Universidad Miguel Hernández de Elche
Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica

Análisis de catálogos robustos desde la perspectiva de la minería de reglas de asociación

Enrique Lazcorreta Puigmartí

Directores

Dr. Federico Botella Beviá

Dr. Antonio Fernández Caballero

Elche, 8 de septiembre de 2017

El dibujo de la contraportada es obra de Fermín Lazcorreta Puigmartí, realizado explícitamente para este trabajo.

Piedad Nieves De Aza Moya, Directora del Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica de la Universidad Miguel Hernández de Elche

HACE CONSTAR:

Que da su conformidad a la lectura de la tesis doctoral presentada por don Enrique Lazcorreta Puigmartí, titulada "Análisis de catálogos robustos desde la perspectiva de la minería de reglas de asociación", la cual ha sido desarrollada dentro del Programa de doctorado en Tecnologías Industriales y de Telecomunicación en este Departamento, bajo la dirección del Dr. Federico Botella Beviá, y el Dr. Antonio Fernández Caballero.

Elche, a veintiuno de julio de dos mil diecisiete.

Piedad Nieves De Aza Moya
Catedrática de Ciencia de Materiales e Ingeniería Metalúrgica.
Directora Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica

UNIVERSITAS
Miguel
Hernández

Federico Botella Beviá, director de tesis y profesor del Departamento de Estadística, Matemáticas e Informática de la Universidad Miguel Hernández de Elche

CERTIFICA

Que Enrique Lazcorreta Puigmartí ha realizado bajo mi supervisión su memoria de tesis doctoral titulada "Análisis de catálogos robustos desde la perspectiva de la minería de reglas de asociación", cumpliendo todos los objetivos previstos, finalizando su trabajo en forma satisfactoria para su defensa pública y capacitándole para optar al grado de Doctor.

Lo que certifico en Elche, a veintiuno de julio de dos mil diecisiete.

Fdo: Dr. Federico Botella Beviá

UNIVERSITAS
Miguel
Hernández

Antonio Fernández Caballero, director de tesis y profesor del Departamento de Sistemas Informáticos de la Universidad de Castilla-La Mancha

CERTIFICA

Que Enrique Lazcorreta Puigmartí ha realizado bajo mi supervisión su memoria de tesis doctoral titulada "Análisis de catálogos robustos desde la perspectiva de la minería de reglas de asociación", cumpliendo todos los objetivos previstos, finalizando su trabajo en forma satisfactoria para su defensa pública y capacitándole para optar al grado de Doctor.

Lo que certifico en Albacete, a veintiuno de julio de dos mil diecisiete.

Fdo: Dr. Antonio Fernández Caballero

UNIVERSITAS
Miguel
Hernández

Ojos…
son ojos por que te ven,
no por que tú los ves.

Sabiduría popular

UNIVERSITAS
Miguel
Hernández

Agradecimientos

Sin el tesón de Federico no habría terminado esta investigación. Antonio se alió con él para introducirme entre ambos en el mundo científico, tan separado del mundo académico, aunque yo pensara desde el principio y hasta el final que deberían ser lo mismo.

Tengo cierta facilidad para reproducir lo que se me explica bien, como todos. Pero escribir una tesis supone demostrar algo más, exponer algo nuevo aunque sea insignificante hoy. El futuro hará grandes nuestras ideas de hoy, o no, según sean nuestras ideas. Hoy hemos de demostrar que tenemos ideas, que aprendemos de quienes han tenido ideas. Hoy hemos de demostrar que utilizamos esas ideas con el propósito de mejorar lo que llamamos ciencia o conocimiento. Federico y Antonio me han enseñado las relaciones y diferencias entre la ciencia y la academia, han formado a otro individuo con la idea de que no se puede seguir en la academia sin aportar algo nuevo a la ciencia. La generosidad de incentivar a alguien a ser generoso con los demás es parte de la ciencia, sin vosotros no estarían estas líneas publicadas.

Hoy comparto esa idea. Soy docente porque creo que puedo enseñar a quien quiere aprender. En la redacción de esta memoria se nota que soy docente, quiero que quede bien explicado lo que narro. En la ciencia también ha de quedar claro qué se quiere demostrar, pero a los científicos les gusta leer sólo lo que les aporta nuevo conocimiento. Este detalle lo he

aprendido al redactar artículos científicos, se ha de ser escueto [Federico, Antonio], exponer ideas claras y contrastadas [Federico, Antonio].

Como docente no me expreso así. Primeo me aseguro de que es correcto lo que voy a explicar, y a continuación lo explico en clase y lo puedo preguntar en un examen sin temor a que se invalide la prueba. Como investigador estoy continuamente cuestionando si lo que digo es correcto hoy y si lo será mañana. Estos años de investigación me han enseñado que he de dudar de todo mientras enseño de qué se ha de dudar en las materias que imparto. He de dudar de las personas que me rodean y asesoran mientras comprendo cuánto puedo aprender de ellas.

Piedad no tenía que ser nadie especial, es sólo la directora actual del departamento al que está adscrito mi tesis. Pero ha demostrado que es capaz de sacrificar parte de su propio tiempo de trabajo/vida/investigación/... para gestionar con alegría y eficiencia la última parte de mi conversión en investigador. No me conocía, pero confió en que quienes me avalaban lo estaban haciendo bien, y yo también por la propiedad de transicción. No es fácil este ejercicio de generosidad sin asumir que todos nos esforzamos en mejorar lo que tenemos. Gracias, Piedad.

Mis amigo estáis ahí, tanto si me habéis ayudado desde una perspectiva investigadora como humana, estáis ahí. No os voy a citar con vuestro nombre, pero sabéis cuánto me habéis ayudado a cerrar esta fase de mi vida. Sabéis que si me preguntáis os contestaré sinceramente si había pensado escribir vuestro nombre. Preguntad, no os defraudaré, o sí, ya me conocéis. Sois tantos y, a veces, tan volátiles, que no quiero dar explicaciones sin preguntas. Gracias es lo que toca en esta sección (de un documento científico). Os quiero, es algo personal y os regalo una canción[1], y sé que muchos de vosotros también me queréis.

Gabriel y Fermiín me han marcado todo lo que pueden marcar dos hermanos, o más. Creo que decidí ser científico para no ser como vosotros, pero soy demasiado parecido (usando una escala logarítmica). Os quiero, y sé que me queréis.

Sin Almu es probable (casi seguro) que no hubiera terminado este objetivo. Contigo formé mi familia, algo más fuerte que la fuerza de gravedad, donde Sara y David tienen gran parte de la culpa, aunque no toda. Lo que más me motiva para buscar un mundo mejor sois vosotros (disculpas al

[1] `https://www.youtube.com/watch?v=1jZziub75eo`

resto del mundo). Intentar que unos científicos decidan que yo soy capaz de buscar un mundo mejor no es tarea fácil, pero si descubro algo insignificante me darán esa confianza, y podré buscar un mundo mejor para Almu, Sara/Toni y David. Os quiero, gracias por todo y más. Vos estime, incondicionalment.

La meva mare, Matilde, está con nosotros gracias a la ciencia, y a sus ganas de estar aquí. Sense tu no hi seriem aquesta investigació ni jo, gracies, mare, per tot el que m'has donat i el que em donaràs. T'estime, i sé que tu també, potser més (cosa de mares...).

El meu pare, Enrique, me enseñó que no podía dejar un mundo peor. Él fue científico de vocación, pero dedicó toda su vida a la docencia. A menudo lo recuerdo con la canción de Joan Manuel Serrat[2], un mensaje incompleto de todo lo que me enseñó. T'estime, i sé que tu també.

[2] https://www.youtube.com/watch?v=w7F7oYD1q90&feature=youtu.be

UNIVERSITAS
Miguel
Hernández

Resumen

Los datasets de clasificación son conjuntos de registros que recogen las características de individuos clasificados de una población. Las características son los valores que toma el individuo en ciertos atributos medibles. La clasificación ha de ser única, un individuo pertenece a una y sólo una de las clases en que se ha dividido la población

El análisis de un dataset de clasificación proporciona reglas de clasificación, mediante las que se puede clasificar a un individuo del que sólo se conozcan algunas de sus características.

Estos datasets contienen información sobre las relaciones existentes entre las diferentes características de la población en estudio. Cuando el dataset es una muestra representativa de la población, podemos dividir esta información en dos tipos bien diferenciados:

1. **Estructural** Si un atributo está relacionado con otro, esta relación se mostrará en los registros del dataset. Si no es posible que un individuo tome el valor x en un atributo y el valor y en otro atributo simultáneamente, no habrá ningún registro con ambos valores.

2. **Probabilística** Si el valor de un atributo aparece simultáneamente con el valor de un atributo de forma frecuente en la población, en el dataset ocurrirá lo mismo.

Cuando el dataset no es una muestra representativa, sólo contiene in-

formación estructural de la población. En este caso, se pueden eliminar los duplicados que contenga el dataset para reducir sus dimensiones y poder analizarlo mejor. Los duplicados sólo proporcionan información probabilística con la que estimar frecuencias poblacionales. Si el dataset no contiene este tipo de información, los duplicados sólo dan información estructural redundante. Sólo es necesario mantener un representante de los registros duplicados en el dataset.

Denominando catálogo al dataset reducido tras eliminar registros duplicados, los catálogos son datasets de clasificación que sólo contienen información sobre la estructura de la población en estudio. La minería de reglas de asociación o la minería de reglas de clasificación asociativa, que se basan en la información probabilística que tiene una muestra representativa de una población, no pueden utilizarse del modo habitual cuando se analiza un catálogo.

Esta tesis presenta una nueva metodología que permite descubrir información sobre la estructura de la población contenida en los catálogos. Al aplicarla sobre datasets difíciles de tratar con algoritmos basados en minería de reglas de asociación, proporciona una colección de catálogos que, utilizando menos atributos que el dataset original, contienen la misma información sobre la estructura de la población en estudio.

Índice general

Resumen xv

Índice de figuras xxi

Índice de tablas xxiii

Índice de conceptos teóricos xxv

Índice de listados xxx

1. Introducción 1

2. Estado del Arte 7

2.1. Minería de Reglas de Asociación 16

2.1.1. Modelo teórico 18

2.1.2. Bases de datos de transacciones, datasets 28

2.1.3. Algoritmos 30

2.1.4. El ítem raro 52

2.2. Minería de Reglas de Clasificación 56

2.2.1. Minería de Reglas de Clasificación Asociativa . . . 62

3. Primeros resultados **69**
3.1. Minería de Reglas de Asociación 69
3.1.1. Implementación de Apriori 70
3.1.2. Generación de reglas asociación 75
3.1.3. Datasets de reglas de asociación 81
3.1.4. Reglas de Oportunidad 88
3.2. Minería de Reglas de Clasificación 97
3.2.1. Reducción de datasets de clasificación 98
3.2.2. Catálogos 105

4. Catálogos **109**
4.1. Significado del soporte en catálogos 114
4.2. Datasets de clasificación 115

5. Modelización **119**
5.1. Modelo teórico 120
5.1.1. Análisis de datasets de clasificación 134
5.2. Modelo aplicado 138
5.2.1. Funciones 139
5.2.2. Estructuras 143
5.2.3. Algoritmo ACDC 144
5.2.4. Post-análisis 145

6. Validación **147**
6.1. Primera lectura de un dataset de clasificación 148
6.2. Atributos constantes 157
6.3. El algoritmo ACDC 158
6.3.1. Número de catálogos robustos del dataset de clasificación 161
6.3.2. Número mínimo de evidencias robustas 162
6.3.3. Evidencias robustas de los sub-catálogos 163
6.3.4. Reducción de atributos 164
6.4. Datasets de grandes dimensiones 164
6.4.1. Profundizando en grandes datasets 167

7. Conclusiones **175**
7.1. Resultados de la investigación 176
7.1.1. Publicaciones 178
7.2. Trabajo futuro 179

Bibliografía **194**

A. Sobre la bibliografía **195**

B. Acrónimos **197**

C. Datos **199**

D. Código **201**
D.1. Librería de funciones comunes 202
D.2. Características de los datasets de clasificación 209
D.3. Análisis de múltiples datasets 215
D.4. Macros y funciones auxiliares 221

Índice alfabético **227**

UNIVERSITAS
Miguel
Hernández

Índice de figuras

1.1. Proceso KDD 2

2.1. Fases del proceso de KDD [Fayyad et al. (1996)] 8
2.2. KDD: Selección de datos 10
2.3. KDD: Preproceso 11
2.4. KDD: Transformación 12

3.1. Tiempos de ejecución con la colección `BMS-POS.dat` 73
3.2. Tiempos de ejecución con la colección `BMS-View1.dat` . . . 74
3.3. Tiempos de ejecución con la colección de datos propios . . 75
3.4. `genrules()` original vs. modificado 80
3.5. $\mathcal{D}$ vs $\mathcal{R}$ 84
3.6. Árbol $\mathcal{L}$ del dataset de clasificación original 101
3.7. Árbol $\mathcal{L}$ del dataset de clasificación reducido 101
3.8. Árbol $\mathcal{L}$ para clasificación del dataset de clasificación reducido 102

4.1. Porcentaje de ítems con soporte inferior a $minSup = 1\,\%$ –ítems raros– contenidos en los 75 datasets de KEEL, en orden alfabético 116

UNIVERSITAS
Miguel
Hernández

Índice de tablas

3.1. Reglas analizadas y encontradas (foodmart) 79
3.2. Reglas analizadas y encontradas (T40I10D100K) 81
3.3. $\mathcal{D}$ vs $\mathcal{R}$ 85
3.4. `T10I4D100K` con un 50 % de confianza y oportunidad mínima 95
3.5. `T10I4D100K` con un 25 % de confianza y oportunidad mínima 95
3.6. Análisis de un dataset original vs su versión reducida . . . 105

6.1. Efecto de la supresión de atributos constantes 158
6.2. Resultados de la ejecución del algoritmo $\mathcal{ACDC}$ sin supervisión 160
6.3. Ejecución supervisada del algoritmo $\mathcal{ACDC}$ 166
6.4. Colección de catálogos robustos del dataset census sin 6 atributos 169
6.5. Colección de catálogos robustos del dataset census sin 10 atributos 169
6.6. Colección de catálogos robustos del dataset census sin 15 atributos 170
6.7. Características de los datasets de clasificación estándar de KEEL 172

UNIVERSITAS
Miguel
Hernández

Índice de conceptos teóricos

2.1. Definición (Población de ítems) 19
2.2. Definición (Itemset) 20
2.3. Definición (Transacción) 20
2.4. Definición (Dataset) 20
2.5. Definición (Regla de Asociación) 21
2.6. Definición (Minería de Reglas de Asociación) 22
2.7. Definición (Soporte de un itemset) 23
2.8. Definición (Soporte mínimo) 24
2.9. Definición (Ítem raro) 24
2.10. Definición (Itemset frecuente) 24
2.11. Definición (Conjunto de itemsets frecuentes) 25
2.12. Definición (Conjunto de candidatos a itemset frecuente) . . 25
2.13. Definición (Soporte de una regla de asociación) 25
2.14. Definición (Confianza de una regla de asociación) 26
2.15. Definición (Confianza mínima) 26
2.16. Definición ($lift$) 27
2.17. Definición ($conviction$) 28
2.1. Propiedad (A priori) 36
2.18. Definición (Rango de un atributo) 58
2.19. Definición (Evento primario) 58

2.2. Propiedad (Unicidad) 58
2.20. Definición (Evento compuesto) 58
2.21. Definición (Base de datos $\mathcal{P}$) 58
2.22. Definición (Registro clasificado) 58
2.23. Definición (Dataset de clasificación) 59
2.24. Definición (Regla de Clasificación) 59
2.25. Definición (Registro clasificado (ARM)) 60
2.26. Definición (Regla de clasificacion asociativa) 63

3.1. Definición (Regla de asociación negativa) 103

5.1. Definición (Caracterización) 121
5.2. Definición (Caracterización completa) 121
5.3. Definición (Caracterización incompleta) 121
5.4. Definición (Evidencia empírica) 121
5.5. Definición (Dataset de clasificación) 121
5.6. Definición (Catálogo) 122
5.1. Lema (Existencia de catálogos en los datasets de clasificación) 122
5.2. Lema (Interpretación del soporte de los ítems de un catálogo) 122
5.7. Definición (Clasificador ingenuo) 123
5.8. Definición (Caracterización desconocida) 123
5.9. Definición (Caracterización robusta) 124
5.10. Definición (Catálogo robusto) 124
5.11. Definición (Caracterización con incertidumbre) 124
5.12. Definición (Catálogo con incertidumbre) 125
5.13. Definición (Conjunto de caracterizaciones) 126
5.1. Teorema (Catálogo robusto) 126
5.1. Corolario (Catálogo con incertidumbre) 126
5.3. Lema (Descomposición de datasets de clasificación) 126
5.4. Lema (Confianza del clasificador ingenuo unitario) 127
5.14. Definición (Experimento incompleto) 128
5.15. Definición (Catálogo reducido) 129
5.16. Definición (Atributo redundante) 130
5.17. Definición (Atributo esencial) 130
5.5. Lema (Caracterizacion de atributo) 130
5.6. Lema (Herencia) 131
5.18. Definición (Colección de catálogos robustos) 132

5.7. Lema (Utilidad del $\mathcal{CCR}_{\mathcal{D}}$) 132
5.1. Proposición (Atributos constantes) 133
5.1. Propiedad (Superconjuntos de catálogos robustos) 133
5.8. Lema (Catálogos de $\mathcal{D}_{\mathcal{E}}$) 135
5.19. Definición (Caracterizaciones con incertidumbre de $\mathcal{CCR}_{\mathcal{D}}$) 136
5.20. Definición (Mayor evidencia completa) 137
5.21. Definición (Conjunto de caracterizaciones incompletas con incertidumbre de $\mathcal{D}_0$) 137
5.22. Definición (Conjunto de evidencias incompletas robustas de $\mathcal{D}_0$) 137

UNIVERSITAS
Miguel
Hernández

Índice de listados

2.1. Algoritmo AIS, 1993 32
2.2. Algoritmo SETM, 1993 33
2.3. Algoritmo SETM, función merge-scan 33
2.4. Algoritmo SETM, selección de candidatos 34
2.5. Algoritmo SETM, itemsets extendibles 34
2.6. Algoritmo Apriori, 1994 36
2.7. Algoritmo Apriori, función `unión` 37
2.8. Algoritmo Apriori, función `poda` 37
2.9. Algoritmo Apriori, función `genrules` 38
2.10. Algoritmo AprioriTID, 1994 38
2.11. Algoritmo DHP (fase 1), 1995 39
2.12. Algoritmo DHP (fase 2), 1995 40
2.13. Algoritmo DHP (fase 3), 1995 41
2.14. Algoritmo Partition, 1995 42
2.15. Algoritmo Basic, 1997 45
2.16. Algoritmo Cumulate, 1997 46
2.17. Algoritmo EstMerge, 1997 46
3.1. Función de unión y poda 70
3.2. Función de poda relajada 71
3.3. Función de unión y poda relajada 71
3.4. Función $genrules()$ modificada 78

3.5. Algoritmo ORFind 93
3.6. Algoritmo FP-ORFind 97
5.1. Obtención del catálogo $\mathcal{D}_0$ 139
5.2. Obtención de $\mathcal{D}^{-\mathcal{I}-i}$ a partir de $\mathcal{D}^{-\mathcal{I}}$ 140
5.3. Obtención de la columna a eliminar 141
5.4. Comprobar si un catálogo es robusto 141
5.5. Aislar incertidumbre del catálogo inicial 142
5.6. Obtener el $\mathcal{CCR}_{\mathcal{D}}$ de un catálogo robusto 143
5.7. Algoritmo $\mathcal{ACDC}$ 144
D.1. Funciones para gestionar datasets de clasificación estándar 202
D.2. Primer análisis de datasets de clasificación estándar 210
D.3. Lector de datasets de clasificación estándar 215
D.4. Macros y funciones auxiliares 221

1

Introducción

Uno de los métodos más utilizados por la ciencia desde sus inicios se basa en la toma de datos sobre los individuos de una población y su análisis en busca de patrones repetidos. El objetivo de esta metodología es siempre el mismo: transformar datos en conocimiento, experiencia en ciencia.

Todas las ciencias participan de algún modo este proceso. La *Física*, las *Matemáticas* y la *Estadística* aportan los cimientos teóricos, modelando los datos para posibilitar su análisis. Las *Ingenierías* transforman el conocimiento en tecnología, creando herramientas que aceleran el proceso de recolección, almacenamiento, análisis y transmisión de datos.

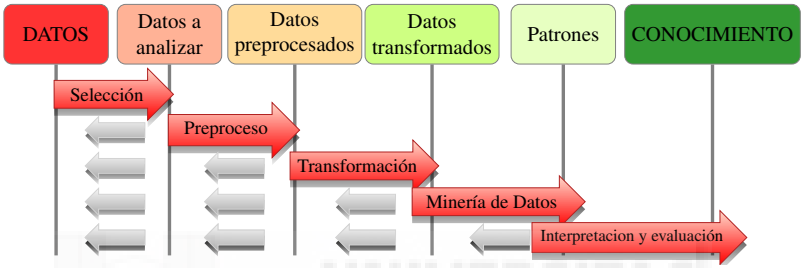
En las últimas décadas, el auge de la tecnología digital ha permitido que entidades de todo tipo recojan y almacenen cantidades enormes de datos. Diferentes áreas de la *Informática*, como la minería de datos o el aprendizaje automático, se han especializado en el análisis de esos datos.

Sin embargo, el número de bases de datos existente crece exponencialmente, a mayor velocidad que la capacidad real de analizarlos. El descubrimiento de conocimiento en bases de datos (del inglés, Knowledge Database Discovery, KDD), modeliza la transformación de datos en conocimiento mediante un proceso iterativo (ver figura 1.1).

Al ser tan grande el número de datos disponibles para cualquier investigación, lo primero que se ha de hacer es una *selección* de los datos que realmente serán analizados. A continuación se aplica un *preproceso* a los datos seleccionados, preparándolos para el análisis al que se van a so-

meter. Una vez preprocesados, se *transforman* para que puedan ser tratados eficientemente con los algoritmos seleccionados para la siguiente fase, *minería de datos*, en la que se descubren modelos o patrones. Finalmente, los resultados obtenidos se *interpretan y evalúan* y se convierten en conocimiento.

Figura 1.1: Proceso KDD



En esta tesis se desarrolla un proceso completo de KDD sobre colecciones de datos –datasets– de clasificación, con el objetivo de reducir o redistribuir la información válida que contienen. En el transcurso de la investigación se utilizaron diversas técnicas de minería de datos, profundizando en el estudio de la minería de reglas de asociación (en inglés, Association Rules Mining, ARM). Se usaron datasets de diferentes características y procedencia, la mayoría de ellos alojados en repositorios públicos. Las dimensiones de los datasets analizados impide su análisis profundo mediante técnicas de ARM, como demuestran múltiples trabajos sobre minería de reglas de clasificación asociativa (en inglés, Classification Association Rules Mining, CARM).

En estos repositorios se recogen datasets de clasificación que ya han pasado por las fases de selección, preproceso y transformación. Se descubrió que el preproceso aplicado a algunos datasets de clasificación alteraba en exceso sus características, invalidando el fundamento teórico de los algoritmos de ARM con que se querían analizar. Se trata de datasets obtenidos tras eliminar los duplicados de los datos seleccionados en la primera fase del proceso de KDD.

Las reglas de asociación son relaciones de dependencia estadística entre los datos de un dataset. Al eliminar los duplicados en la fase de preproceso se obtienen datasets de menores dimensiones, a cambio de perder

la información estadística contenida en los datos originales. Este descubrimiento condujo a una revisión de los conceptos utilizados.

La dependencia matemática es un caso específico de la dependencia estadística. En la primera se estudian relaciones exactas, en la segunda relaciones aproximadas que podrían ser exactas o inexactas.

