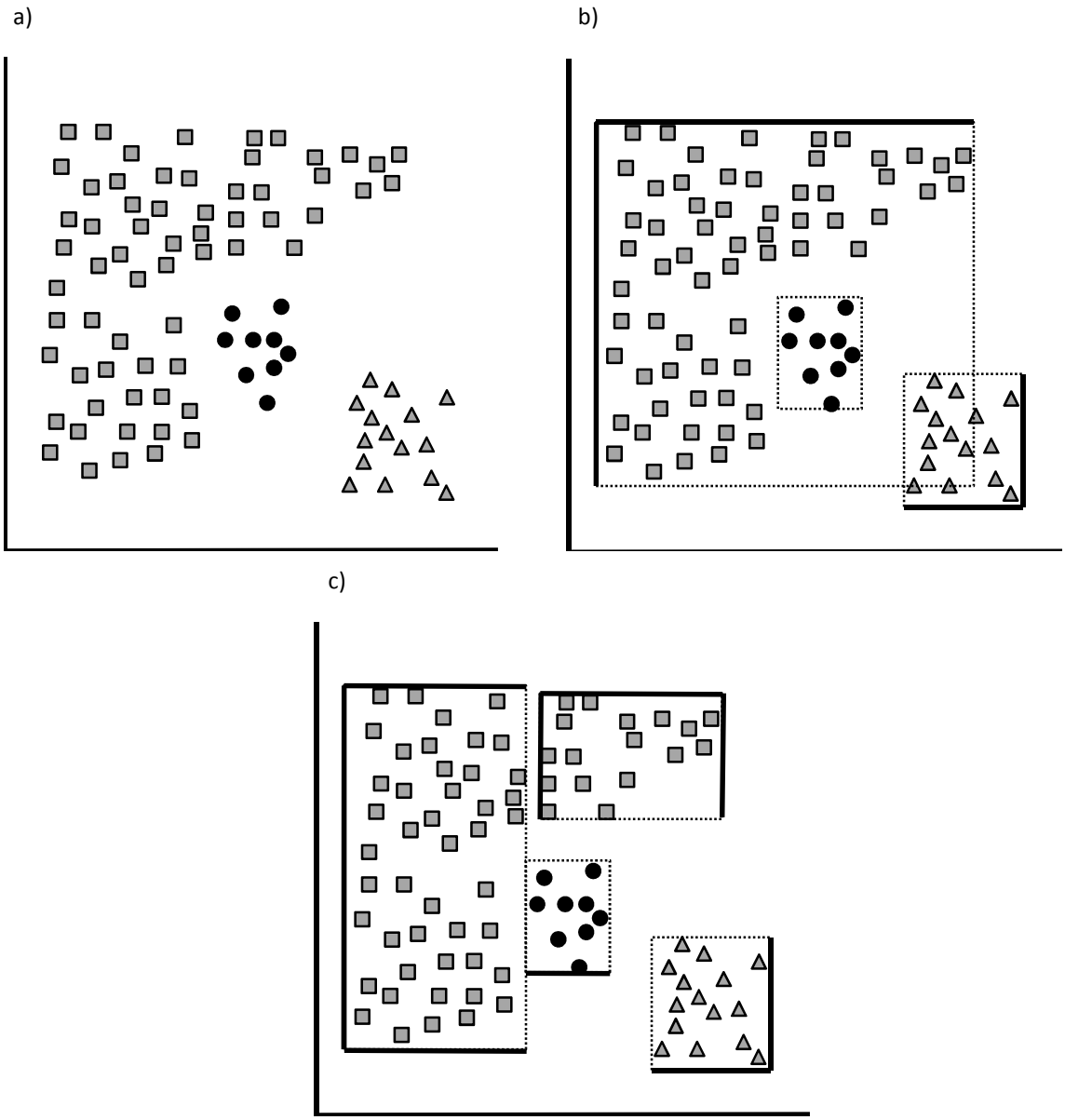

Figura 5-4. Ejemplo que muestra como sería posible "marcar" las caras de un hiper-rectángulo para luego excluir esa cara de la cláusula de las reglas. En a) se muestra la distribución original de los datos de las tres clases. En b) los hiper-rectángulos representativos mínimos iniciales. Se remarcan las caras de los hiper-rectángulo de las clases Cuadrado y Rectángulo que representan un límite de los datos en el espacio. En c) se muestran los hiper-rectángulos finales remarcando las caras que pueden ser descartadas en el armado de las reglas simplificadas y estas serían formadas solo con las caras que están punteadas.

2.1.5. Implementación de una herramienta de supervisión para expertos

En la sección 7 del capítulo 2 se discutió la posibilidad de que un experto en el dominio del problema pudiera supervisar el armado del modelo de datos pudiendo participar en las decisiones que debe tomar la propia estrategia al momento de eliminar una superposición. En esa misma sección, también se describieron las características que debería tener una herramienta que le facilite al experto el armado de datos.

Entre estas características se incluyen aquellas relacionadas con la "memoria" del modelo en donde es capaz de memorizar ciertas acciones llevadas a cabo por el experto para ser sugeridas a posteriori ante la llegada de

nuevos datos y la necesidad de adaptar el modelo, con lo cual se estaría consiguiendo una técnica que podría encuadrarse en el paradigma D^3M propuesto en (Cao, 2010).

Bibliografía

Aggarwal Charu C., Hinneburg Alexander y Keim Daniel A. On the Surprising Behavior of Distance Metrics in High Dimensional Space [Publicación periódica] // Proceedings of the 8th International Conference on Database Theory. - 2001. - págs. 420-434.

Ahmed Chowdhury Farhan [y otros] Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2009. - 12 : Vol. 21. - págs. 1708-1721.

Al-Hegami Ahmed Sultan Classical and Incremental Classification in Data Mining Process [Publicación periódica] // International Journal of Computer Science and Network Security. - 2007. - 12 : Vol. 7. - págs. 179-187.

Alippi Cesare, Boracchi G Giacomo y Roveri Manuel A just-in-time adaptive classification system based on the intersection of confidence intervals rule [Publicación periódica] // Neural Networks. - 2011. - 8 : Vol. 24. - págs. 791-800.

Andrews Robert, Diederich Joachim y Tickle Alan B. Survey and critique of techniques for extracting rules from trained artificial neural networks [Publicación periódica] // Knowledge-Based Systems. - 1995. - 6 : Vol. 8. - págs. 373–389.

Bae Duck-Ho [y otros] SD-Miner: A spatial data mining system [Publicación periódica] // IEEE International Conference on Network Infrastructure and Digital Content. - 2009. - págs. 803-807.

Barakat Nahla H. y Bradley Andrew P. Rule Extraction from Support Vector Machines: A Sequential Covering Approach [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2007. - 6 : Vol. 19. - págs. 729-741.

Beckmann Norbert [y otros] The R*-tree: an efficient and robust access method for points and rectangles [Publicación periódica] // Proceedings of the ACM SIGMOD international conference on Management of data. - 1990. - págs. 322-331.

Beyer Kevin [y otros] When Is "Nearest Neighbor" Meaningful? [Publicación periódica] // International Conference on Database Theory. - 1999. - págs. 217-235.

Bouillant S. [y otros] Real-time flaw detection on complex part: Study of SVM and hyperrectangle based method [Publicación periódica] // IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). - abril de 2002. - págs. IV-3596-IV-3599.

Breiman Leo [y otros] Classification and Regression Trees [Libro]. - Monterey : Wadsworth and Brooks, 1984. - ISBN 0412048418.

Cao Longbing Domain-Driven Data Mining: Challenges and Prospects [Journal] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - pp. 755-769.

Chao S. y Wong Fai An incremental decision tree learning methodology regarding attributes in medical data mining [Publicación periódica] // International Conference on Machine Learning and Cybernetics. - 2009. - págs. 1694-1699.

Chen Shyh-Kwei An exact closed-form formula for d-dimensional quadtree decomposition of arbitrary hyperrectangles [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2006. - 6 : Vol. 18. - págs. 784-798.

Chu Yi-Hong [y otros] Density Conscious Subspace Clustering for High-Dimensional Data [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 1 : Vol. 22. - págs. 16-30.

Cleveland William S. Visualizing Data [Publicación periódica]. - [s.l.] : Hobart Press, 1993. - ISBN 0963488406.

Cover T. y Hart Peter E. Nearest neighbor pattern classification [Publicación periódica] // IEEE Transactions on Information Theory. - 1967. - 1 : Vol. 13. - págs. 21-27.

Cui Xiaohui, Potok Thomas E. y Palathingal Paul Document Clustering using Particle Swarm Optimization [Publicación periódica] // Proceedings 2005 IEEE Swarm Intelligence Symposium. - 2005. - págs. 185-191.