El número de relaciones de dependencia estadística que hay en un dataset de clasificación es tan grande que no puede ser gestionado por los ordenadores que se utilizan en la mayoría de equipos de investigación. Además, todas las relaciones se tratan de forma individual, con lo que son necesarios más recursos de memoria para gestionar cada una de las relaciones descubiertas en un análisis de ARM.

En ARM se resuelven los problemas de desbordamiento de memoria eliminando los datos poco frecuentes. Sin embargo, esta técnica es cuestionable en los datasets de clasificación obtenidos tras eliminar instancias duplicadas. El principal objetivo de esta tesis consiste en obtener las mejores reglas de asociación de un dataset de clasificación sin utilizar criterios basados en la frecuencia de sus datos. Una consecuencia de este objetivo es la necesidad de gestionar eficientemente todos los datos de cualquier dataset de clasificación sin duplicados.

Estos datasets se pueden representar mediante matrices matemáticas, la tecnología actual permite trabajar de forma eficiente con grandes matrices. Las relaciones de dependencia matemática son un subconjunto de las de dependencia estadística, y al obtenerlas gestionando matrices no son necesarios tantos recursos de memoria. Si no se usa la información estadística que proporcionan las reglas de asociación, sólo se puede descubrir que "*existe*" o que "*no existe*" relación entre las características de un individuo y su clasificación. Cuando existe relación entre unas características y una clase, sólo se puede averiguar si esa relación es unívoca o si no lo es, si todos los individuos de similares características pertenecen o no a la misma clase.

Cuando existe una relación unívoca entre las características de un individuo y su clase, se puede afirmar que existe dependencia matemática entre los atributos del problema de clasificación y la clase en estudio. En términos de ARM, una dependencia matemática es una regla de asociación con un 100 % de confianza. Si la relación existente no es unívoca, como no se tiene información estadística sobre los datos recogidos, sólo se puede afirmar que la confianza de la regla de asociación descubierta es

inferior al 100 %.

Considerando que las reglas son "mejores" cuanto mayor es la confianza que tienen, sólo podemos diferenciar a las reglas que tienen un 100 % de confianza de las que no lo alcanzan. La búsqueda de las mejores reglas de asociación que contiene un dataset de clasificación se convierte en descubrir la dependencia o independencia matemática de las columnas del dataset sin duplicados. Las primeras pruebas realizadas nos llevaron a formular la hipótesis que se intenta resolver con esta tesis:

> *Si existen relaciones de independencia matemática entre los atributos de un dataset de clasificación sin duplicados y la clase en estudio, se pueden reducir las dimensiones del dataset sin perder información sobre el problema de clasificación original. Si se aplica el mismo preproceso de eliminación de duplicados al dataset de clasificación reducido, se pueden reducir de nuevo sus dimensiones hasta obtener un dataset sin ninguna relación de dependencia matemática.*

Al no existir estado del arte sobre este planteamiento, el siguiente objetivo de esta tesis consiste en proponer un modelo teórico que permita abordar la búsqueda de las mejores reglas de asociación de cualquier dataset de clasificación.

Por último, se marcó como objetivo demostrar que el modelo planteado se puede utilizar sobre muchos datasets de clasificación que presentan dificultades al ser analizados mediante técnicas clásicas de ARM. Se aplicó el mismo proceso a los 75 datasets de clasificación de un repositorio público, eliminando los duplicados que contenían. Se comprobó que todos los datasets preprocesados se pueden guardar en memoria RAM en forma de matriz, sin necesidad de utilizar un supercomputador. Con la tecnología actual es relativamente fácil calcular la dependencia matemática de las columnas de una matriz. Se comprobó que la mayoría de datasets procesados tenían atributos matemáticamente independientes de la clase definida en el problema de clasificación.

Organización de la tesis

Esta memoria se divide en siete capítulos y cuatro apéndices. Tras la visión general mostrada en este capítulo, en el capítulo 2 se expone el es-

tado del arte de la minería de reglas de asociación, el dilema del ítem raro, y la minería de reglas de clasificación asociativa.

En el capítulo 3 se presentan los resultados obtenidos en los primeros años de investigación de esta tesis. Publicaciones sobre la minería de reglas de asociación y aproximaciones a la minería de reglas de clasificación asociativa.

En el capítulo 4 se presenta una nueva metodología con la que descubrir información previamente desconocida de los datasets de clasificación. Se define el catálogo como un subconjunto de cualquier dataset de clasificación, con propiedades que pueden ser aprovechadas de un modo muy eficiente con la tecnología actual.

En el capítulo 5 se presentan los modelos teórico y aplicado diseñados para desarrollar la metodología propuesta. Se exponen las definiciones necesarias para desarrollar el modelo teórico. Y las propiedades, lemas y teoremas que permiten desarrollar el algoritmo $\mathcal{ACDC}$, con el que se obtiene, a partir de cada dataset de clasificación, una colección de catálogos que contienen la misma información para clasificación que el dataset original.

El capítulo 6 muestra los experimentos realizados con los 75 datasets de clasificación de *KEEL Standard Dataset Repository* (2004). Se trata de una colección heterogénea de datasets de clasificación ampliamente utilizados en el estado del arte revisado en esta memoria. Hay datasets de grandes dimensiones, datasets muy densos y datasets con atributos constantes, y en ningún caso han habido problemas de desbordamiento de memoria.

En el capítulo 7 se exponen las principales conclusiones de este trabajo. Al haber desarrollado el modelo teórico, quedan abiertas algunas líneas de investigación para trabajo futuro.

El apéndice A muestra información sobre la bibliografía utilizada en la presente memoria, el apéndice B muestra un listado con los acrónimos utilizados en esta memoria, el apéndice C proporciona información sobre los datos utilizados, y el apéndice D contiene el código utilizado para llevar a cabo los experimentos del capítulo 6.

UNIVERSITAS
Miguel
Hernández

2

Estado del Arte

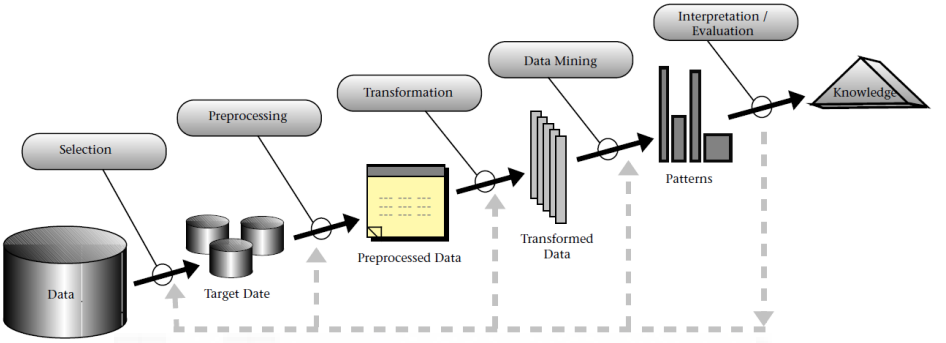
Piatetsky-Shapiro (1989), Piatetsky-Shapiro y Frawley (1991) y Frawley et al. (1992) definen el descubrimiento de conocimiento en bases de datos (KDD) años después de la irrupción del aprendizaje automatizado (del inglés, Machine Learning, ML) y la minería de datos (del inglés, Data Mining, DM). Durante este tiempo se constató que el número de bases de datos (del inglés, Data Base, DB) crecía a un ritmo muy superior al del conocimiento que los dispositivos informáticos eran capaces de obtener a partir de los ellas.

> *The growth in the amount of available databases far outstrips the growth of corresponding knowledge. This creates both a need and an opportunity for extracting knowledge from databases. Many recent results have been reported on extracting different kinds of knowledge from databases, including diagnostic rules, drug side effects, classes of stars, rules for expert systems, and rules for semantic query optimization.*

Fayyad, Piatetsky-Shapiro et al. (1996) describen el KDD como el proceso no trivial de identificar patrones de datos válidos, novedosos, potencialmente útiles y comprensibles. La figura 2.1, extraída de su artículo, muestra una idea del proceso completo de KDD, formado por cinco fases recurrentes que permiten convertir los *datos* en *conocimiento*. Una *selección* de los datos disponibles, convenientemente *preprocesados* y *transformados*,

pueden ser sometidos a técnicas de *minería de datos* que descubran los *patrones* que contienen para ser convertidos en *conocimiento*.

Figura 2.1: Fases del proceso de KDD [Fayyad et al. (1996)]



Este artículo hace una excelente reflexión sobre la relación existente entre KDD, DM, ML y *Estadística*, entre otras áreas dedicadas al análisis de grandes colecciones de datos.

Friedman (1997) considera el proceso de KDD como un análisis exploratorio de datos para grandes DBs. Otros autores describen a su modo las diferentes fases del proceso de KDD (Rokach y Maimon, 2014), siguiendo la propuesta de Fayyad, Piatetsky-Shapiro et al. El proceso no es trivial, y tiene cada vez más alternativas en sus diferentes fases, gracias a los avances proporcionados por la comunidad científica en múltiples áreas de conocimiento.

Un caso ampliamente estudiado es el análisis de los datos que genera la World Wide Web (WWW). Etzioni (1996) se preguntaba si la enorme cantidad de información que estaba generando la WWW, en su época, era un lodazal o una mina de oro. Se estaban adaptando las bases de la DM a la minería web (del inglés, Web Mining, WM). Las nuevas tecnologías están acrecentando el problema planteado por Etzioni. Cada día más, con la aparición y popularización de sistemas de comunicación inalámbrica que permiten comunicar fácilmente dispositivos de proceso, almacenamiento y visualización de datos entre sí, sea cual sea su ubicación. La WWW es el medio de generación, almacenamiento y distribución masiva de datos más representativo. Hoy en día sabemos que es una mina de oro, pero hay que

saber convertir en conocimiento la información que contienen los millones de datos que genera a cada instante. Este conocimiento puede ser utilizado por los medios informáticos de que dispone cada persona en muchas de sus tareas cotidianas.

La minería de uso web (del inglés, Web Usage Mining, WUM) emergía con fuerza para tratar de extraer conocimiento a partir de los datos generados por los usuarios de la web. Para analizar el comportamiento de los usuarios de la WWW, se buscan patrones entre los datos que genera su propio uso. El primer objetivo de esta tesis fue desarrollar un sistema de recomendación web (del inglés, Web Recommender System, WRS) capaz de aprender de los usuarios de un portal web. Los datos iniciales son los registros del archivo de log del servidor en que está alojado el portal. El conocimiento que se quiere adquirir es la *siguiente* página de interés para el usuario mientras esté navegando por el portal, teniendo en cuenta las páginas que ha recorrido en la misma sesión de navegación y en sesiones anteriores.

En los siguientes apartados se profundiza en las fases del KDD, sirviendo de ejemplo la investigación propia desarrollada en el área de WUM.

Selección

Cada vez hay más datos disponibles, y digitalizados, sobre casi cualquier población de la realidad que nos rodea. En cada instante se almacenan millones de datos recogidos por diferentes sensores, o introducidos manualmente en grandes DBs. Estamos en la era de la explosión de la Big Data, y las previsiones son de crecimiento exponencial para la cantidad de datos que se gestionan mediante equipos informáticos.

Las enormes DBs disponibles contienen conocimiento sobre demasiados aspectos en torno a la población de datos en estudio. No es operativo buscar información muy diversificada si aún no se sabe cómo extraer correctamente información más específica. Cuando se quiere utilizar una de estas DBs se ha de revisar qué tipo de información contiene y fijar un objetivo de investigación.

La primera fase del KDD consiste en la definición de un objetivo y la selección de los datos que pueden contener información sobre el objetivo fijado. Las DBs actuales tienen gran cantidad de tablas, registros y relaciones. Utilizando diferentes criterios se pueden seleccionar muchas co-

lecciones de datos distintas. Es habitual incorporar nuevas características a un experimento, filtrar ciertas características temporales o geoespaciales o usando otros criterios, incorporar conocimiento adquirido en otros procesos de KDD realizados sobre la misma DB o sobre otras.

Figura 2.2: KDD: Selección de datos

Cualquier combinación de estas circunstancias proporciona una selección de datos diferente, y el conocimiento que proporciona cada selección puede ser de mayor o menor calidad. Esta fase del proceso genera los datos con los que se intenta resolver el problema propuesto, pero puede modificarse de forma iterativa a partir de información obtenida en cualquiera de las otras fases debido a que el proceso es iterativo.

Por ejemplo, en la preparación de un WRS, se pueden seleccionar, del archivo de log del servidor, los datos de usuarios procedentes de una región concreta, o los que se han producido un día determinado de la semana durante los últimos meses, o los que se realizan antes de confirmar o desistir de una compra online…

Preproceso

El preproceso es una de las fases que más tiempo consume en cualquier proceso de KDD. No todos los valores registrados en los datos seleccionados contienen información relevante para el estudio a realizar. Se deben eliminar todos aquellos que sólo aportan ruido o dificultan innecesariamente la ejecución de las siguientes fases del proceso.

Estas decisiones deben ser tomadas por expertos en la materia final del estudio, su experiencia puede agilizar enormemente el tiempo de obtención de resultados y la calidad de los mismos. La evaluación final puede ayudar también a tomar decisiones en esta fase si se detecta nula influencia de un dato incorporado en esta fase.

En esta fase se decide qué hacer con los datos atípicos pues son datos que dificultan el estudio completo, decisión que aparecerá de nuevo en la fase de minería de datos. Pero su influencia puede ser esencial en algunas ocasiones. Hay investigadores que eliminan directamente estos datos atí-

picos, otros plantean un estudio exclusivo de estos datos. Sea cual sea la decisión tomada debería volverse a este punto una vez finalizado el proceso de KDD, para comprobar si puede explicarse de modo científico la existencia de datos atípicos, en ocasiones debidos simplemente a malas mediciones en el proceso de captación de datos.

Figura 2.3: KDD: Preproceso

También hay que tomar decisiones sobre los datos desconocidos, incluso provocando un estudio previo para hacer una predicción sobre estos datos y poder utilizar estas estimaciones en el resto del proceso. Si un atributo posee muchos datos desconocidos y se decide eliminar del estudio el atributo, se podría estar renunciando a información de extrema calidad. Si se utilizan estimaciones de los datos desconocidos pero son estimaciones incorrectas, el resultado final puede aparecer totalmente sesgado por esta circunstancia.

Siguiendo con el ejemplo de un WRS, en el preproceso se eliminan las solicitudes de páginas o recursos realizadas por robots, las que devuelven un código 404, las imágenes o vídeos que muestran las páginas visitadas por los usuarios…

Transformación

Una vez preprocesados los datos, se debe cambiar su aspecto para ser comprendidos mejor por el analista, y dotarles de un formato con el que puedan ser gestionados de forma más eficiente por los equipos informáticos. En esta fase se da a los datos seleccionados y preprocesados más aspecto de información que de datos.

En general, no se utilizan todos los datos preprocesados, están ahí para ser utilizados en diferentes estudios, o añadidos o eliminados en un estudio en particular, pero no para tratarlos todos de golpe cuando se trabaja con colecciones muy grandes de datos. Los datos pueden ser sometidos a una reducción de dimensiones mediante una selección y extracción de características o mediante el uso de muestreo. Un muestreo bien aplicado puede arrojar mucha información sobre el contenido del conjunto comple-

to de datos preprocesados, lo que ayudaría a decidir qué datos serán los que más conocimiento aporten.

La experiencia de los investigadores ayudará a decidir hechos como que «*el índice de masa corporal es más representativo en este estudio que la altura y peso del individuo por separado*». Aunque ese índice se calcula a partir de los otros dos datos, si en lugar de guardar en memoria dos valores se guarda sólo uno se ganan recursos en la máquina para hacer otros procesos o gestionar otros datos más relevantes para el estudio.

En un WRS se transforman las peticiones al servidor de cada usuario en *sesiones de navegación*, una secuencia de las páginas visitadas por el usuario en el portal web. Esta primera transformación hace más comprensibles los datos preprocesados, pero aún no se han adaptado al formato final que necesitan los algoritmos de DM que los van a analizar.

Los algoritmos de minería de datos consumen muchos recursos de tiempo y de memoria RAM, los datos que reciben han de estar preparados para reducir esta carga todo lo posible. Las cadenas de texto se pueden convertir en códigos, generalmente numéricos, más preparados para ser gestionados en memoria y para realizar ordenamientos y búsquedas. Los atributos numéricos, o los categóricos de rangos grandes, pueden discretizarse/categorizarse para reducir el número de datos a procesar.

Figura 2.4: KDD: Transformación

Botella et al. (2005b) transforman las sesiones de navegación en grafos, y también las convierten en transacciones (2005a) para aplicar algoritmos de minería de reglas de asociación (ver figura 2.4). Carmona et al. (2012) las convierten en registros de clasificación, y utilizan las reglas de clasificación obtenidas para sugerir mejoras en el diseño del portal web. Los atributos estudiados en este análisis recogen información generada por los usuarios del portal: navegador, tipo de visitante, palabras clave usadas, fuente…Todos los atributos se han dividido en un pequeño conjunto de valores representativos, y para formar los registros han anotado el valor concreto de cada atributo en una visita al portal.

Minería de datos

En esta fase se ha de decidir qué tarea de minería de datos se adapta mejor a los datos a procesar y al objetivo final del estudio: *clustering* (Ng y J. Han, 1994; Perkowitz y Etzioni, 1999a, 2000; Goethals, 2003; Saglam et al., 2006; Tsay, T.-J. Hsu et al., 2009), *clasificación* (B. Liu y W. Hsu, 1996; B. Liu, W. Hsu y Ma, 1998; Rokach y Maimon, 2014) o *regresión* (Kohavi y R. Quinlan, 1999). Los objetivos de la DM son básicamente *predicción* y *descripción*, y ambos están estrechamente relacionados. La *predicción* se engloba en las técnicas de DM supervisadas mientras que la descripción incluye técnicas no supervisadas de ML como *clasificación* o *visualización*. El nexo entre ambos objetivos es evidente en el análisis de grandes colecciones de datos: si no se realiza un buen proceso descriptivo de los mismos será difícil obtener un buen modelo de *predicción*.

En la minería de datos se debe plantear bien el problema a resolver. El proceso de DM es sólo una fase del proceso de KDD, una pequeña parte del engranaje (ver figura 2.1), pero si no se ejecuta correctamente no proporciona conocimiento válido y previamente desconocido. Una vez seleccionados, preprocesados y transformados los datos, y elegida la tarea de DM que se quiere aplicar, se ha de decidir con qué algoritmo se aborda esta fase y qué parámetros se ajustan mejor a los datos existentes y los resultados que se espera obtener.

Actualmente es una disciplina en la que trabajan investigadores de todas las ramas del saber. Colecciones masivas de datos (en aumento), dispositivos capaces de almacenarlas, distribuirlas y procesarlas y nuevas metodologías para su conversión en conocimiento, están al alcance de todos.

La *Estadística* y la *Informática* son las ciencias que más aportan a la minería de datos. No hay estudio de DM que no intente justificar la bondad de sus resultados con la *Estadística*, aunque las justificaciones utilizadas no están siempre basadas en un profundo conocimiento de este área. No sería posible la DM sin el concurso de la *Informática*, aunque muchas aportaciones no son realmente útiles a la comunidad científica hasta que se desarrollan con eficiencia.

En esta fase se trata de aprovechar la potencia de cálculo de los actuales dispositivos informáticos para analizar relaciones, similitudes y diferencias entre los datos transformados. Se observan desde distintas perspectivas para que el investigador o el propio sistema detecte patrones específi-

cos o extraños. Perkowitz y Etzioni (1998, 1999a, 1999b, 2000) utilizan estos patrones para dividir un portal web en clusters, conjuntos de páginas que son visitadas de forma conjunta y que son analizadas por el administrador del sitio web para evaluar los resultados y mejorar la experiencia de usuario, si las agrupaciones obtenidas son correctas. El *clustering* divide a la población en grupos cuyos individuos sean homogéneos entre sí, de modo que los diferentes clusters sean lo más heterogéneos posible entre sí, basándose en diferentes medidas de similitud. Schafer et al. (2001) dividen la gran población de usuarios de un portal de comercio electrónico en diferentes clusters, de modo que cuando se quiere hacer una recomendación a un usuario se consultará únicamente en el cluster con que mejor se identifique, El número de datos a procesar será mucho menor y los resultados pueden ser mucho más acertados si se ha hecho correctamente la asignación.

La *Estadística* es la base teórica de la minería de datos, por lo que ésta utiliza muchas de sus técnicas, como el muestreo aplicado a los datos originales de Ozmutlu et al. (2002), tras el que aseguran obtener un subconjunto manejable de datos con la misma información que el conjunto original.

La minería de reglas de asociación (ARM), es una técnica de DM propuesta por Agrawal, Imielinski et al. (1993b). Gran parte de la presente investigación gira en torno a la ARM y a la *clasificación* (Kohavi y R. Quinlan, 1999), que se fusionan en la minería de reglas de clasificación asociativa (del inglés, Classification Association Rules Mining, CARM) (B. Liu, W. Hsu y Ma, 1998). La *clasificación* se enfoca hacia la predicción y la ARM descubre asociaciones entre los valores de los atributos en estudio, lo que puede aprovecharse en algoritmos como CMAR (Thabtah et al., 2006) o AR+OPTBP (Kahramanli y Allahverdi, 2009).

En el análisis de un portal web, las sesiones de navegación se representan mediante secuencias, conjuntos de ítems ordenados que son fácilmente modelables como cadenas de Markov y estudiadas con n-gramas o analizadas con algoritmos adaptados a las secuencias como AprioriAll, AprioriSome o DynamicSome (Agrawal y Srikant, 1995) y GSP (Srikant y Agrawal, 1996b).

El análisis de sesiones de navegación puede estar dirigido a diferentes objetivos. En un WRS, el objetivo es la predicción de las páginas web que desean visitar nuestros usuarios. En esta línea se han propuesto di-

versos planteamientos sobre cómo sintetizar la información que contiene la generalmente enorme base de datos de sesiones de navegación. Entre otras soluciones se propone el uso de *Hypertext Probabilistic Grammar* (Borges y Levene, 1999), el uso de programación genética basado en gramática (Hervás-Martínez et al., 2004a; 2004b), el sistema de predicción `WhatNext` basado en n-gramas (Su et al., 2000) o el modelo `multi-step dynamic` n-`gram` (Sun et al., 2002).

En todos los casos se buscan patrones, secuencias repetidas en diferentes sesiones de navegación. Como se trabaja con grandes colecciones de datos, se utilizan criterios para reducir el número de patrones a descubrir y no desbordar las capacidades de los dispositivos que realizan el análisis.

Integración y evaluación

La evaluación e interpretación de los patrones descubiertos, teniendo en cuenta los objetivos planteados en el proceso completo, ha de poner el foco en la comprensión y utilidad del modelo obtenido. Se debe documentar correctamente en esta fase el conocimiento adquirido para poder utilizarlo en posteriores procesos. El primer problema que se presenta en esta fase es el exceso de conocimiento adquirido. Se ha cambiado una gran cantidad de datos iniciales por una gran cantidad de conocimiento ¿se puede analizar todo este conocimiento en profundidad? Un sistema informático puede ayudar al analista del proceso en este punto, como apuntan B. Liu y W. Hsu (1996), utilizando diferentes criterios para decidir qué conocimiento es más útil en el caso concreto que se está evaluando.

El objetivo final de cualquier proceso de KDD es la integración del conocimiento adquirido en otro sistema que sea capaz de obtener provecho del mismo. El proceso se ha realizado a partir de una «foto fija» de la población en estudio, los datos obtenidos y seleccionados en las primeras fases. Ahora han de integrarse en un sistema que sea capaz de traducir el conocimiento adquirido en beneficios para los usuarios y los propietarios del sistema (mayor usabilidad, personalización, recomendaciones más acertadas, ventas…). Esta transición no es necesariamente sencilla pero se escapa al proceso expuesto en este trabajo.