Darrah Marjorie, Taylor Brian y Skias Spiro Rule Extraction from Dynamic Cell Structure Neural Networks Used in a Safety Critical Application [Publicación periódica] // Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference. - Florida : [s.n.], Mayo de 2004. - págs. 629-634.

Davis Jason V. [y otros] Information-Theoretic Metric Learning [Publicación periódica] // Proceedings of the 24th international conference on Machine learning. - 2007. - págs. 209-216.

Fayyad Usama M. Branching on attribute values in decision tree generation [Publicación periódica] // Proceedings of the twelfth national conference on Artificial intelligence. - Cambridge, MA : [s.n.], 1994. - Vol. 1. - págs. 601-606.

Ferri Ramirez Cèsar, Flach Peter y Hernandez Orallo José Learning Decision Trees Using the Area Under the ROC Curve [Publicación periódica] // Proceedings of the 19th International Conference on Machine Learning. - julio de 2002. - págs. 139-146.

Ferri Ramírez Cèsar, Hernández Orallo José y Ramírez Quintana María José Learning MDL-guided Decision Trees for Constructor-Based Languages [Publicación periódica] // Proceedings of 11th International Conference on Inductive Logic Programming. - septiembre de 2001. - págs. 39-50.

Filippi Anthony M. y Jensen John R. Effect of Continuum Removal on Hyperspectral Coastal Vegetation Classification Using a Fuzzy Learning Vector Quantizer [Publicación periódica] // IEEE Transactions on Geoscience and Remote Sensing. - 2007. - 6 : Vol. 45. - págs. 1857-1869.

Frank A. y Asuncion A UCI Machine Learning Repository [En línea] // Irvine, CA: University of California, School of Information and Computer Science. - 2010. - http://archive.ics.uci.edu/ml.

Freitas Alex A. Data mining and knowledge discovery with evolutionary algorithms [Libro]. - [s.l.] : Springer, 2002. - ISBN 3540433317.

García Salvador [y otros] A First Approach to Nearest Hyperrectangle Selection by Evolutionary Algorithms [Publicación periódica]. - [s.l.] : Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, 2009. - págs. 517-522.

Garcia-Almanza Alma Lilia and Tsang, Edward P.K. Simplifying Decision Trees Learned by Genetic Programming [Publicación periódica] // IEEE Congress on Evolutionary Computation. - 2006. - págs. 2142-2148.

Gupta Upavan y Ranganathan Nagarajan A Game Theoretic Approach for Simultaneous Compaction and Equipartitioning of Spatial Data Sets [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 4 : Vol. 22. - págs. 465-478.

Hart Peter E. The condensed nearest neighbor rule [Publicación periódica] // IEEE Transactions on Information Theory. - 1968. - 3 : Vol. 14. - págs. 515-516.

Hasperué Waldo y Lanzarini Laura Cristina A new clustering strategy for continuous datasets using hypercubes [Publicación periódica] // 36th Conferencia Latinoamericana de Informática. - Asunción, Paraguay : [s.n.], octubre de 2010.

Hasperué Waldo y Lanzarini Laura Cristina Classification Rules obtained from Dynamic Self-organizing Maps [Publicación periódica] // XII Congreso Argentino de Ciencias de la Computación. - San Luis, Argentina : [s.n.], Octubre de 2006. - págs. 1220-1230.

He Qing, Shi Zhong-Zhi y Ren Li-An The classification method based on hyper surface [Publicación periódica] // International Joint Conference on Neural Networks. - mayo de 2002. - págs. 1499-1503.

Hernández Orallo José, Ramiírez Quintana María José y Ferri Ramiréz Cèsar Introducción a la minería de datos [Libro]. - Madrid : Pearson Prentice Hall, 2004. - ISBN 84-205-4091-9.

Holden Nicholas y Freitas Alex A. A hybrid PSO/ACO algorithm for discovering classification rules in data mining [Publicación periódica] // Journal of Artificial Evolution and Applications. - 2008. - Vol. 2008. - págs. 1-11.

Hou Yuexian [y otros] Beyond Redundancies: A Metric-Invariant Method for Unsupervised Feature Selection [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 3 : Vol. 22. - págs. 348-364.

Hu Hsiao-Wei, Chen Yen-Liang y Tang Kwei A Dynamic Discretization Approach for Constructing Decision Trees with a Continuous Label [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2009. - 11 : Vol. 21. - págs. 1505-1514.