La última fase del proceso es la que da validez al mismo: integrar el conocimiento al sistema de modo que pueda ser usado, y evaluar los re-

sultados obtenidos. En este punto se ha de comprobar qué conocimientos se tenían al principio del proceso y qué conocimientos nuevos se han adquirido tras el análisis de los datos.

Al analizar el uso de un portal web, si se observa que un gran número de usuarios visitan conjuntamente las páginas A, F y H, el administrador del portal puede crear enlaces entre ellas, lo que mejoraría su rango de alcance. En un WRS se deberían presentar enlaces a las páginas F y H al usuario que visite la página A, aunque esta página no contenga enlaces a F y H. Para lograrlo se debe estudiar previamente el uso del sitio, detectando si un porcentaje considerable de usuarios que solicitan el recurso A también solicitan los recursos F y H en la misma sesión (Ng y J. Han, 1994). Existen diversas alternativas a esta solución (Su et al., 2000; Sun et al., 2002; X. Chen y X. Zhang, 2003), todos ellos con un esquema común: considerar la información que aportan todos los usuarios, analizarla exhaustivamente para obtener patrones de comportamiento de los usuarios, y resumirla de modo que con pocos recursos se pueda ayudar al máximo a cualquier usuario, mediante la predicción de la siguiente página a visitar.

Esta fase es fundamental para un sistema de recomendación web. Cuando empezó esta investigación tenía acceso al servidor de un portal web usado por los alumnos de mi universidad. La planificación de esta tesis incluía la puesta en marcha de un WRS en dicho portal. Sin embargo, el servidor dejó de funcionar antes de terminar mi periodo de investigación, y no pude llevar a cabo la evaluación e integración planificada. En el resto del capítulo se expone la fase de investigación llevada a cabo en torno al área de WUM y RSs. Representa la base teórica de esta tesis que, unida a la base práctica que adquirí al desarrollar la investigación expuesta en el capítulo 3, me ayudó a tener suficientes conocimientos para descubrir el método de análisis de catálogos descrito en los capítulos 4, 5 y 6.

2.1. Minería de Reglas de Asociación

Agrawal, Imielinski et al. (1993a) son pioneros en la minería de reglas de asociación. La primera definición sobre este área de la ciencia se basa en el análisis hecho por el departamento de planificación de un supermercado en el conocido como ejemplo de la *cesta de la compra*.

Consider a supermarket setting where the database records items pur-

chased by a customer at a single time as a transaction. The planning department may be interested in finding "associations" between sets of items with some minimum specified confidence. An example of such an association is the statement that 90% of transactions that purchase bread and butter also purchase milk. The antecedent of this rule consists of bread and butter and the consequent consists of milk alone. The number 90% is the confidence factor of the rule. Usually, the planner will be interested not in a single rule but rather in sets of rules satisfying some initial specifications. Here are some other examples of the problem of finding associations (we have omitted the confidence factor specification):

- *Find all rules that have "Diet Coke" as consequent. These rules may help plan what the store should do to boost the sale of Diet Coke.*
- *Find all rules that have "bagels" in the antecedent. These rules may help determine what products may be impacted if the store discontinues selling bagels.*
- *Find all rules that have "sausage" in the antecedent and "mustard" in the consequent. This query can be phrased alternatively as a request for the additional items that have to be sold together with sausage in order to make it highly likely that mustard will also be sold.*
- *Find all the rules relating items located on shelves A and B in the store. These rules may help shelf planning by determining if the sale of items on shelf A is related to the sale of items on shelf B.*
- *Find the "best" k rules that have "bagels" in the consequent. Here, "best" can be formulated in terms of the confidence factors of the rules, or in terms of their support, i.e., the fraction of transactions satisfying the rule.*

Note that a transaction need not necessarily consist of items bought together at the same point of time. It may consist of items bought by a customer over a period of time. Examples include monthly purchases by members of a book or music club.

Según su propia experiencia, tras generar gran número de reglas de asociación a partir de gran cantidad de transacciones de un supermercado descubrieron que «*los viernes, muchos de los clientes compran pañales y cervezas*», lo que les llevó a sugerir al propietario del comercio que los viernes promocionara ambos productos de manera conjunta. La base científica de las reglas de asociación es la *Estadística Descriptiva*, sin embargo hacer un planteamiento similar con métodos estadísticos no nos llevaría a descubrir algo tan sorprendente ya que en *Estadística* se ha de formular la hipótesis antes de observar los datos, y llegar a formular concretamente esa hipótesis resulta difícil de imaginar.

2.1.1. Modelo teórico

Agrawal, Imielinski et al. (1993b) proporcionan la primera definición formal de esta disciplina.

> *Let $\mathcal{I} = \{I_1, I_2, \ldots I_N\}$ be a set of binary attributes, called items. Let $\mathcal{D}$ be a database of transactions. Each transaction t is represented as a binary vector, with $t[k] = 1$ if t bought the item I_k, and $t[k] = 0$ otherwise. There is one tuple in the database for each transaction. Let X be a set of some items in $\mathcal{I}$. We say that a transaction t satisfies X if for all items I_k in X, $t[k] = 1$.*
>
> *By an association rule, we mean an implication of the form*
>
> $$X \rightarrow I_j$$
>
> *where X is a set of some items in $\mathcal{I}$, and I_j is a single item in $\mathcal{I}$ that is not present in X. The rule $X \rightarrow I_j$ is satisfied in the set of transactions $\mathcal{D}$ with the confidence factor $0 \leq c \leq 1$ iff at least $c\%$ of transactions in $\mathcal{D}$ that satisfy X also satisfy I_j.*
>
> *Given the set of transactions $\mathcal{D}$, we are interested in generating all rules that satisfy certain additional constraints of two different forms: Syntantic Constrains and Support Constrains.*

Partiendo del conjunto $\mathcal{I}$, formado por N atributos binarios, definen $\mathcal{D}$ como una base de datos de transacciones, vectores binarios de N elementos cuyo valor k-ésimo indica si el ítem I_k está o no presente en la transacción.

Una regla de asociación (del inglés, Association Rule, AR) es una expresión del tipo $X \rightarrow I_j$, donde X es un itemset, un conjunto de ítems de $\mathcal{I}$ al que se denomina antecedente, e $I_j \notin X$ recibe el nombre de consecuente. Las investigaciones posteriores sobre esta área amplían esta definición para estudiar reglas en que el consecuente sea un conjunto de ítems que no pertenezcan al antecedente.

Definen la confianza de la regla de asociación $X \rightarrow I_j$ como el porcentaje de transacciones de $\mathcal{D}$ que, conteniendo a X, también contienen a I_j. El factor de confianza c, más conocido como *confianza mínima*, es un valor que se fija para restringir el número de reglas descubiertas al considerar sólo aquellas que tienen confianza igual o superior a c.

Finalmente, definen la minería de reglas de asociación como la disciplina que busca en un gran almacén $\mathcal{D}$ las ARs que, además de tener confianza mínima, cumplan ciertas restricciones, bien de tipo sintáctico –como la presencia de cierto ítem en el antecedente o el consecuente– o de soporte–obteniendo únicamente las reglas más frecuentes–.

Las restricciones no son imposiciones si no necesidades. Si $\mathcal{I}$ está formado por N ítems, para tratar una transacción hay que utilizar N valores, y se podrán obtener hasta $2^N - N - 1$ itemsets distintos con al menos 2 ítems. Cada uno de estos itemsets puede generar hasta $\sum_{i=1}^{k-1} \binom{k}{i}(2^{k-i} - 1)$ reglas distintas, siendo k el número de ítems que contiene.

Si $N = 100$, hay $2^{100} - 101 \approx 1.3 \cdot 10^{30}$ posibles itemsets en las transacciones, lo que da una idea del número de reglas de asociación que se pueden encontrar entre los itemsets de $\mathcal{D}$ cuando éste es muy grande y el número de ítems distintos también lo es.

Formalizando su propuesta se podrán desarrollar diferentes modelos necesarios para el desarrollo de esta memoria. El punto de partida es un conjunto de N ítems, denotado con $\mathcal{I}$. Una transacción es un subconjunto de $\mathcal{I}$, obtenido en determinadas circunstancias, y se dispone de un gran conjunto de transacciones. Los patrones buscados son los conjuntos de ítems que están presentes simultáneamente en las transacciones. Las transacciones también reciben el nombre de itemset, indicando que son simplemente un conjunto de ítems.

Definición 2.1 (Población de ítems)**.** *Sea $\mathcal{I}$ un conjunto con N* ítems *distintos. Llamaremos itemset a cualquier subconjunto de $\mathcal{I}$, o bien k-itemset si queremos*

indicar en su nombre su tamaño.

$$\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_N\} \tag{2.1}$$

En la cesta de la compra, $\mathcal{I}$ es el conjunto de todos los productos existentes en el comercio. En una tarea de *clasificación* se suelen codificar todos los pares `{atributo=valor}` mediante números enteros consecutivos, con lo que $\mathcal{I}$ contiene únicamente un conjunto de números que representan todos los valores que pueden tomarse en los diferentes atributos en estudio. En minería de uso web, $\mathcal{I}$ es el conjunto de páginas del sitio web.

Definición 2.2 (Itemset)**.** *Llamaremos itemset a cualquier subconjunto de $\mathcal{I}$. Con k-itemset se indica que contiene k ítems.*

$$X \subset \mathcal{I} \tag{2.2}$$

Definición 2.3 (Transacción)**.** *Una transacción $\mathcal{T}$ es un itemset, un subconjunto de $\mathcal{I}$, obtenido en determinadas circunstancias.*

$$\mathcal{T} \subseteq \mathcal{I} = \{I_j \mid I_j \in \mathcal{I},\ j = 1 \ldots k\} \tag{2.3}$$

En la cesta de la compra, una transacción es parte de la información recogida en un ticket, el conjunto de productos que ha adquirido el cliente en una compra. También se podrían crear las transacciones con el conjunto de productos que han comprado los clientes identificados durante el último mes, con lo que habría menos transacciones y se podrían buscar patrones de compra entre nuestros clientes, no entre cada una de sus compras individuales. En un problema de *clasificación*, cada registro que agrupa los valores que toma un individuo para cada uno de los atributos en estudio, es una transacción. En WUM, una transacción es el conjunto de páginas que ha visitado un usuario en una sesión de navegación, o en todas las compras realizadas a lo largo del tiempo en un comercio electrónico.

Definición 2.4 (Dataset)**.** *Un dataset, $\mathcal{D}$, es un conjunto de transacciones.*

$$\mathcal{D} = \{\mathcal{T}\} \tag{2.4}$$

Se ha definido el dataset de transacciones, $\mathcal{D}$, para diferenciarlo de la base de datos que contiene todos los datos recogidos. La minería de reglas de asociación, como técnica de DM, sólo es una fase de un proceso de KDD. El dataset $\mathcal{D}$ se obtiene después de seleccionar, pre-procesar y transformar los datos originales. Cualquier cambio en los parámetros usados en estas fases provoca la obtención de un dataset $\mathcal{D}$ diferente.

En la cesta de la compra se puede crear $\mathcal{D}$ con todos los tickets de un día, de una semana, de un día concreto de la semana durante los últimos 3 meses... En un problema de *clasificación* se puede crear $\mathcal{D}$ utilizando sólo unos atributos de la base de datos original, o sólo los registros de determinadas fechas. En WUM se puede obtener $\mathcal{D}$ a partir de las sesiones de navegación, usando criterios de geolocalización de la IP o la identificación del usuario si ha iniciado sesión privada en el servidor.

No tiene sentido en muchos casos intentar aplicar técnicas de minería de datos a colecciones extremas de datos, a todo el historial de tickets o al conjunto de todas las sesiones de navegación de un sitio web. Si el dataset $\mathcal{D}$ es tan grande se tiene que renunciar a mucha de la información que posee. Esta investigación busca, entre otros objetivos, aprender a extraer el máximo de conocimiento de un gran conjunto de datos, utilizando equipos informáticos al alcance de todos los investigadores.

Al poder crear $\mathcal{D}$ de un modo rápido y con diferentes criterios, se abre el campo de posibilidades de uso de las reglas de asociación. Si se puede generar un dataset $\mathcal{D}$ con información específica para un análisis y extraer de él todas las reglas de asociación representativas en un tiempo adecuado para el estudio que estamos haciendo, se pueden ajustar los análisis utilizando la información proporcionada por los propios datos, que pueden volver a ser generados con nuevos criterios y analizados para obtener conocimiento de mayor calidad o para confirmar o descartar el conocimiento previo al análisis. En poblaciones muy variables, el proceso de KDD se retroalimenta constantemente del conocimiento descubierto en iteraciones previas.

El objetivo de ARM es encontrar las ARs que contiene el dataset $\mathcal{D}$, reglas con la forma $X \rightarrow Y$, que informan sobre el número de veces que aparece el itemset Y en las transacciones que contienen al itemset X.

Definición 2.5 (Regla de Asociación)**.** *Una regla de asociación es una expresión del tipo $X \rightarrow Y$, donde X e Y son dos itemsets mutuamente excluyentes*

($X \cap Y = \emptyset$).

$$R_i = \{X \rightarrow Y\} \tag{2.5}$$

X recibe el nombre de antecedente, Y el de consecuente.

Las reglas de asociación se obtienen al examinar a fondo un dataset de transacciones, y lo único que nos dirán es qué relaciones de co-ocurrencia contiene entre los ítems en estudio. La aportación de la minería de reglas de asociación es la posibilidad real de hacer el análisis de tremendas colecciones de transacciones. Algo tan aparentemente simple se ha de llevar a la práctica con cierta prudencia pues los recursos disponibles en la mayoría de dispositivos informáticos son limitados. A pesar de contar con muchas herramientas derivadas de la teoría de conjuntos, al tratar de implementarlas en un dispositivo informático y manipular grandes cantidades de datos, aparecerán problemas de desbordamiento de memoria, por lo que se entiende que la ARM sea una disciplina más investigada en el ámbito de la *Informática* que en el de la *Estadística*.

Definición 2.6 (Minería de Reglas de Asociación)**.** *La minería de reglas de asociación es el proceso de búsqueda de reglas de asociación en grandes almacenes de transacciones, utilizando los recursos informáticos disponibles y en un tiempo apropiado.*

El análisis propuesto es útil para muchas disciplinas en la época de la tecnología y el *Big Data*. Las transacciones sirven para modelar gran cantidad de estudios y las reglas de asociación puede convertirse en conocimiento muy útil en distintas áreas de investigación, si se adquiere a tiempo de poder ser utilizado.

- La cesta de la compra se puede generalizar a cualquier tipo de comercio, recibiendo mucha atención en el ámbito del *e-commerce*, donde muchas recomendaciones se hacen en base al conocimiento adquirido por el uso de reglas de asociación.

- En muchos estudios de clasificación se codifican los datos observados a cada individuo en un registro con pares `{atributo=valor}` y se realiza el proceso de clasificación observando la co-ocurrencia de los datos de los distintos registros. Las reglas de asociación proporcionan justo esta información por lo que han generado el estudio de

las reglas de clasificación (B. Liu, W. Hsu y Ma, 1998; Thabtah et al., 2006; Kahramanli y Allahverdi, 2009).

- Cuando un médico solicita diferentes análisis está adquiriendo datos del paciente, un conjunto de pares `{atributo=valor}` que puede ser observado como una transacción si los valores son categóricos o se pueden categorizar. Si pudiera comparar esta transacción con las transacciones de otros pacientes a los que se está estudiando o ya han sido diagnosticados, tendría información de mayor calidad para poder emitir un diagnóstico o solicitar un nuevo análisis.
- En un sondeo socio-político con respuestas categóricas, se podría tratar cada encuesta individual como una transacción. La interpretación del sondeo se beneficia del descubrimiento de alguna regla de asociación difícil de encontrar usando métodos puramente estadísticos.
- En minería de uso web se pueden convertir las sesiones de navegación en transacciones y estudiar qué páginas relacionan entre sí los usuarios. Si se pueden obtener las «mejores» reglas de asociación en tiempo real se podrá usar esta información para mejorar un sistema de recomendación web.

Para medir la calidad de las ARs se utilizan dos valores, el soporte y la confianza de la regla. Antes de definirlas se ha de exponer el significado de soporte de un itemset, que siempre tendrá relación con el tamaño del dataset de transacciones, $|\mathcal{D}|$.

Definición 2.7 (Soporte de un itemset)**.** *El soporte de un itemset es su frecuencia en $\mathcal{D}$, el número de veces que aparece el itemset en el dataset de transacciones $\mathcal{D}$.*

$$soporte(X) = |\mathcal{T}_X| = n_X, \;\; \textit{siendo} \;\; \mathcal{T}_X = \{\mathcal{T} \in \mathcal{D} \;/\; X \in \mathcal{T}\} \tag{2.6}$$

Se utilizan las dos versiones de la frecuencia para hablar del soporte. En la anterior ecuación aparece el soporte absoluto y a continuación se muestra el soporte relativo, puesto en relación al número de transacciones con que estamos trabajando.

$$soporte_{relativo}(X) = \frac{n_X}{|\mathcal{D}|} \tag{2.7}$$

En la mayoría de estudios se supone que el soporte de un itemset es representativo de la frecuencia de ese itemset en la población de procedencia de la muestra analizada. Si esta suposición fuera correcta, sería más probable encontrar este itemset en esta población que cualquier otro con menor soporte, por lo que se utiliza para estimar probabilidades sobre la población de origen de la muestra $\mathcal{D}$.

La limitación de recursos en los dispositivos en que se ejecutan los algoritmos de ARM obliga a hacer alguna definición más. El principal problema es el uso de memoria RAM cuando hay muchos ítems sobre los que guardar información. De $\mathcal{I}$ se pueden obtener $2^N - N - 1$ itemsets distintos con más de un ítem, y de cada k-itemset se pueden obtener hasta $2^k - 2$ reglas de asociación, por lo que si N es grande se han de almacenar millones de datos y frecuencias, tantos que pueden llegar a desbordar la capacidad de almacenamiento del dispositivo en que se ejecute el algoritmo de ARM.

La primera solución propuesta consiste en ignorar los ítems que tengan menor soporte, para aprovechar la memoria sólo para guardar los ítems más frecuentes, lo que se supone que aportará información para un conjunto más amplio de individuos de la población. Es una suposición que no debe satisfacer las inquietudes del analista, ya que cuando se trabaja con colecciones muy grandes de transacciones puede decidirse ignorar los ítems cuyo soporte sea inferior al 0.5 %, lo que muchos investigadores califican como «poco representativo» de la población en estudio. Si ese aparentemente pequeño 0.5 % supone que se ignore la información que proporciona un ítem que aparece en miles de transacciones, no se debería dar por terminado el estudio.

Definición 2.8 (Soporte mínimo)**.** *El soporte mínimo, $minSup$, es un valor fijado antes de llevar a cabo la búsqueda de itemsets frecuentes. Si el dataset $\mathcal{D}$ es muy grande y el equipo en que se ejecuta el algoritmo de minería de reglas de asociación tiene recursos insuficientes para el análisis completo de $\mathcal{D}$, se debe fijar un soporte mínimo para que no se produzca desbordamiento de memoria RAM, lo que se logrará ignorando los itemsets cuyo soporte sea inferior a $minSup$.*

Definición 2.9 (Ítem raro)**.** *El ítem x se denomina ítem raro cuando su soporte es inferior al umbral de soporte mínimo fijado.*

$$x \text{ es un ítem raro} \Leftrightarrow soporte(x) < minSup$$

Definición 2.10 (Itemset frecuente)**.** *Un itemset frecuente es el que no contiene ítems raros.*

$$X \text{ es frecuente} \Leftrightarrow soporte(x) \geq minSup, \forall x \in X$$

Definición 2.11 (Conjunto de itemsets frecuentes)**.** *Sea $\mathcal{L}$ el conjunto de todos los itemsets frecuentes de $\mathcal{D}$. Sean $\mathcal{L}_k$ los conjuntos de k-itemsets frecuentes de $\mathcal{D}$.*

$$\mathcal{L} = \cup_k \mathcal{L}_k \tag{2.8}$$

Técnicamente, el valor más grande que puede tomar k coincide con la longitud de la transacción más larga de $\mathcal{D}$. Sin embargo, al trabajar con soporte mínimo es muy probable que k no alcance dicho valor.

Los primeros algoritmos de ARM propuestos por Agrawal, Imielinski et al. (1993b) se diseñaron en una época en que los recursos de memoria RAM de los ordenadores empezaban a incrementarse, pero a precios muy altos y poco accesibles por la mayoría de investigadores. Esta circunstancia obligó a definir conjuntos auxiliares para tener mayor control sobre la memoria RAM utilizada.

Definición 2.12 (Conjunto de candidatos a itemset frecuente)**.** *$\mathcal{C}_k$ es el conjunto de k-itemsets de $\mathcal{D}$ que podrían ser frecuentes.*

Uno de los retos de esa época fue dotar a los algoritmos la capacidad de determinar el conjunto $\mathcal{C}_k$ óptimo para ahorrar el máximo número de operaciones de reserva y liberación de memoria RAM. Hoy en día, este aspecto ha pasado a segundo plano y la mayoría de investigadores no lo exponen en sus trabajos.

Definición 2.13 (Soporte de una regla de asociación)**.** *El soporte de la regla de asociación $X \rightarrow Y$ es la frecuencia del itemset que la forma, $X \cup Y$.*

$$soporte(X \rightarrow Y) = soporte(X \cup Y) = n_{XY} \tag{2.9}$$

$$soporte_{relativo}(X \rightarrow Y) = \frac{soporte(X \cup Y)}{|\mathcal{D}|} = \frac{n_{XY}}{|\mathcal{D}|} \tag{2.10}$$

El soporte de las reglas de asociación de un $\mathcal{D}$ es uno de los criterios de ordenación de reglas más utilizado.

Definición 2.14 (Confianza de una regla de asociación)**.** *La confianza de una regla es la frecuencia con que aparece el consecuente, Y, en las transacciones en que está el antecedente, X.*

$$confianza(X \rightarrow Y) = \frac{soporte(X \cup Y)}{soporte(X)} = \frac{n_{XY}}{n_X} \tag{2.11}$$

Si la regla de asociación $X \rightarrow Y$ tiene soporte s y confianza c, su lectura es sencilla de comprender:

> «En el s % de las transacciones de $\mathcal{D}$ está el itemset $(X \cup Y)$. En el c % de las transacciones en que está presente el itemset X, también está presente el itemset Y»

Se utilizan mucho en tareas de predicción, interpretando la confianza como una probabilidad, aunque para hacerlo correctamente debería estudiarse más a fondo la población de origen de los datos y la representatividad de $\mathcal{D}$.

Definición 2.15 (Confianza mínima)**.** *La confianza mínima, $minConf$, es un valor fijado por el analista para generar sólo las reglas de asociación de mayor confianza.*

En todo proceso de ARM hay dos fases bien diferenciadas, la minería de itemsets frecuentes (en inglés, Frequent Itemsets Mining, FIM) y la obtención de las reglas de asociación a partir de los itemsets frecuentes encontrados. La parte más costosa del proceso es la primera pues se ha de recorrer por completo el dataset $\mathcal{D}$ para comprobar qué ítems se relacionan entre sí y cuáles lo hacen de manera frecuente, superando el soporte mínimo fijado en el estudio. En esta fase se guarda la información obtenida conforme se va leyendo $\mathcal{D}$, con el riesgo de tener desbordamiento de memoria RAM si no se hacen costosas operaciones de comprobación cada vez que se va a necesitar algo más de RAM. En la sección 2.1.3 se exponen diversos trabajos sobre FIM.

En la segunda fase, se utiliza el conjunto de itemsets frecuentes encontrado en la primera y se obtienen todas las reglas que se pueden derivar de ellos, ofreciendo como resultado aquellas que superen la confianza mínima fijada para el estudio. El número de reglas de asociación que puede contener un dataset es tan grande que, una vez descubiertas, hay que ordenarlas en función del interés que tiene cada una de ellas.

Las dos métricas más usadas para definir el interés de una regla de asociación son el soporte y la confianza, considerándose más relevantes las reglas con alto soporte y, en caso de empate, las que tengan mayor confianza. En algunas investigaciones se usan otras métricas que pueden ayudar a ordenar las reglas obtenidas, como *lift* y *conviction*. Todas ellas dependen de la métrica soporte.

Definición 2.16 ($lift$)**.** *El $lift$ de una regla de asociación compara el soporte de la regla con el soporte del consecuente. Si llamamos n_{XY} al número de veces que aparece el itemset $(X \cup Y)$ en $\mathcal{D}$, su $lift$ sería*

$$lift(X \to Y) = \frac{soporte(X \to Y)}{n_Y} = \frac{n_{XY}}{n_X \cdot n_Y} \tag{2.12}$$

Su interpretación se hace en términos de probabilidad. Si hubiera independencia estadística entre los sucesos A («la transacción contiene el itemset X») y B («la transacción contiene el itemset Y») podríamos comprobarlo a partir de las probabilidades $P(A)$, $P(B)$ y $P(A \cap B)$ debido a que dos sucesos independientes cumplen que $P(A \cap B) = P(A) \cdot P(B)$. Utilizando las frecuencias observadas en $\mathcal{D}$ podríamos pensar que dos sucesos son independientes si el producto de sus frecuencias coincide con la frecuencia conjunta de ambos. Utilizando n_X, n_Y y n_{XY} para expresar estas frecuencias, si detectamos que $n_X \cdot n_Y = n_{XY}$ podemos pensar que la presencia de Y en las transacciones de $\mathcal{D}$ es independiente de que también esté presente X. Si ocurriera esto tendríamos que $lift(X \to Y) = \frac{n_{XY}}{n_X \cdot n_Y} = 1$ por lo que el $lift$ debe compararse con la unidad para que nos proporcione alguna información.

Supóngase que X e Y son conjuntos de páginas de un sitio web y al analizar las sesiones de navegación de sus usuarios se descubre la regla de asociación $X \to Y$.

- si $lift(X \to Y) > 1$ se deduce que el conjunto de páginas Y es más popular entre los que ya han visitado X que por sí misma, por lo que es más probable que un usuario que ya ha visitado X quiera visitar Y. El hecho de que el usuario haya visitado X «favorece» la probabilidad de que quiera visitar Y en la misma sesión de navegación.
- Si $lift(X \to Y) < 1$, la presencia de X disminuye la probabilidad de que aparezca Y en la misma sesión de navegación. Quizá no sea buena idea recomendar Y a un usuario que ya ha visitado X.

- Si $lift(X \rightarrow Y) = 1$, la presencia de X no influye en la probabilidad de que aparezca Y en la misma sesión de navegación, son dos sucesos independientes. Quizá no sea buena idea recomendar Y a un usuario que ya ha visitado X, a no ser que no tengamos otra recomendación mejor.

La $conviction$ (Brin et al., 1997) mide la influencia del antecedente sobre el consecuente de una regla de asociación.

Definición 2.17 ($conviction$)**.** *Si llamamos $n_{X\neg Y}$ al número de itemsets que contienen a X y no contienen a Y en $\mathcal{D}$, $n_{X\neg Y} = soporte(X \cup \neg Y)$, la conviction de la regla de asociación $(X \rightarrow Y)$ es*

$$conviction(X \rightarrow Y) = \frac{n_{X\neg Y}}{n_X \cdot n_{\neg Y}} \tag{2.13}$$

$conviction$ toma valores entre 1 y $+\infty$. Si $conviction(X \rightarrow Y) = 1$ interpretamos que ambos itemsets son independientes, no influye el antecedente para que aparezca más o menos veces el consecuente. Cuanto mayor es el valor de $conviction$, más influye el antecedente sobre el consecuente.

2.1.2. Bases de datos de transacciones, datasets

La mayoría de experimentos de minería de datos surgen por la necesidad de extraer información contenida en grandes bases de datos, el formato más habitual para guardar y gestionar grandes colecciones de datos. Sin embargo, los análisis no se realizan sobre todos los datos recogidos en cada DB. En el KDD son fundamentales las primeras fases de *selección*, *preproceso* y *transformación* para obtener un dataset que pueda ser analizado por un algoritmo concreto de DM.

La minería de reglas de asociación es una fase del proceso de descubrimiento de conocimiento en bases de datos, por lo que es fácil pensar que los algoritmos de ARM se analiza el contenido de bases de datos. En los primeros artículos sobre ARM se trabajaba en esta dirección, pero se abandona conforme avanzan las investigaciones y se descubre la necesidad de eficiencia al tratar con grandes colecciones de datos.

En el caso de ARM, se analizan datasets de transacciones. La elección del tipo de archivo utilizado para guardar las transacciones es fundamental para conseguir eficiencia en la ejecución de sus algoritmos. Se pueden

usar archivos binarios específicos para dar mayor eficiencia a alguna implementación concreta. J. Zhang et al. (2004) y Sheng Zhang et al., 2005 extraen el dataset $\mathcal{D}$ de archivos XML. H. Li y H. Chen (2009) descubren itemsets frecuentes no-derivables en datos distribuidos por streaming. Para una mejor comunicación entre diferentes equipos es necesario utilizar algún tipo de archivo estándar, destacando los archivos de texto plano como los más genéricos y, por tanto, más fáciles de compartir entre diferentes comunidades científicas.

Agrawal, Imielinski et al. (1993b) proponen dotar a los gestores de bases de datos (en inglés, Data Base Management System, DBMS) de nuevas operaciones eficientemente implementadas para poder hacer minería de datos sobre clasificación, asociación y secuencias. Con ellas, los investigadores podrían trabajar directamente sobre DBs utilizando combinaciones de dichas operaciones, y obtener patrones válidos para los tres modelos abordados. Houtsma y Swami (1995) proponen el uso de funciones ya implementadas en lenguajes nativos de DBMSs, `SQL` concretamente.

Otro trabajo que busca el uso de DBMSs para extraer reglas de asociación es el de Holsheimer et al. (1995). En su favor está la flexibilidad que permiten los DBMSs para tratar los datos desde múltiples perspectivas, en su contra el tiempo necesario para obtener resultados frente a los algoritmos específicos de ARM, desarrollados mediante lenguajes de programación compilado. Una de las propuestas de este artículo es el uso de la jerarquía de los ítems para reducir el tamaño del dataset a analizar, concretamente hablan de la cesta de la compra y de descubrir reglas que involucren genéricamente al ítem «cerveza» en lugar de usar como ítem cada una de las marcas de cerveza que están en $\mathcal{I}$, idea que será tomada por otros investigadores para reducir las dimensiones del problema y poder llevar a cabo estudios más amplios.

Al definir las transacciones como vectores binarios se puede interpretar que es mejor guardar los datos a nivel de bit, ya que sólo son necesarios dos valores para representar cada dato, lo que convertiría al dataset $\mathcal{D}$ en una matriz de ceros y unos, cuyas $|\mathcal{D}|$ filas representan cada una de las transacciones y sus N columnas cada uno de los ítems de $\mathcal{I}$. Es fácil pensar en trabajar con matrices de estas características para aprovechar la potencia de los ordenadores al trabajar a nivel de bit, sin embargo no se plantea nadie este formato hasta el trabajo de Dong y M. Han (2007), que proponen comprimir $\mathcal{D}$ en una BitTable y usar el algoritmo BitTableFI. Indican

que esta estructura posibilita una rápida generación de candidatos y de su recuento por utilizar funciones básicas de unión e intersección de bits, obteniendo mejor rendimiento que los algoritmos con que se compara.

Trabajar con uniones e intersecciones de bits puede ser muy eficiente para un procesador si se consigue programar de forma óptima, sin embargo, al desarrollar un programa con un lenguaje de alto nivel no es fácil trabajar a nivel de bit –en `C++` se utiliza un Byte, ocho bits, para guardar un valor `bool`– por lo que la mayoría de implementaciones hechas sobre algoritmos de ARM no utilizan esta definición de $\mathcal{I}$.

Generalmente, se representa el ítem I_k utilizando el número k, con lo que las transacciones no se guardan como N-tuplas de ceros y unos si no como conjuntos de números enteros.

$$\{1, 1, 0, 1, 0, 0, 1\} \Rightarrow \{1, 2, 4, 7\}$$

El formato más utilizado para guardar $\mathcal{D}$ de este modo es mediante archivos de texto plano en que cada transacción es una línea del archivo, lo que se conoce como *representación horizontal de* $\mathcal{D}$. Algunos algoritmos están especialmente diseñados para aprovechar la *representación vertical de* $\mathcal{D}$, en que en cada línea se escribe el par {`TID` $item$}, de modo que cada transacción ocupa tantas líneas consecutivas como ítems contenga. A lo largo de esta memoria se tratará a $\mathcal{D}$ como un conjunto de números enteros y no una matriz de bits.

Aunque los algoritmos presentados de forma teórica a lo largo de esta investigación no especifican el formato físico de los datos a procesar, las comparaciones realizadas sobre los distintos algoritmos no sería correcta si no se experimentara en condiciones similares, por lo que en la mayoría de los casos encontraremos referencias a datasets públicos recogidos en diferentes repositorios. Los datasets disponibles en *FIM Data Repository* (2004), *UCI Machine Learning Repository* (2013) y *KEEL Standard Dataset Repository* (2004) tienen formato de texto plano, la mayoría de ellos con representación horizontal.

2.1.3. Algoritmos

El aspecto más estudiado en este área de conocimiento es la modelización de estrategias que permitan obtener el máximo conocimiento posible

en el menor tiempo posible, usando siempre los limitados recursos disponibles. Para ello se han presentado gran cantidad de algoritmos que, partiendo de un dataset $\mathcal{D}$, descubren las «mejores» reglas de asociación que contiene.

La revisión de los algoritmos para ARM publicados por diferentes investigadores ayuda a comprender las dificultades reales que presenta la búsqueda de reglas de asociación en grandes coleccione de datos. Cada nuevo algoritmo intenta resolver algún problema que aún no se había detectado, o resolver problemas conocidos de un modo más eficiente.

Agrawal, Imielinski et al. (1993b) presentan el algoritmo AIS (ver listado 2.1), que se basa en múltiples lecturas de la base de datos de transacciones, el dataset $\mathcal{D}$. Comienzan con un conjunto vacío de itemsets frecuentes, $\mathcal{L}$, y un conjunto de candidatos a itemset frecuente al que llaman frontera, $\mathcal{C}$. En la primera lectura de $\mathcal{D}$ se crea $\mathcal{C}_1$ contando el soporte de todos los ítems que contiene, se guardan en $\mathcal{L}_1$ los que tienen soporte mínimo y se añaden a $\mathcal{C}_2$ todas las combinaciones de los ítems de $\mathcal{L}_1$. En cada nueva lectura de $\mathcal{D}$, $k = 2\ldots$, se realizan los siguientes pasos:

1. Se busca cada $(k-1)$-itemset de la frontera en cada transacción.

2. Si se encuentra, se extiende con todos los ítems de la transacción y se añade la extensión al conjunto $\mathcal{C}_{k+1}$, creándolo con soporte 1 si no existe o incrementando su soporte si ya existe.

3. Al terminar la k-ésima lectura de $\mathcal{D}$ se vacía la frontera, se guardan en $\mathcal{L}_k$ los candidatos que tienen soporte mínimo y se añaden a la frontera los que se estima que se podrán extender en la siguiente iteración.

4. Si la frontera no está vacía, se vuelve al paso 1.

Empiezan a encontrarse los problemas de recursos intrínsecos a esta disciplina, llegando a proponer un algoritmo que guarda en disco la información que está en memoria y no es necesaria. Este algoritmo se llamará cada vez que se necesite memoria para almacenar nuevos datos, lo que restará algo de eficiencia al proceso completo.

Proponen también una serie de heurísticas para estimar el soporte de un k-itemset antes de generar su candidato, de modo que si se estima que

Listado 2.1: Algoritmo AIS, 1993

```
procedure LargeItemsets
begin
  let Large set ℒ = ∅;
  let Frontier set ℱ = {∅};

  while ℱ ≠ ∅ do begin
    -- make a pass over the database
    let Candidate set 𝒞 = ∅;
    forall database tuples t do
      forall itemsets f in ℱ do
        if t contains f then begin
          let 𝒞_f = candidate itemsets that are extensions of f
              and contained in t;
          forall itemsets c_f in 𝒞_f do
            if  c_f ∈ 𝒞 then
              c_f.count = c_f.count + 1;
            else begin
              c_f.count = 0;
              𝒞 = 𝒞 + c_f;

      -- consolidate
      let ℱ = ∅;
      forall itemsets c in 𝒞 do begin
        if count(c)/dbsize > minsupport then
          ℒ = ℒ + c;
        if c should be used as a frontier in the next pass then
          ℱ = ℱ + c;
end
```

tendrá un soporte bajo no se genere el candidato y el sistema no caiga por falta de memoria para almacenar los candidatos. La generación de candidatos se hace bajo la suposición de independencia de los ítems presentes en cada transacción. Si los itemsets X e Y son disjuntos y tienen soporte x e y respectivamente, estiman que el itemset $X \cup Y$ tendrá soporte $z = x \cdot y$, por lo que esperan que será frecuente en $\mathcal{D}$ si $z \geq minSup$.

Houtsma y Swami (1995) sugieren trabajar directamente sobre las bases de datos de transacciones aprovechando las funciones propias de los DBMS en lugar de desarrollar aplicaciones específicas para trabajar con almacenes $\mathcal{D}$ guardados como texto plano. Proponen el algoritmo SETM (ver algoritmo 2.2), que genera los candidatos al leerlos en las transacciones, igual que AIS, y guarda una copia de cada itemset junto al `TID` de la transacción que lo contiene. Al trabajar con DBs en disco no necesita mu-

chos recursos de memoria para guardar tanta información, pero no puede competir en eficiencia con programas preparados específicamente para extraer la misma información de un archivo de texto plano.

Listado 2.2: Algoritmo SETM, 1993

```
k := 1;
sort R_1 on item;
C_1 := generate counts on R_1;
repeat
  k := k + 1;
  sort R_{k-1} on trans_id, item_1, ..., item_{k-1};
  R'_k := merge-scan R_{k-1}, R_1;
  sort R'_k on item_1, ..., item_{k-1};
  C := generate counts on R'_k;
  R_k := filter R'_k to retain supported patterns;
until R_k = {}
```

Como se está buscando la co-existencia de ítems en transacciones, se puede aprovechar el orden lexicográfico del código utilizado para representar a cada ítem. El soporte del itemset XY coincide con el de YX por lo que si al guardar los itemsets se hace con sus ítems ordenados lexicográficamente, sólo se guardará el itemset XY. Este orden se va a aprovechar para hacer más eficientes las operaciones a realizar por el algoritmo.

Listado 2.3: Algoritmo SETM, función merge-scan

```
INSERT INTO R'_k
SELECT p.trans_id, p.item_1, ..., p.item_{k-1}, q.item
FROM R_{k-1} p, SALES q
WHERE q.trans_id = p.trans_id AND
      q.item > p.item_{k-1}
```

Las funciones del algoritmo son consultas a la base de datos. Para obtener R'_k se ejecuta la consulta del listado 2.3, que extiende cada k-itemset frecuente encontrado en el paso anterior con los ítems frecuentes de $\mathcal{D}$ que sean lexicográficamente mayores que el mayor de los ítems del k-itemset. Concretamente, en la extensión de un k-itemset basta con añadir uno a uno los ítems de la transacción que contienen al k-itemset y son lexicográficamente mayores al mayor de los ítems del k-itemset.

$\mathcal{C}$ se obtiene contando los itemsets del conjunto R'_k y guardando sólo los que tienen soporte mínimo, como muestra el listado 2.4.

Listado 2.4: Algoritmo SETM, selección de candidatos

```
INSERT INTO C_K
SELECT p.item_1, ..., p.item_k, COUNT(*)
FROM R'_k p
GROUP BY p.item_1, ..., p.item_k
HAVING COUNT (*) >= min_support
```

El conjunto auxiliar R_k se obtiene seleccionando las tuplas de R'_k que pueden ser extendidas, como muestra el listado 2.5.

Listado 2.5: Algoritmo SETM, itemsets extendibles

```
INSERT INTO R_k
SELECT p.trans_id, p.item_1, ..., p.item_k
FROM R'_k p , C_k q
WHERE p - item_1 = q.item_1 AND
      ...
      p - item_k = q.item_k
ORDER BY p.trans_id, p.item_1, ..., p.item_k
```

Estos primeros algoritmos de ARM utilizan la misma estrategia para evitar desbordamiento de memoria al generar los candidatos a itemset frecuente. Como el número de candidatos es teóricamente muy grande cuando trabajamos con grandes almacenes $\mathcal{D}$ y muchos ítems en $\mathcal{I}$, y como es de esperar que el número de itemsets en $\mathcal{D}$ sea sensiblemente menor que el teórico, en estos algoritmos se propone reservar memoria únicamente para contar los itemsets que realmente están en $\mathcal{D}$, es decir, se reserva memoria cada vez que se encuentra un itemset en $\mathcal{D}$ para el que aún no hemos hecho dicha reserva.

Este planteamiento consigue su objetivo a costa de eficiencia, cada una de las millones de operaciones de reserva de memoria que se llevan a cabo consume un tiempo, su acumulación resulta en un tiempo de ejecución muy superior al que se conseguiría haciendo una buena estimación de los itemsets contenidos en $\mathcal{D}$ y realizando la reserva de memoria en una única instrucción.

En ambos artículos se trata por encima la generación de reglas de asociación. En su definición se especifica que el consecuente es un único ítem por lo que una vez conocidos todos los k-itemsets frecuentes basta con recorrerlos uno a uno y separarlos en dos itemsets, el antecedente con $k-1$ ítems y el consecuente con el ítem restante. Se descubren todas las ARs de $\mathcal{D}$ y se seleccionan las que tienen soporte mínimo.

Agrawal, Imielinski et al. (1993b) definen dos fases bien diferenciadas en el proceso de ARM:

1. En primer lugar hay que encontrar los conjuntos de ítems que están presentes en un porcentaje mínimo de transacciones de $\mathcal{D}$.

2. A continuación se descubren las reglas de asociación que se deducen de esos conjuntos.

La primera fase es la que mayores aportaciones científicas ha recibido en la ARM pues al trabajar con grandes colecciones de transacciones puede consumir más recursos de los disponibles o tardar más tiempo del adecuado para dar por finalizado el análisis. Cabe destacar que la tecnología utilizada para el desarrollo de los algoritmos expuestos en esta sección ha evolucionado notablemente desde los inicios del ARM, las comparaciones realizadas a través de los artículos presentados pueden haber variado debido a que actualmente tenemos a nuestra disposición recursos de memoria y de cómputo muy superiores a los utilizados en las dos décadas precedentes.

La búsqueda de los itemsets frecuentes presentes en $\mathcal{D}$ es uno de los cuellos de botella de la ARM. Es una fase en que los consumos de recursos se han de optimizar, sobre todo las necesidades de memoria RAM y el uso de procedimientos con alta carga de proceso. Es tal su importancia que ha creado su propia disciplina, la minería de itemsets frecuentes (FIM).

La atención mostrada por la comunidad científica a la búsqueda de algoritmos y estructuras de datos específicos para FIM se revela en la gran cantidad de trabajos publicados. Debido al gran volumen de información que se ha de manipular, se han propuesto todo tipo de representaciones de los almacenes $\mathcal{D}$, destinados a reducir su volumen sin perder la información que contienen. Se ha propuesto el uso de estructuras especializadas o de taxonomías, de técnicas basadas en el muestreo de los almacenes $\mathcal{D}$ para la extracción de la máxima cantidad de conocimiento oculto, o técnicas basadas en computación en paralelo para lograr mayor velocidad en la

ejecución o un reparto de carga de los recursos de memoria. La *Informática* proporciona a los investigadores la posibilidad de emular cualquier situación, con la única limitación de los recursos de tiempo y de memoria para llevar a cabo el proceso.

Agrawal y Srikant (1994) proponen el algoritmo más estudiado en ARM, Apriori (ver algoritmo 2.6), y dos alternativas que se adaptan mejor a colecciones concretas de transacciones, AprioriTID (ver algoritmo 2.10) y AprioriHybrid.

Listado 2.6: Algoritmo Apriori, 1994

```
L_1 = {large 1 − itemsets};
for (k = 2; L_{k−1} ≠ ∅; k++) do begin
  C_k = apriori − gen(L_{k−1}); //Ver funciones 2.7 y 2.8
  forall transactions t ∈ D do begin
    C_t = subset(C_k, t);
    forall candidates c ∈ C_t do
      c.count++;
  end
  L_k = {c ∈ C_k / c.count ≥ minSup};
end

Answer = L = ⋃_k L_k
```

La clave de Apriori está en la generación de candidatos, la fase del algoritmo que más recursos consume y que más incide en la eficiencia del resto de operaciones del algoritmo. Su propuesta se basa en la siguiente propiedad, que es la que da nombre al algoritmo.

Propiedad 2.1 (A priori)**.** *Si un candidato a k-itemset contiene un (k-1)-itemset que no es frecuente, a priori sabemos que no puede ser frecuente.*

Gracias a esta observación, en su algoritmo sólo se generan los candidatos a k-itemset frecuente cuyos subconjuntos de $k-1$ ítems sean todos frecuentes. Esto supone un uso mayor de procesador en esta fase (tiempo de ejecución) en beneficio de un menor uso de memoria principal, ya que se reduce notablemente el número de itemsets que teóricamente se pueden obtener del conjunto $\mathcal{I}$ de ítems en estudio.

La principal diferencia entre Apriori y los dos primeros algoritmos expuestos radica en la generación de candidatos. En Apriori no se requiere una lectura de $\mathcal{D}$ para obtener los candidatos pues se obtienen a partir

de los itemsets frecuentes encontrados en una lectura previa. La función `apriori-gen`($\mathcal{L}_{k-1}$) es la encargada de generar los candidatos a k-itemset frecuentes, $\mathcal{C}_k$. Se ejecuta en dos fases, en primer lugar se generan todos los posibles candidatos y a continuación se eliminan aquellos que contienen algún (k-1)-itemset infrecuente.

Listado 2.7: Algoritmo Apriori, función `unión`

```
insert into C_k
  select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
    from L_{k-1}p, L_{k-1}q
    where  p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2},
           p.item_{k-1} < q.item_{k-1};
```

Listado 2.8: Algoritmo Apriori, función poda

```
forall itemsets do
  forall (k-1)-subsets s of c do
    if (s ∉ L_{k-1}) then
      delete c from C
```

La separación de las funciones de generación y poda hacen más legible el algoritmo (ver funciones 2.7 y 2.8), pero su implementación es poco eficiente pues requiere una reserva de memoria excesiva en la primera fase. En Botella et al. (2005a) se propone una solución eficiente a este problema uniendo ambas funciones en una sola (véanse las funciones 3.1, 3.2 y 3.3).

Una vez descubierto el conjunto $\mathcal{L}$ de itemsets frecuentes, se usa la función `genrules` (ver función 2.9) para generar las reglas de asociación con confianza superior al mínimo fijado por el analista.