Hu Xiaohua Using Rough Sets Theory and Database Operations to Construct a Good Ensemble of Classifiers for Data Mining Applications [Publicación periódica] // Proceedings of the 2001 IEEE International Conference on Data. - 2001. - págs. 233-240.

Hu Xiaohua y Cercone Nick Mining Knowledge Rules from Databases: A Rough Set Approach [Publicación periódica] // Proceedings of the Twelfth International Conference on Data Engineering. - 1996. - págs. 96-105.

Hua Haiyang y Zhao Huaici A Discretization Algorithm of Continuous Attributes Based on Supervised Clustering [Publicación periódica] // Chinese Conference on Pattern Recognition. - 2009. - págs. 1-5.

Hung Chihli y Huang Lynn Extracting Rules from Optimal Clusters of Self-Organizing Maps [Publicación periódica] // Second International Conference on Computer Modeling and Simulation. - 2010. - págs. 382-386.

Kasten Eric P. y McKinley Philip K. MESO: Supporting Online Decision Making in Autonomic Computing Systems [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2007. - 4 : Vol. 19. - págs. 485-499.

Kasten Erik P. y Mckinley Philip K. Meso: Perceptual memory to support online learning in adaptive software [Publicación periódica] // Proceedings of the 3rd International Conference on Development and Learning. - 2004.

Katayama Norio y Satoh Shin'ichi The SR-tree: an index structure for high-dimensional nearest neighbor queries [Publicación periódica] // Proceedings of the 1997 ACM SIGMOD international conference on Management of data. - 1997. - págs. 369-380.

Kearns Michael y Mansour Yishay On the Boosting Ability of Top-Down Decision Tree Learning Algorithms [Publicación periódica] // Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing. - 1995. - págs. 459-468.

Khoshgoftaar Taghi M., Seliya Naeem y Liu Yi Genetic programming-based decision trees for software quality classification [Publicación periódica] // IEEE International Conference on Tools with Artificial Intelligence. - 2003. - págs. 374-383.

Konig R., Johansson U. y Niklasson L. Genetic programming - a tool for flexible rule extraction [Publicación periódica] // IEEE Congress on Evolutionary Computation. - 2007. - págs. 1304-1310.

Lee Seok-Lyong y Chung Chin-Wan Hyper-rectangle based segmentation and clustering of large video data sets [Publicación periódica] // Information Sciences. - 2002. - 1-2 : Vol. 141. - págs. 139-168.

Li J. y Liu H. Ensembles of cascading trees [Publicación periódica] // Third IEEE International Conference on Data Mining. - 2003. - págs. 585-588.

Lin Fan [y otros] A Novel Spatio-temporal Clustering Approach by Process Similarity [Publicación periódica] // Sixth International Conference on Fuzzy Systems and Knowledge Discovery. - 2009. - págs. 150-154.

Ma Jie [y otros] Rules Extraction from ANN Based on Clustering [Publicación periódica] // International Conference on Computational Intelligence and Natural Computing. - 2009. - págs. 19-21.

Maji Pradipta y Pal Sankar K. Feature Selection Using f-Information Measures in Fuzzy Approximation Spaces [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 6 : Vol. 22. - págs. 854-867.

Malone James [y otros] Data mining using rule extraction from Kohonen self-organising maps [Publicación periódica] // Neural Computing and Applications. - 2006. - 1 : Vol. 15. - págs. 9-17.

Marinica Claudia y Guillet Fabrice Knowledge-based Interactive Post-Mining of Association Rules Using Ontologies [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 6 : Vol. 22. - págs. 784-797.

Martens D. [y otros] Classification With Ant Colony Optimization [Publicación periódica] // IEEE Transactions on Evolutionary Computation. - 2007. - 5 : Vol. 11. - págs. 651-665.

Mladenić Dunja Data mining and decision support: integration and collaboration [Libro]. - Dordrecht : Kluwer Academic Publishers, 2003. - ISBN - 1-4020-7388-7.

Morik Katharina, Bhaduri Kanishka y Kargupta Hillol Introduction to data mining for sustainability [Publicación periódica] // Data Mining and Knowledge Discovery. - 2012. - 2 : Vol. 24. - págs. 311-324.

Ng Raymond T. [y otros] Exploratory Mining and Pruning Optimizations of Constrained Associations Rules [Publicación periódica] // Proceedings of the ACM SIGMOD international conference on Management of data. - 1998. - págs. 13-24.