En el mismo artículo, Agrawal y Srikant proponen el algoritmo AprioriTID (ver algoritmo 2.10), basado en la representación vertical de $\mathcal{D}$. Este cambio de representación de $\mathcal{D}$ surge por la poca eficiencia de la representación horizontal usada en Apriori para algunos bancos de transacciones específicos. Esta observación sigue presente en muchas investigaciones y pone de manifiesto que no existe un algoritmo «mejor» que todos para cualquier dataset de transacciones. En un intento de lograr este objetivo proponen en el mismo artículo el algoritmo Apriori-Hybrid, que combina características de Apriori y AprioriTID, aprovechando que el primero es

Listado 2.9: Algoritmo APRIORI, función genrules

```
forall large itemsets l_k, k ≥ 2 do
   call genrules(l_k , l_k);

// The genrules generates all valid rules ã → (l_k − ã), ∀ã ⊂ a_m
procedure genrules(l_k: large k−itemset,
                   a_m: large m−itemset)
1)  A = {(m − 1)-itemsets a_{m−1} | a_{m−1} ⊂ a_m};
2)  forall a_{m−1} ∈ A do begin
3)    conf = support(l_k)/support(a_{m−1});
4)    if (conf ≥ minconf) then begin
7)      output the rule a_{m−1} ⇒ (l_k − a_{m−1}),
          with confidence = conf and support = support(l_k);
8)      if (m − 1 > 1) then
9)        call genrules(l_k , a_{m−1}); // to generate rules with subsets
                                        // of a_{m−1} as the antecedents
10)   end
11) end
```

más eficiente para valores pequeños de k y que conforme se va incrementando su valor mejora la eficiencia del segundo.

Listado 2.10: Algoritmo APRIORITID, 1994

```
L_1 = {large 1−itemsets};
C̄_1 = database \D;
for (k = 2; L_{k−1} ≠ ∅; k++) do begin
  C = apriori − gen(L_{k−1}); //New candidates
  C̄ = ∅;
  forall entries t ∈ C̄_{k−1} do begin
    // determines candidate itemsets in C_k contained
    // in the transaction with identifier t.TID
    C_t = {c ∈ C_k|(c − c[k]) ∈ t.set−of−itemsets ∧(c − c[k − 1]) ∈ t.set−of−itemsets};
    forall candidates c ∈ C_t do begin
      c : count++;
    end
    if C_t ≠ ∅ then
      C̄_k += < t.TID, C_t >;
  end
  L_k = {c ∈ C | c : count ≥ minsup};
end

Answer = ⋃_k L_k;
```

También plantean algunas mejoras en la búsqueda de reglas de aso-

ciación, aunque se trata de la fase menos problemática de la ARM y serán expuestas en la sección 3.1.2.

De forma independiente a la presentación de Apriori, Mannila et al. (1994) observan que la generación de candidatos puede ralentizar o incluso abortar la ejecución del algoritmo de ARM y proponen un análisis que permite eliminar los candidatos innecesarios para el estudio. Comparan su algoritmo, OCD, con AIS (ver listado 2.1), indicando a partir de sus experimentos que mejora notablemente su rendimiento. Basan su reducción de candidatos a procesar en un análisis combinatorio de los k-itemsets frecuentes obtenidos en la k-ésima lectura de $\mathcal{D}$, una forma compleja de obtener los mismos candidatos que proporciona Apriori. También proponen el uso de muestreo para obtener las reglas de asociación mediante el análisis de parte de $\mathcal{D}$, sin embargo lo justifican pobremente y sólo dicen que se obtienen buenos resultados en algunas pruebas que han hecho. Mencionan continuamente el trabajo de Agrawal y Srikant (1994) insistiendo en que obtienen resultados similares de forma totalmente independiente, partiendo del conjunto C'_k, un superconjunto del generado por Apriori y coincidente con el obtenido en la fase de unión. Dos años más tarde presentan conjuntamente Agrawal, Mannila et al. (1996), una revisión conjunta de sus trabajos.

Listado 2.11: Algoritmo DHP (fase 1), 1995

```
s = a minimum support;
set all the buckets of H2 to zero; /* hash table */
forall transaction t ∈ D do begin
  insert and count 1−items occurrence in a hash tree;
  forall 2−subsets x of t
    H2[h2(x)]++;
end
L1 = {c | c.count > s, cisinaleafnodeofthehashtree};
```

Park et al. (1995; 1997) hacen una propuesta muy interesante para reducir el número de candidatos: leer los (k+1)-itemsets de cada transacción mientras se hace el recuento de $\mathcal{C}$ y convertirlos en un entero mediante una función hash, si al finalizar la lectura actual de $\mathcal{D}$ algún código hash no tiene soporte mínimo, no es necesario crear candidatos para los (k+1)-itemsets que producen ese código hash.

Listado 2.12: Algoritmo DHP (fase 2), 1995

```
k = 2;
D_k = D /* database for large k-itemsets */
while (| {x|H_k[x] ≥ s} | ≥ LARGE) {
  /* make a hash table */
  gen_candidate(L_{k-1}, H_k, C);
  set all the buckets of H_{k+1} to zero;
  D_{k+1} = ∅;
  forall transactions t ∈ D_k do begin
    count_support(t, C, k, t̂); /* t̂ ⊆ t */
    if |t̂| > k do begin
      make_hasht(t̂, H_k, k, H_{k+1}, ẗ);
      if |ẗ| > k then D_{k+1} = D_{k+1} ∪ {ẗ};
    end
  end
  L_k = {c ∈ C_k|c.count ≥ s};
  k++;
}
```

Proponen el algoritmo DHP (ver listados 2.11, 2.12 y 2.13) que reduce notablemente el número de candidatos generados por Apriori. La propuesta es muy acertada, pero en ciertas circunstancias puede provocar una ralentización del algoritmo ya que obliga en cada lectura de $\mathcal{D}$ a la codificación de un gran número de (k+1)-itemsets, sin retener más información que su número hash (para evitar un uso excesivo de memoria utilizan una función hash no unívoca, i.e., dos (k+1)-itemsets distintos pueden generar el mismo número hash, por lo que si encuentran que un número hash determinado tiene soporte mínimo no pueden garantizar que los itemsets que generan dicho número tenga también soporte mínimo).

Si el número de ítems distintos de $\mathcal{D}$, $|\mathcal{I}|$, es muy grande, y las transacciones no tienen una longitud excesiva, este algoritmo permite reducir drásticamente el soporte mínimo fijado por el analista, umbral que ha de ser grande en este tipo de datasets debido a la magnitud que puede alcanzar el número de candidatos a 2-itemsets. Los autores indican que en posteriores niveles este número se reduce por las propias características del dataset por lo que podría prescindirse de la costosa generación de códigos hash.

J. Han y Fu (1995) extienden el estudio de reglas de asociación añadiendo información a la base de datos de transacciones. Proponen añadir al estudio una taxonomía de los ítems del servicio de modo que puedan

Listado 2.13: Algoritmo DHP (fase 3), 1995

```
gen_candidate(L_{k-1}, H_k, C);
while (|C| > 0) {
  D_{k+1} = ∅;
  forall transactions t ∈ D_k
    count_support(t, C, k, t̂); /* t̂ ⊆ t */
    if (|t̂| > k) then D_{k+1} = D_{k+1} ∪ {t̂};
  end
  L_k = {c ∈ C | c.count ≥ s};
  if (|D_{k+1}| = 0) then break;
  C_{k+1} =apriori_gen(L_k) /* refer to Agrawal and Srikant (1994) */
  k++;
}
```

obtener información a múltiples niveles, reduciendo mediante agrupación el número de ítems a procesar y pudiendo obtener mejor información en el análisis. La primera lectura de $\mathcal{D}$ trata a todos los ítems según su pertenencia al máximo nivel de la taxonomía usada, de modo que la mayoría de ítems, que son pocos en comparación con el total al estar agrupados, tengan soporte mínimo y puedan seguir siendo analizados. En la siguiente lectura de $\mathcal{D}$ sólo se consideran las transacciones que contienen ítems frecuentes al primer nivel de la taxonomía, tratándolos ahora como ítems de un nivel inferior. Se prosigue con sucesivas lecturas de $\mathcal{D}$ y niveles inferiores en la taxonomía, hasta detenerse cuando no se encuentran nuevos k-itemsets. Proponen los algoritmos ML_T2L1, ML_T1LA, ML_TML1 y ML_T2LA.

Conforme van aumentando las propuestas sobre ARM se incrementan notablemente los datasets de transacciones existentes, con lo que los algoritmos van perdiendo su eficiencia y los analistas se ven obligados a incrementar el soporte mínimo para poder llevar a cabo el estudio. Para resolver el problema de la magnitud, Savasere et al. (1995) proponen el algoritmo Partition (ver algoritmo 2.14), que comienza dividiendo $\mathcal{D}$ en n particiones y realiza el análisis en varias fases, utilizando un soporte mínimo inferior al fijado por el analista, pues es de esperar que el soporte de cada itemset en una partición será menor a su soporte en $\mathcal{D}$.

La fase I del algoritmo realiza n iteraciones sobre $\mathcal{D}$, una por cada partición creada. Durante la iteración i sólo se considera la partición p_i. La función `gen_large_itemsets` toma una partición p_i como input y genera

Listado 2.14: Algoritmo PARTITION, 1995

```
P = partition_database(D)
n = number of transactions
for i = 1 to n do begin // Phase I
  read_in_partition(p_i ∈ P)
  L^i = gen_large_itemsets(p_i)
end
for (i = 2; L_i^j ≠ ∅, j = 1, 2, ..., n; i++) do
  C_i^G = ⋃_{i=1,2,...,n} L_i^j // Merge Phase
for i = 1 to n begin // Phase II
  read_in_partition(p_i ∈ P)
  forall candidates c ∈ C^G gen_count(c, p_i)
end
L^G = {c ∈ C^G | c.count ≥ minSup}

Answer = L^G;
```

los k-itemsets locales de todos los tamaños, ${\mathcal{L}_1}^i, {\mathcal{L}_2}^i, \ldots, {\mathcal{L}_k}^i$ como output. En la fase de mezcla son combinados todos los itemsets frecuentes de la misma longitud de las n particiones. El conjunto de candidatos globales de longitud j se obtiene con $C_j^G = \bigcup_{i=1}^{n} {\mathcal{L}_j}^i$. En la fase II, el algoritmo cuenta el soporte de todos los candidatos globales en las n particiones y obtiene todos los itemsets con soporte mínimo. El algoritmo lee $\mathcal{D}$ una vez en la fase I y otra vez en la fase II, lo que supone un gran ahorro en el proceso de lectura de datos del disco.

Mueller (1995) hace un buen análisis de los algoritmos existentes sobre ARM y propone el uso de una nueva estructura de datos, `SPTID`, y los algoritmos SEAR , SPEAR, SPINC y una versión paralela de los dos primeros.

Toivonen (1996) propone un método de muestreo con el que encontrar todas las reglas de asociación de una base de datos con una única lectura de la misma, si no se busca excesiva precisión, y con una segunda lectura si se quiere mayor precisión. La idea consiste en guardar en memoria una muestra representativa de $\mathcal{D}$ y obtener los itemsets frecuentes según un soporte mínimo inferior al planteado en el estudio. Al trabajar sólo con parte de $\mathcal{D}$ y poder alojarlo en memoria se gana mucho en eficiencia, pudiendo obtenerse de forma muy rápida los principales patrones presentes en $\mathcal{D}$ y, si se considera necesario, se puede hacer una segunda lectura de $\mathcal{D}$ para analizar las transacciones no presentes en la muestra inicial. El uso de muestras de $\mathcal{D}$ para obtener información preliminar de las reglas de

asociación que contiene se ha mostrado eficiente en muchos trabajos de este área, pero en ninguno de ellos se analiza la representatividad de la muestra ya que proceder a su análisis conllevaría un gasto de tiempo que iría en contra de uno de los principales propósitos de los algoritmos de ARM, su inmediata respuesta.

Agrawal y Shafer (1996) hacen su incursión en la ejecución de algoritmos de ARM en paralelo, con el fin de repartir entre varios procesadores la enorme carga de cómputo y memoria requeridos para su proceso cuando trabajamos con grandes DBs de transacciones. Proponen tres algoritmos - Count Distribution, Data Distribution y Candidate Distribution - con los que ponen a prueba las dificultades de comunicación entre procesadores, uso de memoria compartida y sincronización de memorias locales. Se trata de un interesante artículo de referencia si se quiere paralelizar la obtención de reglas de asociación. En los tres algoritmos parten de una misma base: dividen $\mathcal{D}$ en particiones disjuntas que son repartidas entre los N equipos que realizarán el proceso, usando procesadores, memoria principal y en disco totalmente independientes y compartiendo únicamente los resultados necesarios.

El algoritmo Count Distribution se limita a realizar en cada paso la generación y recuento de candidatos de la partición que le corresponde, combinando los resultados al final del mismo. En esta aproximación no se aprovecha la posibilidad de utilizar memoria compartida, ya que cada equipo utiliza su propia memoria para evaluar el árbol completo de itemsets frecuentes. Para aprovechar el incremento de memoria principal al añadir nuevos equipos proponen el algoritmo Data Distribution, cuya primera fase es idéntica a la del algoritmo Count Distribution, pero a continuación cada procesador hace el recuento de candidatos mutuamente excluyentes con el resto de procesadores. Al compartir todos el mismo dataset $\mathcal{D}$, se debe utilizar un equipo que permita una rápida comunicación entre procesadores. Ambos algoritmos requieren de una sincronización de todos los procesadores al finalizar cada paso para intercambiar recuentos o itemsets frecuentes, lo que causa paradas de procesadores si el flujo de trabajo no está correctamente balanceado.

El algoritmo Candidate Distribution pretende resolver este problema al no requerir sincronización hasta el final del proceso. En el paso l, determinado heurísticamente, el algoritmo divide $\mathcal{L}_{l-1}$ entre los procesadores de modo que el procesador P^i pueda generar un único $\mathcal{C}_m^i(m \geq l)$

independiente del resto de procesadores ($\mathcal{C}_m^i \cap \mathcal{C}_m^j = \emptyset, i \neq j$). Simultáneamente, las transacciones se replican selectivamente de modo que cada procesador pueda hacer el recuento de $\mathcal{C}_m^i$ de forma independiente al resto de procesadores. Hasta el paso l se utilizará el algoritmo Count Distribution o Data Distribution, a partir de $k = l$ se procede con el algoritmo Candidate Distribution.

Brin et al. (1997) proponen un nuevo algoritmo para el minado de reglas de asociación, DIC, que reduce el número de lecturas de la base de datos y genera un número menor de candidatos a itemset que los algoritmos basados en muestras. Se basan en ir creando un árbol similar a $\mathcal{L}$, leyendo $\mathcal{D}$ en bloques de M transacciones, de los que va deduciendo qué itemsets pueden ser frecuentes y cuáles no. También hablan de reglas de implicación, que no sólo miden la co-ocurrencia de ítems en transacciones, considerando lo que ellos llaman «verdadera implicación» entre antecedentes y consecuentes normalizados. Proponen ideas sobre el reordenamiento de los ítems presentes en $\mathcal{D}$ para lograr mayor eficiencia. Introducen el concepto de *conviccción* (ver definición 2.17) como alternativa a la confianza, obteniéndola a partir de $P(A) \cdot P(\neg B)/P(A, \neg B)$.

Srikant y Agrawal (1997) proponen el algoritmo de extracción de reglas de asociación Basic (ver algoritmo 2.15), que analiza datasets de transacciones con ítems jerarquizados en una taxonomía. Este enfoque, a diferencia de futuros trabajos que también usan taxonomías, no reduce el número de ítems a procesar en la fase de FIM pues considera como ítems tanto a los ítems de $\mathcal{D}$ como a sus ancestros en la jerarquización, completando todas las transacciones de $\mathcal{D}$ con los ancestros de cada uno de sus ítems, sin duplicados. Realmente se aumenta el conjunto de ítems a considerar, $\mathcal{I}$, los ítems poco frecuentes desaparecerán del estudio pero quedarán en las transacciones sus ancestros, de modo que siempre se puede obtener información sobre ellos, aunque sea a otro nivel de la taxonomía utilizada.

Una vez utilizado el algoritmo Basic, exponen algunas optimizaciones, que dan lugar al algoritmo Cumulate (ver algoritmo 2.16).

1. Filtrado de ancestros añadidos a las transacciones. No añaden a las transacciones todos los ancestros de sus ítems ya que alguno de ellos, o incluso el propio ítem, no forman parte de los candidatos a estudiar.

2. Pre-cómputo de ancestros. En lugar de recorrer la taxonomía para

Listado 2.15: Algoritmo Basic, 1997

```
L_1 := {frequent 1-itemsets};
k := 2; // k represents the pass number
while (L_{k-1} ≠ ∅) do begin
  C := New candidates of size k generated from L_{k-1};
  forall transactions t ∈ D do begin
    Add all ancestors of each item in t to t, removing any duplicate;
    Increment the count of all candidates in C that are contained in t;
  end
  L_k := All candidates in C with minimum support;
  k := k + 1;
end
Answer := ⋃_k L_k;
```

conocer los ancestros de un ítem se crea una tabla de ancestros para cada ítem.

3. Eliminación de itemsets que contienen un ítem y su ancestro. Justificado en los siguientes lemas:

   Lema El soporte de un itemset X que contiene tanto el ítem x como su ancestro $\hat{x}$ coincide con el soporte del itemset $X - \hat{x}$.

   Lema Si $\mathcal{L}_k$, el conjunto de k-itemsets frecuentes, no incluye ningún itemset que contenga tanto un ítem como su ancestro, el conjunto de candidatos $\mathcal{C}_{k+1}$ generado no contendrá ningún itemset que contenga tanto un ítem como su ancestro.

También presentan el algoritmo EstMerge (ver algoritmo 2.17), que en cada lectura de $\mathcal{D}$ hace un muestreo previo basado en el subconjunto $\mathcal{D}_S$ para obtener una mejor estimación de los candidatos a k-itemset, $\mathcal{C}_k$. Esta lectura extra de ciertas transacciones de $\mathcal{D}$ inicialmente consumen algo más de recursos y de tiempo, sin embargo cuando se lee $\mathcal{D}$ para obtener $\mathcal{L}_k$ no se buscará ninguno de los candidatos que no han superado el soporte mínimo corregido para la muestra (con la muestra se utiliza un soporte mínimo reducido por el factor 0.9), con lo que el tiempo empleado con la muestra se recupera ampliamente al ahorrar muchísimas operaciones de búsqueda y conteo de candidatos.

Listado 2.16: Algoritmo CUMULATE, 1997

```
Compute T*, the set of ancestors of each item, from T // Optimization 2
L1 := {frequent 1-itemsets};
k := 2 // k represents the pass number
while (Lk-1 ≠ ∅) do begin
  C := New candidates of size k generated from Lk-1;
  if (k = 2) then // Optimization 3
    Delete any candidate in C2 that consists of an item and its ancestor;
  Delete any ancestors in T* that are not present in any of the
      candidates in C // Optimization 1
  forall transactions t ∈ D do begin
    forall item x ∈ t do
      Add all ancestors of x in T* to t;
    Remove any duplicates from t;
    Increment the count of all candidates in C that are contained in t;
  end
  Lk := All candidates in C with minimum support;
  k := k + 1;
end
Answer := ⋃k Lk;
```

Listado 2.17: Algoritmo ESTMERGE, 1997

```
L1 := {frequent 1-itemsets};
Generate DS, a sample of the database, in the first pass;
k := 2; // k represents the pass number
C''1 := ∅; // C''k represents candidates of size k to
        // be counted with candidates of size k+1
while (Lk-1 ≠ ∅ or C''k-1 ≠ ∅) do begin
  Ck := New candidates of size k generated from Lk-1 ⋃ C''k-1;
  Estimate the support of the candidates in C by making a pass over DS;
  C'k := Candidates in C that are expected to have minimum support and
       candidates all of whose parents are expected to have minimum
           support;
  Find the support of the candidates in C'k ∪ C''k-1 by making a pass over D;
  Delete all candidate in C whose ancestors in C'k do not have minimum
      support;
  C''k := Remaining candidates in C that are not in C'k;
  Lk := All candidates in C'k with minimum support;
  Add all candidates in C''k-1 with minimum support to Lk-1;
  k := k + 1;
end
Answer := ⋃k Lk;
```

Srikant, Vu et al. (1997) proponen tres nuevos algoritmos, REORDER, MULTIPLEJOINS y DIRECT, para introducir restricciones sobre los ítems a pro-

cesar dentro de los algoritmos existentes de ARM. Justifican su trabajo en que a menudo el analista necesita hacer un filtrado de reglas de asociación en función de la presencia o ausencia de cierto ítem –o de su ancestro o descendiente en una taxonomía– y que es mucho más eficiente aplicar ese filtro antes de proceder con el algoritmo de ARM que después de haberlo hecho.

M. J. Zaki et al. (1997) proponen cuatro nuevos algoritmos, Eclat, MaxEclat, Clique y MaxClique. Todos ellos se basan en el clustering de itemsets y en el esquema *divide y vencerás*. En todos se propone leer $\mathcal{D}$ sólo una vez, con el ahorro de tiempo que ello supone. Eclat utiliza la representación vertical de $\mathcal{D}$ y genera un número de candidatos mayor que Apriori debido a que no guarda el soporte de todos los itemsets leídos. Su principal ventaja sobre Apriori está en que sólo realiza una lectura de $\mathcal{D}$, lo que supone una ejecución más rápida en general en una época en que el acceso a disco era aún demasiado costosa.

Bayardo (1998) afronta el problema de la presencia de k-itemsets largos en $\mathcal{D}$, que dificultan su análisis con los métodos tradicionales, citando básicamente Apriori. Propone el algoritmo Max-Miner para extraer únicamente los itemsets frecuentes *maximales*, aquellos que no tienen un superconjunto que sea también frecuente. Dado que todo itemset frecuente es un subconjunto de un itemset frecuente maximal, la salida de su algoritmo representa implícitamente y de modo conciso a todos los itemsets frecuentes presentes en $\mathcal{D}$. El ahorro que supone el almacenamiento de sólo los itemsets frecuentes maximales frente a Apriori proporciona a su algoritmo mayor rapidez de ejecución en algunas DBs, sin embargo este ahorro puede perderse cuando queremos obtener información más detallada sobre algunos itemsets frecuentes que no sean maximales.

Holt y Chung (1999) aplican los algoritmos Apriori y DHP al minado de DBs de texto y proponen los algoritmos Multipass-Apriori, M-Apriori y Multipass DHP (M-DHP) específicos para tratar DBs de texto usando técnicas de ARM.

J. Han, Pei et al. (2000) introducen una nueva estructura, FP-Tree, para guardar los itemsets presentes en $\mathcal{D}$, y con ella el algoritmo FP-Growth, con el que evitan la generación de candidatos, uno de los cuellos de botella del algoritmo Apriori. En su artículo proponen otro algoritmo para construir la estructura FP-Tree en la que recoger toda la información relevante de $\mathcal{D}$. FP-Tree realmente es una representación compacta de $\mathcal{D}$ que se pue-

de manipular en la memoria principal del ordenador. Es por ello que si el número de ítems distintos que contiene es muy grande, las prestaciones de FP-Growth se reducen notablemente teniendo que trabajar con un soporte alto para poder llevar a cabo su ejecución.

Hipp et al. (2000) presentan una comparativa entre los cuatro algoritmos más representativos de ARM: Apriori, FP-Growth, Partition y Eclat, concluyendo que no existen grandes diferencias en cuestión de tiempo de ejecución y que la propia distribución de $\mathcal{D}$ puede influir en el hecho de que un algoritmo funcione mejor en un dataset concreto respecto a los demás algoritmos.

Özel y Güvenir (2001) proponen el algoritmo PHP, basado en DHP pero utilizando hashing perfecto, con lo que no es necesario hacer doble recuento de los candidatos a $(k+1)$-itemsets frecuentes. No especifican la función hash que usan ni aseguran que su método pueda ser utilizado con cualquier colección $\mathcal{D}$, lo que no es descartable pues el número de candidatos a $(k+1)$-itemsets puede ser tan grande que no sea viable su almacenamiento en un vector de códigos hash.

D.-I. Lin y Kedem (2002) hacen una propuesta interesante desde el punto de vista de la minería de itemsets frecuentes, completar la búsqueda bottom-up de itemsets frecuentes mediante una búsqueda top-down simultánea que puede reducir drásticamente el número de candidatos generado por el algoritmo y el número de lecturas de $\mathcal{D}$. El único inconveniente de este algoritmo es que no se guarda el soporte real de todos los itemsets frecuentes cuyos candidatos no han sido generados, con lo que sería necesaria una nueva lectura de $\mathcal{D}$ para obtenerlo. En su artículo indican que los algoritmos de ARM utilizados son eficientes cuando los itemsets frecuentes de $\mathcal{D}$ son cortos, decreciendo su eficiencia conforme aumenta su longitud. Presentan un nuevo algoritmo que combina las búsquedas bottom-up y top-down, Pincer-Search, para mantener y actualizar una nueva estructura de datos, MFCS (*Maximum Frequent Candidate Set*) con la que obtener el conjunto de itemsets frecuentes maximales, MFS, sin necesidad de usar un conjunto grande de candidatos, lo que indican que ahorrará mucho tiempo de CPU al proceso por ahorrar búsquedas y conteo de candidatos que no llegarán a ser itemsets frecuentes.

En su artículo presentan un esquema general muy claro sobre el proceso de minería de itemsets frecuentes:

> *Throughout the execution, the set of all itemsets is partitioned, perhaps implicitly, into three sets:*
>
> 1. frequent*: This is the set of those itemsets that have so far been discovered as frequent.*
> 2. unfrequent*: This is the set of those itemsets that so far have been discovered as infrequent.*
> 3. unclassified*: This is the set of all the other itemsets.*
>
> *Initially, the frequent and the infrequent sets are empty. Throughout the execution, they grow at the expense of the unclassified set. The execution terminates when the unclassified set becomes empty, and then, of course, all the maximal frequent itemsets are discovered.*

Holt y Chung (2002) presentan el algoritmo IHP mostrando con sus experimentos que se comportan mejor que Apriori y DHP en colecciones con transacciones largas. En la primera lectura de $\mathcal{D}$, IHP realiza el recuento de cada ítem de $\mathcal{I}$, además de crear una `TID Hash Table` (THT) para cada ítem, utilizando una función hash, lo que supone un aumento de requerimiento de memoria principal pero limitado por el valor máximo obtenido en la función hash, y no por el número total de transacciones como ocurría en Partition. Su principal aportación consiste en la eliminación de candidatos mediante una comprobación previa de las THT de cada ítem presente en el k-candidato a comprobar, esta reducción supone un recuento más rápido de los candidatos dado su menor número. También incorporan técnicas de reducción de ítems en transacciones y suprimen las transacciones que no pueden generar más k-itemsets frecuentes, técnicas ya revisadas en la literatura expuesta en esta sección.

Mansuri y Berengoltz (2002) definen las *reglas de dependencia* basadas en el test χ^2, y proponen un algoritmo para descubrirlas. En su tesis, Goethals (2002) propone heurísticas que ayudan a limitar las fronteras de los conjuntos de itemsets a descubrir. Su trabajo se puede aplicar a cualquier algoritmo derivado de Apriori.

Zhao y Bhowmick (2003) presentan un completo informe sobre minería de reglas de asociación. Clasifican los algoritmos revisados en función de las estructuras que utilizan para gestionar los itemsets contenidos en el dataset $\mathcal{D}$. Goethals (2003) presenta otro completo informe sobre la teoría y algoritmos de ARM más valorados hasta la fecha.

Hong et al. (2004) proponen un algoritmo que busca reglas de asociación difusas en transacciones con datos cuantitativos, reduciendo tiempos de ejecución en esta fase. Pei et al. (2004) analizan las restricciones aplicables a los algoritmos de ARM. Debido a la gran cantidad de itemsets frecuentes y reglas de asociación que pueden contener el dataset $\mathcal{D}$, se plantean la necesidad de permitir al analista fijar su foco en parte del conocimiento a extraer de $\mathcal{D}$, mediante restricciones que permitan añadir al análisis conocimientos previos sobre el contexto en estudio. Crean una clase de restricciones aplicables al algoritmo FP-Growth, argumentando que Apriori no permite aplicarlas. Coenen, Leng y Ahmed (2004) presentan nuevos algoritmos basados en el uso de las estructuras T-Tree y P-Tree.

Tsay y Chang-Chien (2004) proponen el algoritmo CDAR, que comienza dividiendo $\mathcal{D}$ en clusters de modo que los k-itemsets frecuentes se busquen en el k-ésimo cluster obtenido en el primer paso. Aunque indican que sólo se necesita una lectura de $\mathcal{D}$ para ejecutar su algoritmo, al revisar su algoritmo se comptueba que para llevarlo a término son necesarias varias lecturas del dataset, además de suficiente espacio en memoria para almacenarlo en forma de clusters. El conjunto de itemsets frecuentes que proporciona no es el mismo que proporciona Apriori.

Jamali et al. (2005) proponen un método de codificado para reducir notablemente el tamaño de las bases de datos de transacciones, presentando un algoritmo que utiliza la base de datos codificada. Burdick et al. (2005) presentan el algoritmo MAFIA, incidiendo en la búsqueda de itemsets frecuentes maximales. Gouda y M. Zaki (2005) presentan el algoritmo GenMax, un método de búsqueda que permite encontrar los itemsets frecuentes maximales. En su tesis, G. Liu (2005) presenta una detallada descripción de su investigación sobre minería de itemsets frecuentes, proponiendo el algoritmo AFOPT. Utiliza una estrategia de adaptación que hace que su algoritmo sea eficiente tanto en bases de datos densas como dispersas.

Tsay y Chiang (2005) presentan el algoritmo CBAR. Dividen el dataset $\mathcal{D}$ en clusters en función de la longitud de las transacciones y buscan los k-itemsets frecuentes en el cluster $\mathcal{D}_k$, que sólo contiene transacciones de longitud k, de modo que si se comprueba que un candidato es frecuente no se sigue contando su presencia en las transacciones de mayor tamaño, y si su presencia no alcanza el soporte mínimo fijado por el analista se conserva como candidato hasta que se haya comprobado en el resto de

clusters si tiene suficiente soporte. Esto supone una ganancia de tiempo de ejecución pero impide conocer el soporte real de muchos itemsets, lo que conducirá a una nueva lectura de $\mathcal{D}$ para concluir con la generación de reglas de asociación. Cabe destacar que en bases de datos como `chess.dat` cuyas transacciones tienen todas la misma longitud, no se obtiene ningún beneficio.

Rozenberg y Gudes (2006) proponen un algoritmo que permite obtener las reglas de asociación presentes en bases de datos distribuidas, protegiendo la privacidad de cada una de las DBs.

Shichao Zhang et al. (2007) presentan el algoritmo Efficient Dynamic Database Updating Algorithm (EDUA) para minar bases de datos dinámicas. Dong y M. Han (2007) presentan el algoritmo BitTableFI, que utiliza una nueva estructura, BitTable. L. Liu et al. (2007) utilizan paralelismo para optimizar el proceso.

Estructuras

La implementación de cualquier algoritmo conlleva el modelado de estructuras de datos que puedan ser utilizadas de forma eficiente en cuanto a consumo de recursos informáticos, memoria RAM y tiempo de acceso principalmente.

La mayoría de algoritmos de ARM utilizan el árbol $\mathcal{L}$ (Agrawal, Imielinski et al., 1993b) o la estructura FP-Tree (J. Han, Pei et al., 2000). El resto de algoritmos utilizan estructuras derivadas de una de ellas, modificando algún detalle que pueda mejorar su eficiencia.

D.-I. Lin y Kedem (2002) proponen almacenar los itemsets en la estructura MFCS (Maximum Frequent Candidate Set), capaz de gestionarlos de un modo más eficiente, permitiendo hacer su recuento en dos sentidos, comenzando con 1-itemset e incrementando su tamaño –del mismo modo que hace Apriori– y almacenando simultáneamente los k-itemsets frecuentes que va encontrando en $\mathcal{D}$ para detectar los frecuentes en sentido inverso, reduciendo su tamaño.

Cheung y Zaiane (2003) proponen una nueva estructura, CATS-Tree, que extiende la idea de la estructura FP-Tree para mejorar la compresión de $\mathcal{D}$ y permitir el minado de itemsets frecuentes sin generación de candidatos. A destacar la escalabilidad de esta estructura que permite modificar el soporte mínimo sin tener que reconstruirla, lo que hace posible también

la inserción o eliminación de transacciones sin repetir todo el proceso de ARM.

G. Liu (2005) propone el uso de la estructura AFOPT, y otras dos estructuras para almacenar bases de datos condicionales. Utiliza una estrategia de adaptación que permite elegir la mejor estructura en función de la densidad de las bases de datos condicionales.

Ahmed et al. (2006) presentan un método de particionado de $\mathcal{D}$ que resuelve parte del problema de trabajar con la memoria principal cuando el dataset de transacciones es muy grande y denso. En su propuesta, se organizan los datos de $\mathcal{D}$ en unas estructuras que pueden ser tratadas independientemente y, por tanto, pueden ser alojadas en memoria evitando múltiples lecturas de disco. Calders y Goethals (2006) proponen una representación comprimida de los itemsets frecuentes, los itemsets frecuentes no-derivables, que pueden resolver ciertos problemas de uso de memoria en la aplicación de los algoritmos de FIM. Plantean el problema que surge con los algoritmos de ARM al imponer soportes mínimos demasiado bajos o trabajar con datos muy correlacionados, debido al enorme número de itemsets frecuentes que puede producir el análisis. Presentan las reglas deductivas, con las que poder eliminar las redundancias presentes en las fronteras de los conjuntos de itemsets frecuentes y posibilitar el análisis sin tener que recurrir a grandes equipos.

Dong y M. Han (2007) utilizan la estructura BitTable, que comprime $\mathcal{D}$ horizontal y verticalmente para la generación y recuento eficientes de los candidatos a itemset frecuente. G. Liu et al. (2007) proponen el uso de CFP-tree, una estructura compacta para el almacenamiento de itemsets y optimizada para realizar búsquedas.

Tsay, T.-J. Hsu et al. (2009) presentan otra estructura, FIU-tree, con la que sólo se ha de acceder dos veces a $\mathcal{D}$ para encontrar todos los itemsets frecuentes que contiene.

2.1.4. El ítem raro

Desde la irrupción de la informática en pequeñas y grandes empresas, se guarda en bases de datos todo tipo de información, entre otra las transacciones que modelan la cesta de la compra o las sesiones de navegación de un usuario a través de un portal web. El volumen de datos almacenados crece a diario por lo que su análisis inicial se realiza con técnicas como la

minería de reglas de asociación, capaces de extraer información sobre la coexistencia de ítems en grandes colecciones de datos.

El uso de soporte mínimo en ARM permite abordar el problema sobre grandes bases de datos con la tecnología actual, pero impide obtener información sobre una gran cantidad de los ítems en estudio, ítems que tiene menor soporte que el fijado como mínimo para ser estudiados, por lo que se denominan ítems raros. En la mayoría de algoritmos de ARM, todos los ítems raros desaparecen por completo del análisis una vez detectado que lo son.

El dilema del ítem raro (B. Liu, W. Hsu y Ma, 1999) ha sido estudiado por muchos investigadores para obtener reglas de asociación que involucren a ítems poco frecuentes. Sin embargo todas las propuestas realizadas sobrecargan las tareas del algoritmo usado y de sus analistas, y no aportan reglas de uso frecuente.

Los ítems raros son aquellos que por alguna razón no aparecen con frecuencia en las transacciones del dataset analizado. En la cesta de la compra, los ítems raros suelen ser muy caros o duraderos, también pueden ser novedades que tardarán en ser comprados de forma frecuente. En un portal web los ítems raros son páginas muy especializadas, o de baja calidad, o páginas de novedades que aún no han recibido suficientes visitas. En el problema de clasificación, un ítem raro puede ser el valor de un atributo poco frecuente en la población en estudio, p.ej. el valor *pelirrojo* en el atributo *color-de-pelo* de un estudio sobre los habitantes de cualquier ciudad española.

Los ítems exclusivos (por su alto precio, larga durabilidad o especialización) serán siempre ítems raros aunque se incorporen a un sistema de recomendación (en inglés, Recommender System, RS) ya que son infrecuentes por naturaleza. Sin embargo, los ítems nuevos deberían ser incorporados artificialmente a los RSs hasta conseguir que alcancen la categoría de ítems frecuentes, momento en el cual ya pueden ser tratados mediante la búsqueda clásica de reglas de asociación.

Los primeros trabajos en plantear soluciones al dilema del ítem raro proponen agrupar varios ítems infrecuentes en uno solo, de modo que se incremente su frecuencia (J. Han y Fu, 1995), o bien separar la base de datos en grupos de ítems con frecuencias homogéneas y estudiar cada grupo independientemente (Lee et al., 1998). En el primer caso no se obtiene información sobre los ítems particulares que forman el ítem compuesto

estudiado, y en el segundo no se descubren reglas que relacionen ítems dispuestos en diferentes grupos.

J. Han y Fu (1995) proponen dividir el dataset $\mathcal{D}$ utilizando una jerarquía, y buscar las reglas de asociación tanto entre ítems como entre las categorías en las que sitúan a cada uno de ellos. Con esta estrategia se puede reducir el número de ítems a vincular, cuando se trabaja con ítems raros se puede ver su relación con otros ítems mediante la categoría a la que pertenecen, por lo que todos los ítems pueden estar representados en alguna regla de asociación y se puede redirigir el análisis para estudiar mejor la relación de dichos ítems raros con el resto de ítems de la población. El procedimiento es correcto pero conlleva análisis específicos y la creación y mantenimiento de la jerarquía propuesta, lo que puede provocar una ralentización del análisis y falta de actualidad si aparecen nuevos ítems en la población y no se realiza en paralelo una correcta jerarquización de los mismos.

B. Liu, W. Hsu y Ma (1999) y Palshikar et al. (2007) ponen el énfasis en la distribución de los ítems presentes en $\mathcal{D}$. La investigación precedente sobre minería de reglas de asociación considera una distribución uniforme de los ítems de $\mathcal{D}$, lo que no es correcto en la mayoría de las ocasiones y produce una gran cantidad de reglas de asociación que no aportan información de calidad sobre las relaciones reales existentes entre los ítems de $\mathcal{D}$. Esa información de baja calidad presenta otro problema: no deja suficientes recursos para analizar otra información de mayor calidad pero menor presencia en la base de datos. Para resolver esto proponen el uso de soportes mínimos múltiples en lugar del clásico soporte mínimo único, de modo que los ítems con mayor soporte en $\mathcal{D}$ necesiten más evidencias para mostrar relaciones entre sí que los ítems con poca presencia, a los que los algoritmos clásicos simplemente ignoran considerándolos ítems raros.

B. Liu, W. Hsu y Ma (1999) proponen el algoritmo MSApriori, una modificación de Apriori que usa múltiples soportes. Cada ítem tiene su propio soporte mínimo, obtenido de $\mathcal{D}$. Primero comprueban el soporte del ítem en $\mathcal{D}$, si no supera el soporte mínimo general se le asigna un soporte mínimo particular calculado como una fracción del soporte real del ítem. Una vez determinados los soportes mínimos de cada ítem ordenan todos los ítems en base a su soporte particular y después proceden con el algoritmo haciendo comparaciones con cada uno de los soportes involucrados en cada k-itemset. Así consiguen que se considere la relación entre dos

ítems frecuentes sólo cuando esa relación es muy frecuente en $\mathcal{D}$. De este modo, se alivia considerablemente la carga de memoria requerida por el algoritmo y se puede abordar el estudio sobre un número mayor de ítems. La asignación de soporte a cada ítem también puede hacerse por parte del analista, lo que conlleva algo de trabajo supervisado que puede no estar actualizado cuando aparecen nuevos ítems en la población en estudio.

Yun et al. (2003) proponen el uso de un soporte relativo para cada itemset basado en la confianza de las reglas de asociación que generan sus ítems, lo que permite guardar información de los itemsets con menor soporte que el soporte mínimo, siempre que superen su soporte relativo, garantizando que las reglas de asociación que generan sean de calidad. Weiss (2004) analiza a fondo las características de los ítems raros y presenta un marco unificado para tratarlos con diferentes técnicas de minería de datos. Contiene un excelente estado del arte sobre el tema tratado. Kouris et al. (2003; 2005) proponen el uso de múltiples soportes mínimos mediante un algoritmo obtenido a partir de las ideas presentadas con los algoritmos DIC y MSApriori. Hu e Y.-L. Chen (2006) proponen el algoritmo CFP-growth basado en la estructura `FP-tree`. Utilizan múltiples soportes mínimos para descubrir reglas de asociación que involucren a los ítems raros del dataset. Tseng y W.-Y. Lin (2007) plantean el minado de reglas de asociación generalizadas bajo la influencia de una taxonomía, utilizando múltiples soportes. Kiran y Reddy (2009) proponen una modificación interesante del algoritmo MSApriori. Incorporan al análisis medidas de tendencia de los ítems de $\mathcal{D}$, con las que se obtienen múltiples soportes mínimos adaptados a las características del dataset.

En la mayoría de propuestas, el soporte mínimo se obtiene en función del soporte real del ítem, con lo que muchos ítems raros se incorporan al sistema con facilidad. Sin embargo siguen quedando muchos ítems sobre los que no se obtiene información, pues si se incorporan al estudio se detiene la ejecución del algoritmo por falta de recursos. Además, las reglas que proporcionan siempre sugieren una relación entre un ítem raro como antecedente y uno que no lo es como consecuente, lo que en un sistema de recomendación no es práctico porque se usaría únicamente cuando el usuario solicite un ítem raro, lo que ocurre en muy pocas ocasiones dada la naturaleza del ítem solicitado.

En la sección 3.1.4 se exponen las reglas de oportunidad, que permiten a los algoritmos de búsqueda de reglas de asociación encontrar simultá-

neamente información de interés sobre los ítems que no superen el soporte mínimo, con un consumo mínimo de recursos, sin intervención de los analistas y aportando conocimiento útil. Con esta propuesta se alivia parte del problema generado por el dilema del ítem raro.

Lo único que sabe el investigador sobre un ítem raro es que se trata de un ítem con poca presencia en un dataset $\mathcal{D}$, sabe que el soporte del ítem es inferior a un valor mínimo prefijado. En los algoritmos de ARM, se ignoran los ítems raros para poder llevar a cabo el análisis, no existe otro motivo. La única justificación existente para no incorporar los ítems raros al análisis es que *se supone que representan a pocos individuos de la población en estudio*, suposición que sería correcta si $\mathcal{D}$ fuese una muestra representativa de la población en estudio.

La minería de datos tiene su origen en la *Estadística*, pero la formalización, implementación y pruebas de los algoritmos de DM se llevan a cabo desde la *Informática*, lo que puede provocar que no se consideren conceptos fundamentales del problema que se pretende resolver. El uso de soporte en ARM se justifica originalmente en que la frecuencia muestral es un buen estimador de la frecuencia poblacional cuando se trabaja con muestras representativas, y lo es en la descripción hecha en el artículo de Agrawal, Imielinski et al. (1993a). Sin embargo, no cualquier dataset es una muestra representativa de una población. En el capítulo 4 se hace una reflexión sobre la representatividad de los datasets utilizados en cientos de artículos científicos para poner a prueba algoritmos de minería de reglas de asociación.

2.2. Minería de Reglas de Clasificación

En esta sección se hace una introducción al problema de clasificación y a la aportación que se hace desde la perspectiva de la minería de datos a la resolución de este problema, la minería de reglas de clasificación (en inglés, Classification Rules Mining, CRM). A continuación se analizan las relaciones existentes entre CRM y ARM, cuya fusión da lugar a nuevas metodologías de DM en torno a la minería de reglas de clasificación asociativa (en inglés, Classification Association Rules Mining, CARM).

Sea $\mathcal{P}$ una base de datos que contiene las características o propiedades de los individuos de una población, información sobre $\mathcal{N}$ atributos

medidos a dichos individuos. $\mathcal{P}$ se puede dividir en $\mathcal{Q}$ grupos disjuntos, denominados *clases*.

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \ldots \cup \mathcal{P}_\mathcal{Q} \; / \; \mathcal{P}_i \cap \mathcal{P}_j = \emptyset \; \forall i \neq j \tag{2.14}$$

El problema de clasificación consiste en descubrir *reglas de clasificación* que ayuden a determinar a qué clase pertenece cada individuo de $\mathcal{P}$ (Piatetsky-Shapiro y Frawley, 1991). Para llevar a cabo un experimento de clasificación:

1. Se parte de un conjunto de individuos de $\mathcal{P}$ ya clasificados, denominado *dataset de entrenamiento*.
2. Se analizan las similitudes entre las características de los individuos de una misma *clase*, y las diferencias con las características de los individuos de las otras clases, obteniendo las *reglas de clasificación*.
3. Se ordenan las *reglas de clasificación* en función de su *utilidad*, definida antes de iniciar el experimento.
4. Se utilizan las *reglas de clasificación* obtenidas para
   - comprobar su precisión al utilizarlas sobre un *dataset de pruebas*, que contiene individuos ya clasificados, y
   - clasificar a los nuevos individuos que se incorporen a $\mathcal{P}$.

Esta tesis no aborda el problema de clasificación completo, propone metodologías para llevar a cabo la fase de descubrimiento de reglas de clasificación. Los datos que se analizarán son los registros clasificados que contiene el dataset de entrenamiento, dataset de clasificación–o simplemente dataset– en adelante. A continuación se presentan los conceptos necesarios para desarrollar esta sección, adaptando la notación para poder enlazar los problemas de clasificación con las técnicas de ARM, que básicamente consisten en buscar relaciones entre variables categóricas.

Sea $\mathcal{A}_i$ el atributo i-ésimo del problema de clasificación. El rango teórico de $\mathcal{A}_i$ puede ser infinito cuando representa a una variable aleatoria numérica. En un análisis basado en minería de datos, sin embargo, el conjunto de valores distintos que toma una variable es siempre finito, de cardinalidad igual o inferior al número de registros que contiene el dataset de clasificación a analizar. Por lo tanto, se define el rango de un atributo en función de los datos observados y no de conceptos teóricos.

Definición 2.18 (Rango de un atributo)**.** *El rango del atributo $\mathcal{A}_i$ es el conjunto de valores que toma el atributo en el dataset de clasificación.*

$$rg(\mathcal{A}_i) = \{v_i^1, v_i^2, \ldots\} \tag{2.15}$$

Para no complicar la notación usada, se denota el cardinal del rango del atributo $\mathcal{A}_i$ utilizando la función módulo. El número de valores distintos que toma el atributo $\mathcal{A}_i$ en el dataset de clasificación es $|rg(\mathcal{A}_i)|$.

Definición 2.19 (Evento primario)**.** *Al observar en un individuo el valor v_i^p, se obtiene el* evento primario $\{\mathcal{A}_i = v_i^p\}$.

Por simplicidad, se usa v_i^p para denotar el evento primario $\{\mathcal{A}_i = v_i^p\}$.

Propiedad 2.2 (Unicidad)**.** *Dos eventos primarios, v_i^p y v_i^q, del mismo atributo $\mathcal{A}_i$, son mutuamente excluyentes si $p \neq q$.*

Definición 2.20 (Evento compuesto)**.** *Un* evento compuesto *es un vector ordenado de eventos primarios. Con la expresión* evento k-compuesto *se indica que el evento contiene k eventos primarios.*

Esta definición permite referirse a un evento primario como un evento 1-compuesto. Se utiliza en cada caso el término más adecuado al contexto.