Ni Bingbing, Yan Shuicheng y Kassim Ashraf A. Learning a Propagable Graph for Semisupervised Learning: Classification and Regression [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2012. - 1 : Vol. 24. - págs. 114-126.

Omran Mahamed G. H., Salman Ayed y Engelbrecht Andries P. Dynamic clustering using particle swarm optimization with application in image segmentation [Publicación periódica] // Pattern Analysis & Applications. - 2006. - 4 : Vol. 8. - págs. 332-344.

Parthalain N., Shen Qiang y Jensen R. A Distance Measure Approach to Exploring the Rough Set Boundary Region for Attribute Reduction [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 3 : Vol. 22. - págs. 305-317.

Parthaláin Neil Mac, Shen Qiang y Jensen Richard A Distance Measure Approach to Exploring the Rough Set Boundary Region for Attribute Reduction [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - [s.l.] : 22, 2010. - Vol. 3. - págs. 305-317.

Pawlak Zdzislaw Rough set theory and its applications [Publicación periódica] // Journal of Telecommunications and information Technology. - 2002. - Vol. 3. - págs. 7-10.

Pawlak Zdzisław Rough sets: theoretical aspects of reasoning about data [Libro]. - [s.l.] : Springer, 1991. - ISBN 0792314727.

Piao Minghao, Li Meijing y Ryu Keun Ho Using Significant Classification Rules to Analyze Korean Customers' Power Consumption Behavior: Incremental Tree Induction using Cascading-and-Sharing Method [Publicación periódica]. - 2010. - págs. 1649-1653.

Qingyun Chi Research on incremental decision tree algorithm [Publicación periódica] // International Conference on Electronic and Mechanical Engineering and Information Technology. - 2011. - págs. 303-306.

Quinlan John Ross C4.5: Programs for Machine Learning [Libro]. - [s.l.] : Morgan Kaufmann Publishers, Inc., 1993. - ISBN 1-55860-238-0.

Quinlan John Ross Induction of Decision Trees [Publicación periódica] // Machine Learning. - 1986. - 1 : Vol. 1. - págs. 81-106.

Rakotomalala Ricco TANAGRA: a free software for research and academic purposes [En línea] // Proceedings of EGC'2005, RNTI-E-3, vol. 2, pp.697-702. - 2005. - http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html.

Ramaswamy Sharadh y Rose Kenneth Adaptive Cluster Distance Bounding for High-Dimensional Indexing [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 6 : Vol. 23. - págs. 815-830.

Salzberg Steven A Nearest Hyperrectangle Learning Method [Publicación periódica] // Machine Learning. - 1991. - Vol. 6. - págs. 251-276.

Schlimmer Jeffrey C. y Fisher Douglas A Case Study of Incremental Concept Induction [Publicación periódica] // Proceedings of the Fifth National Conference on Artificial Intelligence. - 1986. - págs. 496-501.

Shali Amin, Kangavari Mohammad Reza y Bina Bahareh Using genetic programming for the induction of oblique decision trees [Publicación periódica] // Sixth International Conference on Machine Learning and Applications. - 2007. - págs. 38-43.

Shaorong Feng y Zhixue Han An incremental associative classification algorithm used for malware detection [Publicación periódica] // 2nd International Conference on Future Computer and Communication. - 2010. - págs. V1-757-V1-760.

Shehzad K. Simple Hybrid and Incremental Post-Pruning Techniques for Rule Induction [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2011. - pág. (en prensa).

Somol P. [y otros] Adaptive floating search methods in feature selection [Publicación periódica] // Pattern Recognition Letters. - 1999. - Vol. 20. - págs. 1157-1163.

Sun Haojun y Wang Shengrui Measuring the component overlapping in the Gaussian mixture model [Publicación periódica] // Data Mining and Knowledge Discovery. - 2011. - 3 : Vol. 23. - págs. 479-502.

Sutton Richard S. y Barto Andrew G. Reinforcement Learning: An Introduction [Libro]. - Cambridge, MA : MIT Press, 1998. - ISBN 0262193981.

Tettamanzi Andrea y Tomassini Marco Soft computing: integrating evolutionary, neural, and fuzzy systems [Libro]. - [s.l.] : Springer, 2001.

Thornton Chris Hypercuboid-Formation Behaviour of Two Learning Algorithms [Publicación periódica] // International Joint Conference on Artificial Intelligence. - 1987. - págs. 301-303.

Tsai Hsiao-Ping, Yang De-Nian y Chen Ming-Syan Mining Group Movement Patterns for Tracking Moving Objects Efficiently [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2011. - págs. 266-281.

Utgoff Paul E. [y otros] Decision Tree Induction Based on Efficient Tree Restructuring [Publicación periódica] // Machine Learning. - 1996. - 1 : Vol. 29. - págs. 5-44.

Utgoff Paul E. An Improved Algorithm for Incremental Induction of Decision Trees [Publicación periódica] // In Proceedings of the Eleventh International Conference on Machine Learning. - 1994. - págs. 318-325.

Utgoff Paul E. ID5: An Incremental ID3 [Publicación periódica] // In Proceedings of ML. - 1988. - págs. 107-120.

Utgoff Paul E. Incremental Induction of Decision Trees [Publicación periódica] // Journal Machine Learning. - 1989. - 2 : Vol. 4. - págs. 161-186.

Vachkov G. Online incremental image classification by use of human assisted fuzzy similarity [Publicación periódica] // IEEE International Conference on Information and Automation. - 2010. - págs. 834-839.

van der Merwe D.W. y Engelbrecht A.P. Data clustering using particle swarm optimization [Publicación periódica] // Congress on Evolutionary Computation. - 2003. - Vol. 1. - págs. 215-220.

van Rijsbergen C. J. Information retrieval [Libro]. - [s.l.] : Butterworths, 1979. - ISBN 0408709294.

Wang Qiang, Chen Huanhuan y Shen Yi Decision Tree Support Vector Machine based on Genetic Algorithm for fault diagnosis [Publicación periódica] // IEEE International Conference on Automation and Logistics. - 2008. - págs. 2668-2672.

Wang Xiujun y Shen Hong An Improved Growing LVQ for Text Classification [Publicación periódica] // Sixth International Conference on Fuzzy Systems and Knowledge Discovery. - 2009. - págs. 114-118.

Wei Jin-Mao, Wang Shu-Qin y Yuan Xiao-Jie Ensemble Rough Hypercuboid Approach for Classifying Cancers [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 3 : Vol. 22. - págs. 381-391.

Wenjun Liu y Fei You A rule extraction algorithm of continuous domain decide table [Publicación periódica] // Seventh International Conference on Fuzzy Systems and Knowledge Discovery. - agosto de 2010. - págs. 69-72.

Wettschereck Dietrich, Dietterich Thomas G. y Sutton Richard An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms [Publicación periódica] // Machine Learning. - 1995. - Vol. 19. - págs. 5-27.

Wong Pak Chung Visual data mining [Publicación periódica] // IEEE Computer Graphics and Applications. - 1999. - 5 : Vol. 19. - págs. 20-21.

Xiang Yang [y otros] Summarizing transactional databases with overlapped hyperrectangles [Publicación periódica] // Data Mining and Knowledge Discovery. - 2011. - 2 : Vol. 23. - págs. 215-251.

Zeng Huiwen y Trussell H.J. Constrained Dimensionality Reduction Using a Mixed-Norm Penalty Function with Neural Networks [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 3 : Vol. 22. - págs. 365-380.

Zhang Chenqi, Yu Philip S. y Bell David Introduction to the Domain-Driven Data Mining Special Section [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 6 : Vol. 22. - págs. 753-754.

Zhang Defu, Leung Stephen C. H. y Ye Zhimei A Decision Tree Scoring Model Based on Genetic Algorithm and K-Means Algorithm [Publicación periódica] // Third International Conference on Convergence and Hybrid Information Technology. - 2008. - págs. 1043-1047.

Zhang Tian, Ramakrishnan Raghu y Livny Miron BIRCH: an efficient data clustering method for very large databases [Publicación periódica] // Proceedings of the ACM SIGMOD international conference on Management of data. - Montreal, Quebec, Canada : ACM, 1996. - págs. 103-114.

Zhang Wei, Lin Zhouchen y Tang Xiaoou Learning Semi-Riemannian Metrics for Semisupervised Feature Extraction [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2011. - 4 : Vol. 23. - págs. 600-611.

Zhao Suyun [y otros] Building a Rule-Based Classifier—A Fuzzy-Rough Set Approach [Publicación periódica] // IEEE Transactions on Knowledge and Data Engineering. - 2010. - 5 : Vol. 22. - págs. 624-638.