Definición 2.21 (Base de datos $\mathcal{P}$)**.** *La base de datos que contiene las características de los individuos de una población, $\mathcal{P}$, es un conjunto de eventos $\mathcal{N}$-compuestos. $(v_1^{p_1}, \ldots, v_{\mathcal{N}}^{p_{\mathcal{N}}})$ es un registro de $\mathcal{P}$.*

Para no trabajar con $\mathcal{Q}$ datasets, como sugiere la ecuación 2.14, se añade a cada registro de $\mathcal{P}$ una etiqueta con la clase de pertenencia del individuo al que representa el registro. Generalmente se añade la etiqueta como primera columna del dataset de clasificación, aunque puede hacerse como última columna, lo único que ha de saber el investigador es cuál es su posición en el dataset de clasificación.

Definición 2.22 (Registro clasificado)**.** *Si se añade a un registro de $\mathcal{P}$ una etiqueta con la clase c_i a la que pertenece el registro, se obtiene un* registro clasificado, *representado por el vector ordenado*

$$(c_i, v_1^{p_1}, \ldots, v_{\mathcal{N}}^{p_{\mathcal{N}}}) \tag{2.16}$$

Esta definición permite volver a utilizar el concepto de dataset $\mathcal{D}$, esta vez adaptado a un problema de clasificación y no a uno de minería de reglas de asociación.

Definición 2.23 (Dataset de clasificación)**.** *Un* dataset de clasificación *es un conjunto de registros clasificados.*

El objetivo de la minería de reglas de clasificación es encontrar las mejores reglas de clasificación (en inglés, Classification Rule, CR) del dataset de clasificación.

Definición 2.24 (Regla de Clasificación)**.** *Una* regla de clasificación *es una expresión del tipo* $X \rightarrow c_i$ *, donde* X *es un evento compuesto y* $c_i,\ i = 1 \ldots \mathcal{Q}$*, es la clase a la que pertenece el evento* X*.*

Agrawal, Imielinski et al. (1993a) describen el problema de clasificación como una búsqueda de reglas que permitan dividir un dataset de clasificación en grupos disjuntos. En su artículo presentan un marco unificado para resolver problemas de clasificación, asociación y secuencias, por lo que tienen que utilizar una notación propia para representar a tres tipos diferentes de datos. Utilizan el término *tupla*, que puede interpretarse como vector ordenado –para CRM– o como conjunto de ítems en el que no importa el orden de los mismos y no hay duplicados –para ARM–.

Definen los registros de un dataset de clasificación como tuplas etiquetadas, donde la etiqueta identifica el grupo al que pertenece cada registro, y la tupla especifica las propiedades del objeto o individuo que se quiere clasificar. Esta definición está pensada para utilizar algoritmos de minería de reglas de asociación (ARM) en problemas de clasificación, pero si no se aplica una transformación a los datos originales se perdería la información proporcionada por el uso de vectores en clasificación.

Para evitar esta pérdida de información se han de codificar todos los valores del dataset de clasificación de modo que no haya ningún problema en recuperar la información perdida al pasar de trabajar con vectores a trabajar con conjuntos. La codificación se inicia asignando a las clases los códigos $1 \ldots \mathcal{Q}$, a continuación se codifican los valores del atributo $\mathcal{A}_1$ con los códigos $\mathcal{Q}+1 \ldots \mathcal{Q}+|rg(\mathcal{A}_1)|$, los de $\mathcal{A}_2$ con los códigos $\mathcal{Q}+|rg(\mathcal{A}_1)|+1 \ldots \mathcal{Q}+|rg(\mathcal{A}_1)|+|rg(\mathcal{A}_2)|$, hasta terminar con la codificación del atributo $\mathcal{A}_{\mathcal{N}}$.

Definición 2.25 (Registro clasificado (ARM))**.** *El registro clasificado de la ecuación 2.16*

$$(c_i, (v_1^{p_1}, \ldots, v_{\mathcal{N}}^{p_{\mathcal{N}}}))$$

se representa con el conjunto de códigos

$$\{i, v_1, \ldots, v_{\mathcal{N}}\} \tag{2.17}$$

Cada elemento de un registro clasificado representa a una clase o un atributo, y existe una función que permite asociar los elementos del conjunto con el valor de la clase o atributo correspondiente.

El siguiente registro clasificado se ha extraído del dataset `mushroom` (UCI):

$$p, x, s, n, t, p, f, c, n, k, e, e, s, s, w, w, p, w, o, p, k, s, u$$

donde p indica que se trata de un ejemplar de seta venenoso (*poisonous*), y $(x, s \ldots u)$ son los valores de los 22 atributos que se han medido en dicho ejemplar. El primer atributo, por ejemplo, es la forma del sombrero (`cap-shape`), y x significa que es convexo.

Si se codifican las clases y los valores de los atributos usando números enteros consecutivos, se obtiene un registro clasificado con el aspecto de una transacción.

$$1, 3, 9, 13, 23, 25, 34, 36, 38, 40, 52, 54, 59, 63, 67, 76, 85, 86, 90, 93, 98, 107, 113$$

La diferencia entre ambas representaciones radica en que la segunda no contiene ítems duplicados. Los algoritmos de ARM utilizan transacciones en las que no importa el orden de los ítems que contienen, ignorando cualquier duplicado. En la primera representación puede haber valores duplicados porque son registros clasificados representados por vectores. El primer valor está siempre asociado a la clase en estudio, y el orden del resto de valores indica a qué atributo pertenece el valor. En este caso puede haber valores duplicados, pero su significado es distinto porque cada uno se refiere a un atributo distinto.

En este ejemplo, la función que devuelve la clase de pertenencia del registro toma sólo 2 valores en el dataset analizado:

$$f(1) = \{\texttt{tipo} - \texttt{seta} = venenosa\}$$
$$f(2) = \{\texttt{tipo} - \texttt{seta} = comestible\}$$

La función que transforma el resto de códigos en información sobre el experimento de clasificación tiene 117 valores:

$$
\begin{aligned}
&f(3) = \{\texttt{forma} - \texttt{sombrero} = convexa\}\\
&f(4) = \{\texttt{forma} - \texttt{sombrero} = campana\}\\
&\dots\\
&f(8) = \{\texttt{forma} - \texttt{sombrero} = hundido\}\\
&f(9) = \{\texttt{cap} - \texttt{surface} = fibrosa\}\\
&\dots\\
&f(113) = \{\texttt{habitat} = urbano\}\\
&\dots\\
&f(119) = \{\texttt{habitat} = bosque\}
\end{aligned}
$$

El objetivo de la clasificación es descubrir reglas para caracterizar cada uno de los grupos, es decir, descubrir todas las reglas cuyo consecuente sea uno de los grupos en que se ha dividido la base de datos. En términos de ARM, se trata de reglas de asociación con una restricción sintáctica. Esto permite el análisis de datasets de clasificación utilizando algoritmos de ARM, obteniendo *reglas de clasificación* en lugar de *reglas de asociación*.

Las reglas de clasificación definidas por Agrawal, Imielinski et al. (1993a) tienen la forma

$$(3, \dots) \rightarrow 1$$

que se interpreta decodificando los valores de la regla:

Si {`forma-sombrero` = convexa $\wedge \dots$} *entonces* (`tipo-seta` = venenosa)

y se completa esta información con el soporte y confianza de la regla de clasificación descrita.

Es exactamente la misma información que proporcionan las reglas de clasificación utilizadas por J. R. Quinlan (1993), que propone uno de los algoritmos más utilizado en clasificación, C4.5. El problema es que los conceptos utilizados son totalmente diferentes, Agrawal, Imielinski et al. (1993a) realmente han adaptado las técnicas de minería de reglas de asociación al problema de clasificación, mientras que J. R. Quinlan (1993) trabaja directamente en esta disciplina utilizando árboles de decisión. En la sección 2.2.1 se analizan las reglas de clasificación asociativa, que pueden resolver esta ambigüedad.

Uso de soporte mínimo en datasets de clasificación

Al aplicar técnicas de minería de reglas de asociación sobre datasets de clasificación se han de considerar aspectos específicos de los algoritmos de ARM que pueden afectar a los resultados que se buscan en clasificación.

El uso de soporte mínimo en los algoritmos de ARM permite llevar a cabo el análisis completo de los ítems más frecuentes del dataset, a cambio de ignorar los ítems con poco soporte en la muestra analizada. Cuando no se utiliza, es muy probable que se aborte la ejecución del algoritmo por sobrecarga de memoria, como ocurre con el dataset de clasificación `mushroom` utilizado para ejemplarizar esta sección.

Son muchos los artículos que utilizan el dataset `mushroom` para experimentar con sus propuestas. Cuando aplican un umbral de soporte mínimo están renunciando a parte de la información que contiene el dataset. Borgelt (2004) llega a reducir el soporte mínimo a un 1.23 %. Aunque puede parecer un umbral bajo y es aceptado por la mayoría de investigadores de CARM sin cuestionar sus consecuencias, deja fuera del análisis cualquier dato que aparezca en menos de 100 ocasiones en el dataset de clasificación. Dong y M. Han (2007) aplican al mismo dataset de clasificación un soporte mínimo entre el 4 % y el 20 %. Luo y X.-M. Zhang (2007) usan valores entre el 5 % y el 30 % para este umbral. La mayoría de artículos revisados utilizan un 1 % o un 5 % de soporte mínimo sin ninguna justificación, sólo por el hecho de que son dos valores muy utilizados en la investigación publicada al respecto.

La información que se pierde al no analizar los ítems raros de un dataset de clasificación puede ser o no relevante, el hecho es que no disponemos de ella una vez terminado el análisis. En la sección 3.2.1 se presenta una compresión sin pérdidas de los datasets de clasificación con la que se puede analizar `mushroom` sin renunciar a ninguno de los datos que contiene, es decir, utilizando un soporte mínimo del 0 %.

2.2.1. Minería de Reglas de Clasificación Asociativa

La minería de reglas de clasificación (CRM) y la minería de reglas de asociación (ARM) son dos áreas de la *Informática* ampliamente investigadas y utilizadas. Ambas se basan en la extracción del conocimiento contenido en datasets. CRM extrae de un dataset de clasificación un conjunto pequeño de reglas de clasificación para crear un clasificador (J. R. Quinlan,

1993), mientras ARM extrae de un dataset todas las reglas de asociación que superan un umbral mínimo de soporte y confianza (Agrawal, Imielinski et al., 1993b).

Ambas técnicas proporcionan reglas del tipo "*si X, entonces Y*" o "*si X e Y, entonces Z y W*", donde la expresión a la izquierda de "*entonces*" se denomina antecedente y la expresión de su derecha consecuente.

ARM usa algoritmos no supervisados de DM, extrae las reglas sin información previa sobre el consecuente de la regla, mientras CRM usa algoritmos supervisados de ML en que la extracción de reglas se hace en base a los consecuentes (Rai et al., 2012).

Para evitar conflictos al estudiar reglas de clasificación, y debido a que la definición usada por J. R. Quinlan es anterior, B. Liu, W. Hsu y Ma (1998) definen las reglas de clasificación asociativa (en inglés, Classification Association Rule, CAR) en base a las reglas de clasificación definidas por Agrawal, Imielinski et al. (1993a).

> *Classification rule mining aims to discover a small set of rules in the database that forms an accurate classifier. Association rule mining finds all the rules existing in the database that satisfy some minimum support and minimum confidence constraints. For association rule mining, the target of discovery is not pre-determined, while for classification rule mining there is one and only one predetermined target. techniques. The integration is done by focusing on mining a special subset of association rules, called class association rules (CARs).*

Definición 2.26 (Regla de clasificacion asociativa)**.** *En un problema de clasificación, una regla de clasificación asociativa es una regla de asociación cuyo consecuente es la clase en estudio.*

Una vez definidas las CARs, B. Liu, W. Hsu y Ma (1998) se basan en el algoritmo Apriori (Agrawal y Srikant, 1994) y en su conocimiento sobre los datasets de clasificación para proponer el algoritmo CBA (Classification Based on Associations), primer algoritmo de minería de reglas de clasificación asociativa (CARM).

> *The integration is done by focusing on a special subset of association rules whose right-hand-side are restricted to the classification class*

attribute. We refer to this subset of rules as the class association rules (CARs). An existing association rule mining algorithm (Agrawal and Srikant 1994) is adapted to mine all the CARs that satisfy the minimum support and minimum confidence constraints. This adaptation is necessary for two main reasons:

1. *Unlike a transactional database normally used in association rule mining (Agrawal and Srikant 1994) that does not have many associations, classification data tends to contain a huge number of associations. Adaptation of the existing association rule mining algorithm to mine only the CARs is needed so as to reduce the number of rules generated, thus avoiding combinatorial explosion (see the evaluation section).*
2. *Classification datasets often contain many continuous (or numeric) attributes. Mining of association rules with continuous attributes is still a major research issue (Srikant and Agrawal 1996; Yoda et al 1997; Wang, Tay and Liu 1998). Our adaptation involves discretizing continuous attributes based on the classification predetermined class target. There are many good discretization algorithms for this purpose (Fayyad and Irani 1993; Dougherty, Kohavi and Sahami 1995).*

Data mining in the proposed associative classification framework thus consists of three steps:

- *discretizing continuous attributes, if any*
- *generating all the class association rules (CARs), and*
- *building a classifier based on the generated CARs.*

Su propuesta se divide en tres procesos: 1) discretización, 2) generación de CARs y 3) construcción de un clasificador. Cada uno de ellos tiene la suficiente entidad como para ser estudiados de forma independiente.

Desde su presentación, la minería de reglas de clasificación asociativa ha recibido mucha atención por científicos que comprueban que, mejorando ciertos aspectos del algoritmo CBA, se consigue de un modo más eficiente clasificadores más precisos que los proporcionados por la minería de reglas de clasificación, C4.5 entre otros.

K. Wang et al. (2000) proponen el algoritmo ADT (Association based Decision Tree), que construye el árbol de decisión de la minería de reglas de clasificación usando reglas de asociación.

W. Li et al. (2001) presentan CMAR (Classification based on Multiple Association Rules) un nuevo algoritmo de CARM, basado en FP-Growth, que utiliza múltiples reglas de clasificación asociativa para construir el clasificador.

Meretakis (2002) presenta una visión unificada de los problemas de asociación y clasificación. Propone un algoritmo de clasificación basado en reglas de clasificación asociativa y compara los resultados obtenidos al evaluar 21 datasets de clasificación de UCI con su algoritmo y con otros ya existentes.

Yin y J. Han, 2003 comprueban que los algoritmos de CARM generan un número excesivo de reglas. Combinan las ventajas de los algoritmos de CARM con las de los algoritmos de CRM y proponen el algoritmo CPAR (Classiffication based on Predictive Association Rules), que genera las reglas directamente desde el dataset de clasificación para evitar el problema de desbordamiento de memoria de los algoritmos de ARM.

Coenen y Leng (2004; 2007) presentan una excelente revisión de las metodologías usadas en CARM. Proponen el algoritmo TFPC (Total From Partial Classification) y publican un software que lo implementa, para que todos los investigadores puedan discretizar/normalizar los datasets de clasificación que utilizan en sus experimentos, con el objetivo de que se puedan realizar mejores comparativas en los artículos científicos. Se trata de LUCS-KDD (Coenen, 2003).

Thabtah et al., 2006 proponen el algoritmo MCAR (Multi-class Classification based on Association Rules). Según sus experimentos, mejora la fase de ordenamiento de reglas de clasificación con respecto a las anteriores propuestas, mejorando también los tiempos de ejecución al realizar sólo una lectura del dataset de clasificación.

Y. J. Wang et al. (2008) indican que a pesar de las ventajas de precisión y eficiencia de los algoritmos de CARM, el alto número de CARs generado por esos algoritmos dificulta su uso. Proponen utilizar únicamente las reglas más significativas.

Hernández León, Carrasco Ochoa et al. (2010; 2012) destacan que el uso de reglas de clasificación asociativa se extiende a muchas áreas de investigación: predicción de los tipos de interacción proteína-proteína, pre-

dicción del comportamiento del consumidor, segmentación de texto, clasificación de texto, determinación de los tipos de unión de empalme de ADN, clasificación de mamografías, detección de cáncer de mama, obtención de grupos de genes mínimos potencialmente responsables del desarrollo de cáncer, análisis de datos del censo georreferenciado del mundo real o la diferenciación de células madre mesenquimales en mamíferos, entre otras. Proponen el algoritmo CAR-IC (Classification Association Rules with Inexact Coverage), que permite descubrir reglas específicas con alta confianza al basarse en la métrica Netconf (Ahn y Kim, 2004). Comparan la eficiencia de su propuesta con otras existentes utilizando 15 datasets de clasificación de UCI.

Pinho Lucas et al., 2012 introducen las reglas de clasificación asociativa en un sistema de recomendación. D. Nguyen, L. Nguyen, Vo y Hong (2015) y D. Nguyen, L. Nguyen, Vo y Pedrycz (2016) argumentan que la mayoría de reglas de clasificación asociativa generadas por los algoritmos existentes son redundantes o insignificantes. El elevado número de reglas obtenidas puede confundir al analista, además de reducir la eficiencia de los algoritmos de CARM. Proponen el uso de restricciones de clase para reducir la cantidad de reglas descubiertas.

Discretización

La discretización es una técnica que reduce las dimensiones de los datasets de clasificación al agrupar los valores de los atributos numéricos en intervalos o categorías. Es una fase necesaria cuando se trabaja con atributos numéricos de grandes rangos pero, como toda transformación con pérdidas, pierde gran parte de la información que contienen los datos crudos.

Se lleva a cabo en la fase de transformación de un proceso de KDD, por lo que es previo a cualquier análisis de minería de datos. Cada investigador puede utilizar el método de discretización que considere más adecuado, ajustando los parámetros que sean necesarios, por lo que los datasets de clasificación finalmente analizados mediante DM no siempre son los mismos. Esto dificulta la comparación de eficiencia de las diferentes propuestas revisadas, *LUCS-KDD discretised/normalised ARM and CARM Data Library* (2003) ofrecen a los investigadores un software con el que discretizar/normalizar datasets de clasificación y poder hacer comparaciones

utilizando exactamente los mismos datos.

Fayyad e Irani (1993) proponen una heurística para crear intervalos ajustados a los valores de un atributo continuo de modo que se minimice su entropía. Dougherty et al. (1995) hacen una discusión sobre los métodos de discretización existentes y utilizan 16 datasets de clasificación de *UCI Machine Learning Repository* (2013) para comparar su eficiencia. Srikant y Agrawal (1996a) hacen una detallada descripción de los problemas que presenta el análisis de atributos cuantitativos mediante técnicas de ARM, basadas en atributos categóricos. También discuten los problemas que genera la discretización de los atributos categóricos al trabajar con los conceptos de soporte y confianza. Khanmohammadi y Chou, 2016 proponen un método para discretizar atributos numéricos, como la presión sanguínea en bases de datos médicas.

La regresión estadística es utilizada para buscar tendencias en colecciones de datos numéricos, se puede aplicar un análisis de regresión a los valores de un atributo numérico para convertirlo en otro atributo con menor rango. Existen más técnicas, como el clustering, que ayudan a agrupar grandes colecciones de datos distintos. Sin embargo, todas las transformaciones expuestas en esta sección son agresivas con los datos originales, modifican los datos que se han observado. Por ejemplo, si los eventos $\{x = 1.3\}$ y $\{x = 1.1\}$ se guardan como dos eventos $\{x \in [1, 2)\}$, y el primero estaba clasificado como c_1 y el segundo como c_2 ¿cómo se puede guardar esta ambigüedad? No se guarda de ningún modo, se decide qué clase se asocia a los valores de x del intervalo creado y se utiliza únicamente esa clase al reconstruir el dataset de clasificación con los valores discretizados.

Selección de instancias y/o atributos

La selección de instancias y la selección de atributos son otras técnicas usadas en un experimento de clasificación para reducir el número de datos a analizar perdiendo el mínimo de información posible.

En *Estadística* hay técnicas, como el análisis de componentes y la regresión, que permiten descubrir el grado de dependencia entre dos variables. Si se descubre que la clase en estudio es independiente de uno de los atributos incorporados al dataset de clasificación, se puede prescindir de ese atributo por no proporcionar información para la clasificación.

Si de un dataset muy grande se obtiene una muestra representativa, se puede reducir drásticamente el número de instancias a analizar y obtener resultados muy próximos a los que se obtendrían analizando el dataset completo.

Derrac et al. (2010) presentan un algoritmo híbrido para lograr esta reducción, reduciendo simultáneamente el número de instancias y el número de atributos a analizar.

De nuevo se plantea una reducción basada en una transformación de los datos con pérdida de información. Si no se garantiza la independencia entre un atributo y la clase en estudio, aunque su grado de dependencia sea muy pequeño indica que ese atributo tiene cierta influencia sobre la clase. Eliminando esa información se pueden encontrar evidencias que, estando perfectamente clasificadas en el dataset original, presenten ambigüedad en el dataset reducido.

Reducción sin pérdidas de datasets de clasificación

En los capítulos 4, 5 y 6 se muestra cómo se pueden reducir las dimensiones de un dataset de clasificación mediante un algoritmo sin supervisión que selecciona de forma automática las instancias y atributos que más información proporcionan al experimento de clasificación, sin pérdida alguna de la información contenida en el dataset de clasificación original.

3

Primeros resultados

En este capítulo se exponen las propuestas hechas durante este primer periodo de investigación. La sección 3.1 presenta los resultados obtenidos en la investigación desarrollada sobre minería de reglas de asociación. La sección 3.2 presenta los resultados obtenidos en la investigación desarrollada sobre la adaptación de reglas de asociación al problema de clasificación.

3.1. Minería de Reglas de Asociación

En esta sección se muestran las aportaciones más relevantes obtenidas durante la investigación sobre minería de reglas de asociación (ARM).

En la sección 3.1.1 se exponen los resultados obtenidos al implementar el algoritmo Apriori con una relajación de la función de poda y la fusión de unión y poda relajada en una única función.

En la sección 3.1.2 se muestra una optimización del proceso de generación de reglas de asociación.

En la sección 3.1.3 se presenta el algoritmo Apriori2 que, partiendo de un dataset de transacciones, genera un dataset de reglas de asociación con formato de transacción. En el análisis del dataset de reglas de asociación se pueden descubrir las relaciones existentes entre diferentes reglas y comprobar si generan más información que las asociaciones inter-ítems del dataset original.

En la sección 3.1.4 se presentan las reglas de oportunidad, con las que se puede obtener información sobre ítems raros sin modificar el algoritmo de ARM utilizado, y usando muy pocos recursos. Estas reglas pueden usarse en sistemas de recomendación para recoger información sobre cuál es el momento más oportuno para recomendar los ítems raros de los que se ha recogido información.

3.1.1. Implementación de Apriori

El algoritmo Apriori (ver listado 2.6, pág. 36) utiliza las funciones `unión` y `poda` (ver funciones 2.7 y 2.8) para generar el conjunto de candidatos a itemsets frecuentes. En la unión se combinan todos los itemsets frecuentes descubiertos en la anterior iteración, mientras que en la poda se eliminan las combinaciones que, a priori, no pueden ser frecuentes, por contener algún (k-1)-itemset infrecuente.

Al ejecutar el algoritmo, el número de llamadas a estas funciones está limitado por la longitud de la transacción más larga que contiene el dataset, generalmente un número pequeño. Sin embargo, debido a que el número de itemsets frecuentes de un dataset crece exponencialmente, en la ejecución de la función `unión` se reserva memoria para millones de candidatos que, en la función `poda`, se ha de liberar al descubrir que el candidato no generará un itemset frecuente. Al tiempo empleado en reservar millones de porciones de memoria para liberarla inmediatamente después sin haberla utilizado, se suma el tiempo empleado en recorrer dos veces –una en la `unión` y otra en la `poda`– una estructura de tipo árbol con millones de hojas.

Listado 3.1: Función de unión y poda

```
insert into C_k
select c = {p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}}
from L_{k-1} p, L_{k-1} q
where  (p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1})
       and forall (k-1)-subsets s of c
              s ∈ L_{k-1};
```

Si se realiza la `unión` y la `poda` en una única función (ver función 3.1) se dota de más eficiencia al algoritmo, debido a que no se reserva memoria para los candidatos que ya sabemos que no serán frecuentes, por lo que

no se tendrá que liberar la memoria reservada y no hay que recorrer todos los candidatos del árbol para descubrir cuáles no pueden llegar a ser frecuentes.

Listado 3.2: Función de poda relajada

```
forall itemsets do
  forall 2-subsets s of c do
    if (s ∉ L_2) then
      delete c from C
```

En la fase de poda, para cada posible candidato formado por la combinación de dos itemsets frecuentes, se ha de buscar (k-1) nodos en el árbol $\mathcal{L}_{k-1}$ para confirmar si un candidato, a priori, puede ser frecuente. Se puede obtener un resultado aproximado que requiere menos tiempo de ejecución, relajando la poda (ver función 3.2) de modo que sólo se compruebe que los dos últimos ítems del candidato aparecen conjuntamente de forma frecuente en el dataset.

Listado 3.3: Función de unión y poda relajada

```
insert into C_k
select c = {p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}}
from L_{k-1}p, L_{k-1}q
where (p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1})
      and (p.item_{k-1}, q.item_{k-1}) ∈ L_2;
```

El listado 3.3 muestra cómo realizar, en una función, la fase de unión y la de poda relajada. El uso de esta función, en lugar de las dos propuestas en el algoritmo original, ahorra millones de instrucciones de reserva de memoria, de liberación de memoria y de movimientos dentro de un extenso árbol para localizar uno de sus nodos. Este ahorro se refleja en tiempos de ejecución

Comparación de implementaciones

La minería de reglas de asociación va adquiriendo más atención en el mundo científico gracias a la tecnología, que cada vez permite adquirir, almacenar y distribuir mayor cantidad de datos y proporciona mejores

herramientas para su análisis. Todas las disciplinas científicas se benefician de esta situación, y es en la *Informática* donde se centran las mayores expectativas sobre este tema.

La implementación de cualquier aportación teórica es esencial para su validación. Una mala implementación puede provocar que no se siga con un desarrollo teórico que en la práctica da «malos» resultados. Se pretende tratar con enormes cantidades de datos y se quiere hacer en un tiempo prudencial, obteniendo el máximo de información, conocimiento de calidad que podría pasar desapercibido entre tal cantidad de datos.

La investigación de Bodon (2003, 2004, 2005, 2006) y de Rácz et al. (2005) aporta mucho conocimiento sobre los problemas informáticos derivados de la implementación de algoritmos de ARM. Bodon es uno de los desarrolladores de A `C++` Frequent Itemset Mining Template Library[1], cuyo objetivo es dotar a los investigadores de herramientas específicas para la tarea de FIM. Es una librería escrita en `C++` que utiliza la librería estándar STL para garantizar su funcionalidad y modularidad.

Bodon mantiene una página web con su propia implementación de Apriori[2], donde se pueden obtener los ejecutables para Linux y Windows, el código fuente y documentación detallada sobre el código y algunas consideraciones teóricas. La comparativa realizada en esta sección se completa con otras dos implementaciones de Apriori, la primera realizada por Bart Goethals[3] y la segunda por Christian Borgelt[4].[5] En la sección «Implementaciones» de FIMI[6] hay mucho material sobre éste y otros algoritmos de minería de itemsets frecuentes.

Borgelt desarrolla su código en `C` mientras que Bodon y Goethals utilizan `C` y `C++`. `C` es un lenguaje muy eficiente aunque el código generado en `C` puede llegar a ser muy difícil de mantener y modificar. `C++` puede utilizar la eficiencia de `C` y trabajar con `clases`, que facilitan enormemente el desarrollo, mantenimiento y modificación del código. En las conclusiones de esta sección influye mucho esta circunstancia, el código de Borgelt gana en eficiencia frente a las otras dos implementaciones, pero su mantenimiento

[1] `http://www.cs.bme.hu/~bodon/en/fim_env/index.html`
[2] `http://www.cs.bme.hu/~bodon/en/apriori/`
[3] `http://adrem.ua.ac.be/~goethals/software/`
[4] `http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html#assoc`
[5] En el `DVD` están las tres implementaciones, en la carpeta `bin`.
[6] `http://fimi.ua.ac.be/src/`

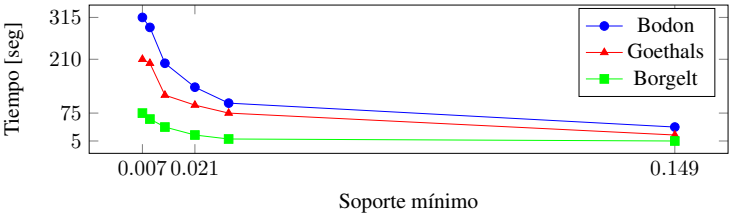
o expansión para añadir nuestras aportaciones requiere de muchas más horas de trabajo, contando con el riesgo de hacer un mal uso de una *estructura* que no es capaz de autogestionarse, mientras que las `clases` sí que permiten tener un control total sobre ellas.

La comparacion se llevó a cabo con dos datasets publicados en *FIM Data Repository* (2004), `BMS-POS.dat` y `BMS-View1.dat`, y con datos propios de un servidor.Los datos propios fueron seleccionados y preprocesados a partir del archivo log del servidor, en la etapa de transformación se convirtieron las páginas web –largas cadenas de caracteres– en códigos numéricos, creando archivos con los que poder reconocer las páginas a partir del código utilizado en la fase de minería de datos. Con `BMS-POS.dat` y `BMS-View1.dat` no se pudo hacer lo mismo por no tener información sobre las páginas web correspondientes a los números utilizados para codificarlas.

Las ejecuciones se hicieron en similares condiciones, para poder hacer comparaciones válidas, en un Intel Pentium 4 con procesador a 2.53GHz, con 512KB L2 de caché y 1.5GB de RAM, ejecutando MS-Windows 2003 Server SP2 de 32 bits.Uno de los objetivos de esta tesis es que se puedan aplicar todas las propuestas realizadas sin necesidad de un supercomputador, este objetivo se ha mantenido hasta el final de esta investigación.

Para decidir el soporte mínimo a utilizar en cada prueba se hicieron antes algunos ensayos comenzando con soporte mínimo nulo –con lo que se buscan todos los itemsets presentes en $\mathcal{D}$– obteniendo en los tres casos problemas de desbordamiento de memoria y pérdida de la información obtenida tras la salida inesperada de la ejecución de los programas. Se fue incrementando el soporte mínimo hasta alcanzar la primera ejecución completa y, a partir de ahí, se decidió el rango de valores a usar como soporte mínimo para cada uno de los datasets analizado.

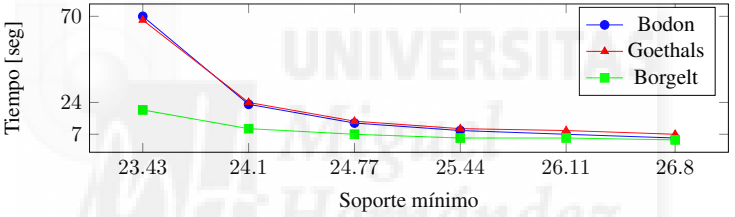
Figura 3.1: Tiempos de ejecución con la colección `BMS-POS.dat`



El dataset `BMS-POS.dat` (11.4MB) tiene 515,597 transacciones con 1,657 ítems distintos y un total de 3,360,020 ítems. Los tiempos de ejecución para valores del soporte mínimo entre 0.007 % y 0.149 % se muestran en la figura 3.1.

La figura 3.2 muestra los tiempos de ejecución obtenidos al analizar el dataset `BMS-View1.dat` (2.1MB), que contiene 149,639 líneas con el par {transacción, ítem} –representación vertical de $\mathcal{D}$– con 497 ítems distintos. Se convirtió en un archivo en que cada línea contuviera sólo una transacción–representación horizontal de $\mathcal{D}$– `BMS-View2.dat`[7] (1MB), pues las implementaciones utilizadas leen este formato de archivo, obteniendo un total de 59,602 transacciones.

Figura 3.2: Tiempos de ejecución con la colección `BMS-View1.dat`

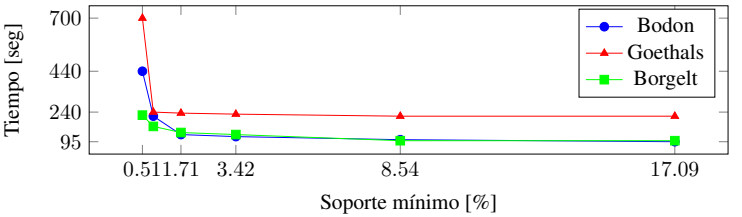


Los datos propios (5.1MB) recogen 585,149 transacciones con 2,213 ítems distintos y un total de 1,732,617 ítems. Los tiempo de ejecución de las implementaciones puestas a prueba con este dataset $\mathcal{D}$ se muestran en la figura 3.3, con soporte mínimo entre 0.51 % y 17.09 %. En este caso se observan tiempos de ejecución muy grandes –alcanzando los 11 minutos al usar la implementación de Goethals con soporte mínimo del 0.51 %– lo que pone en evidencia, al compararlo con los tiempos de ejecución de `BMS-POS.dat`, que además del número de transacciones y de datos a procesar, en los procesos de minería de reglas de asociación son importantes las relaciones presentes en sus ítems, es decir, puede afectar más al análisis la propia distribución de los datos que su cantidad.

La implementación de Borgelt, realizada con `C`, es la más eficiente de las tres en todas las ejecuciones, excepto para valores intermedios del soporte mínimo –entre 1.71 % y 17.09 %– en el análisis de nuestros datos, en

[7] Disponible en el `DVD` adjunto, en la carpeta `datos`

Figura 3.3: Tiempos de ejecución con la colección de datos propios



que los tiempos son sensiblemente menores al usar la implementación de Bodon.

3.1.2. Generación de reglas asociación

Una vez resuelta la fase de minería de itemsets frecuentes, se utiliza el conjunto $\mathcal{L}$ para obtener las reglas de asociación que contiene el dataset $\mathcal{D}$. Ya se han descubierto todos los itemsets con soporte superior al mínimo prefijado, $minSup$, pero aún no se conocen las reglas de asociación. Para ello se ha de recorrer todo $\mathcal{L}$, de cada uno de los itemsets frecuentes que contiene obtener todas sus particiones en dos subconjuntos no vacíos y comparar la confianza de las dos reglas que genera con el umbral mínimo establecido para el estudio, $minConf$.

Apriori es un algoritmo hermoso, sencillo y moldeable, de ahí la atención que ha recibido por parte de la comunidad científica. Está descrito de un modo muy general por lo que se puede modificar con facilidad cualquiera de sus partes sin perder su esencia. La mayoría de los trabajos expuestos en las secciones 2.1.3 y 2.1.4 se centran en mejorar la primera fase de este algoritmo, la minería de itemsets frecuentes. Debido a su gran necesidad de recursos y de procesos de lectura/escritura en disco se pueden plantear muchas estrategias para optimizar su ejecución. Si se lee con atención el párrafo anterior se deduce que la fase de generación de reglas de asociación también tiene mucha carga de procesos:

1. Se ha de leer cada itemset frecuente de $\mathcal{L}$, lo que se conseguirá mediante un bucle que recorra todos sus nodos.

2. Cada itemset se divide en un par de subconjuntos no vacíos, X_1 y X_2, que serán el antecedente y consecuente de dos reglas de asociación

diferentes, $X_1 \rightarrow X_2$ y $X_2 \rightarrow X_1$.

3. Para obtener la confianza de cada regla se ha de buscar en $\mathcal{L}$ el soporte de X_1 y de X_2, calcular su cociente y determinar si tienen o no confianza mínima.

Los dos primeros pasos son elementales, sin embargo el paso 3 es más complejo de lo que parece a simple vista. Para encontrar el soporte de cada k-itemset se necesitan k búsquedas en $\mathcal{L}$. Primero se debe localizar en $\mathcal{L}_1$ el nodo que representa a su primer ítem, una vez encontrado se entra en la rama que se deriva de él y se busca en $\mathcal{L}_2$ el nodo que representa a su segundo ítem, siguiendo el proceso hasta localizar su k-ésimo ítem en $\mathcal{L}_k$. Aunque se usen algoritmos de búsqueda eficientes en cada rama, se han de hacer cientos de comparaciones para encontrar en $\mathcal{L}$ cada uno de los itemsets obtenidos en el paso 2. En su implementación se pueden usar diferentes estrategias de optimización, en función de las estructuras y algoritmos utilizados.

`genrules()`

La función `genrules()` encuentra las reglas que generan los itemsets frecuentes descubiertos en la primera fase de los algoritmos de ARM. Aparentemente es la parte más rápida del algoritmo Apriori. El algoritmo original propuesto por Agrawal y Srikant (1994) se muestra en el listado 2.9.

`genrules()` recibe como parámetros dos k-itemsets, el primero es siempre l_k y el segundo es un subconjunto de l_k, del que se extraerán los antecedentes de las *reglas derivadas* de l_k. En cada llamada a `genrules()` se obtiene la confianza de una regla del tipo $a_m \Rightarrow c_i$, donde $c_i = l_k - a_m$, lo que supone dos llamadas a la función `soporte()`, función que ha de recorrer $\mathcal{L}$ hasta el nivel determinado por el número de ítems del k-itemset que recibe como parámetro, realizando $k + (k - m)$ búsquedas.

Aunque no lo reflejan en la función `genrules()` que publican, Agrawal y Srikant (1994) observaron la siguiente característica:

> *We showed earlier that if $a \Rightarrow (l - a)$ does not hold, neither does $\tilde{a} \Rightarrow (l - \tilde{a})$ for any $\tilde{a} \subset a$. By rewriting, it follows that for a rule $(l - c) \Rightarrow c$ to hold, all rules of the form $(l - \tilde{c}) \Rightarrow \tilde{c}$ must also hold, where $\tilde{c}$ is a non-empty subset of c. For example, if the rule AB $\Rightarrow$ CD holds, then the rules ABC $\Rightarrow$ D and ABD $\Rightarrow$ C must also hold.*

Tras lo cual proponen un segundo método, más rápido cuando $minConf$ es grande: si se detecta que conf(ABC $\Rightarrow$ D) $< minConf$ no es necesario calcular la confianza de las reglas AB $\Rightarrow$ CD, AC $\Rightarrow$ BD, BC $\Rightarrow$ AD, A $\Rightarrow$ BCD, B $\Rightarrow$ ACD, C $\Rightarrow$ ABD. Cuando se quieren descubrir reglas sin altas confianzas, esto no es un adelanto si no una comprobación más que ralentizaría la ejecución del algoritmo.

Se puede mejorar notablemente el rendimiento del algoritmo analizando las características de las reglas de asociación que genera. El siguiente ejemplo ilustra cómo se repiten muchas operaciones de búsqueda en el conjunto $\mathcal{L}$.

> Sea el k-itemset $l_k = \{1, 2, 3, 4, 5, 6\}$.
>
> 1. La ejecución de $genrules(l_k, l_k)$ realizará una llamada recursiva a $genrules(l_k, \{1, 2, 3, 4, 5\})$ y ésta realiza una llamada a $genrules(l_k, \{1, 2, 3, 4\})$, entre otras.
> 2. Posteriormente, la llamada inicial a $genrules(l_k, l_k)$ realizará una llamada a $genrules(l_k, \{1, 2, 3, 4, 6\})$ y ésta llamará de nuevo a $genrules(l_k, \{1, 2, 3, 4\})$, entre otras.
>
> En consecuencia, se llama a $genrules(l_k, \{1, 2, 3, 4\})$ dos veces, lo que generará duplicados de todas las llamadas hechas con los subconjuntos de $\{1, 2, 3, 4\}$ como segundo parámetro. Esto genera un gran número de búsquedas y cálculos repetidos, exponencial en función de k, que no aportan nada al estudio pues se trata de cálculos ya realizados y convenientemente almacenados.

Esta característica hace aún más ineficiente el algoritmo original debido al modo en que se almacena el conjunto de itemsets frecuentes, $\mathcal{L}$. Para buscar el soporte de un k-itemset se debe recorrer $\mathcal{L}$ desde su raíz buscando cada uno de los ítems que forman l_k. Si hacer una búsqueda ya es costoso por tener que hacer k búsquedas individuales, repetirla cuando ya se ha hecho es un desperdicio de recursos.

El listado 3.4 presenta una nueva propuesta, una modificación de la función $genrules()$ en la que se ha respetado la numeración del pseudocódigo original (véase el listado 2.9) para que sea más fácil identificar los cambios introducidos.

Listado 3.4: Función $genrules()$ modificada

```
forall large itemsets l_k, k ≥ 2 do
  supp_l_k = support(l_k);
  call genrules(l_k, l_k, supp_l_k);

// The genrules generates all valid rules ã ⇒ (l_k − ã),
// for all ã ⊂ a_m
procedure genrules(l_k: large k–itemset,
                   a_m: large m–itemset,
                   supp_l_k: double)
x)  static a_m_processed; // set containing the sets of l_k processed
                          // in previous  calls to genrules
x)  if (m == k)
x)    a_m_processed.clear(); // Initialize a_m_processed for each new l_k
1)    A = {(m − 1)-itemsets a_{m−1} | a_{m−1} ⊂ a_m};
2)    forall a_{m−1} ∈ A do begin
x)      if a_{m−1} ∈ a_m_processed then
x)        continue;
x)      a_m_processed.add(a_{m−1});
3)      conf = support_l_k/support(a_{m−1});
4)      if (conf ≥ minconf) then begin
7)        output the rule a_{m−1} ⇒ (l_k − a_{m−1}),
              with confidence = conf
                  and support = support(l_k);
8)      if (m − 1 > 1) then
9)        call genrules(l_k, a_{m−1}, supp_l_k); // to generate rules with
                                                 // subsets of a_{m−1} as the
                                                 // antecedents
10)   end
11) end
```

El primer cambio consiste en añadir como parámetro de $genrules()$ el soporte de l_k. Con ello se evitan numerosas llamadas a la función $soporte()$, aprovechando el momento en que se ejecuta $genrules()$. No se hace ninguna llamada a $soporte(l_k)$ pues al procesar cada k-itemset de $\mathcal{L}$ ya se ha localizado previamente y se puede consultar directamente su soporte en lugar de buscarlo, ahorrando múltiples llamadas recursivas que hace el algoritmo original.

El cambio más relevante es el uso de la variable estática `a_m_processed` y su inicialización cada vez que el bucle entra en un nivel nuevo de $\mathcal{L}$. En el bucle sólo se ha de comprobar si el itemset ya ha sido utilizado, si lo ha sido se ignora y se continúa con el siguiente itemset, si no lo ha sido se anota como utilizado y se sigue el flujo normal del algoritmo original. Se utilizan pocos recursos para guardar este vector en memoria ya que está

basado en un único k-itemset y `a_m_processed` contendrá únicamente sus subconjuntos.

Para probar la eficiencia de esta nueva propuesta frente a la propuesta original se ejecutaron ambos algoritmos, con diferentes soportes y confianzas mínimos, sobre los datasets `foodmart` y `T40I10D100K`, contando el número de reglas que se analizan con cada uno de los métodos y el número de reglas que contiene cada dataset $\mathcal{D}$ en esas condiciones.

Las tablas 3.1 y 3.2 muestran los resultados obtenidos en la ejecución de ambos algoritmos. En las columnas bajo el epígrafe «analizadas» se indica el número de reglas analizadas con cada método y entre paréntesis la relación entre este valor y las reglas finalmente encontradas. También se muestra el tiempo de ejecución en cada una de las situaciones tratadas para constatar que la reducción de cálculos influye en el tiempo necesario para llevarlas a cabo.

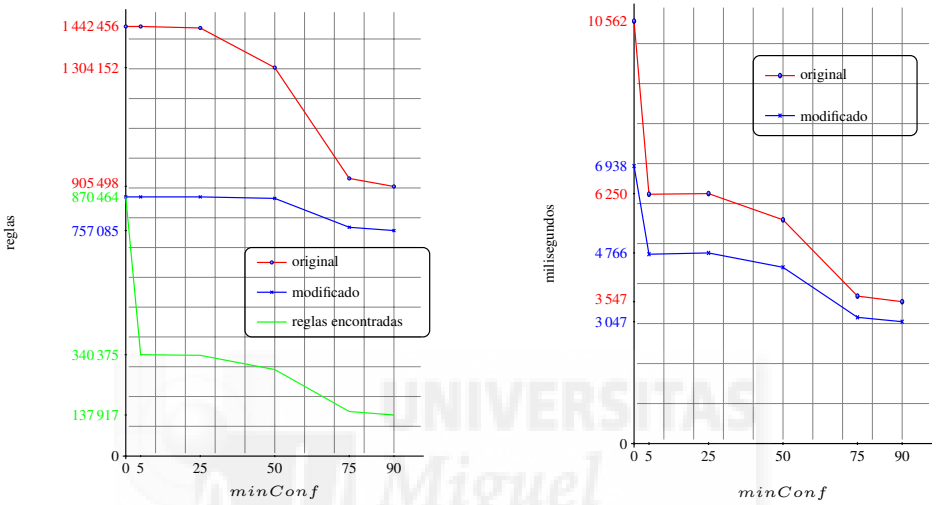
Tabla 3.1: Reglas analizadas y encontradas (foodmart)

			reglas		
			analizadas		encontradas
fichero	$minSup$	$minConf$	original	modificado	
foodmart	0.005 %	90 %	905 498	757 085	137 917
(Cálculos)			(656.6 %)	(548.9 %)	
Tiempo			3.6 sg	3.1 sg	
		75 %	932 697	768 962	149 711
			(623.0 %)	(513.6 %)	
			3.8 sg	3.2 sg	
		50 %	1 304 152	865 132	290 441
			(449.0 %)	(297.9 %)	
			5.7 sg	4.5 sg	
		25 %	1 437 624	870 460	338 221
			(425.1 %)	(257.4 %)	
			6.3 sg	4.8 sg	
		5 %	1 442 456	870 464	340 375
			(423.8 %)	(255.7 %)	
			6.3 sg	4.8 sg	
		1 %	1 442 456	870 464	868 427
			(166.1 %)	(100.2 %)	
			10.8 sg	6.9 sg	
		0 %	1 442 456	870 464	870 464
			(165.7 %)	(100 %)	
			10.7 sg	7.0 sg	

La figura 3.4 ilustra los resultados obtenidos al analizar el almacén `foodmart`. Es fácil observar en estas gráficas que la reducción de cálculos

realizados y el ahorro de tiempo de ejecución son mayores cuanto menor es el umbral de confianza utilizado.

Figura 3.4: `genrules()` original vs. modificado (`foodmart`, $minSup = 0.005\,\%$)



La observación más destacable de estos resultados es la eficiencia del nuevo método frente al original cuando no se impone un umbral de confianza ($minConf = 0\,\%$). En estas circunstancias se obliga al algoritmo a descubrir todas las reglas existentes en $\mathcal{D}$. El algoritmo modificado analiza únicamente las reglas de asociación que contiene el dataset $\mathcal{D}$, mientras que el original analiza un 65.7 % más de las reglas encontradas en el análisis de `foodmart` y hasta un 13,016.0 % más en el peor de los casos estudiados, al analizar `T40I10D100K` con soporte mínimo del 0.5 %.

Al incrementar la confianza mínima se analizan más reglas de las que finalmente serán aceptadas, ya que muchas de ellas no superarán el umbral fijado. A pesar de que se acercan los valores obtenidos usando ambos algoritmos, siempre se produce un gran ahorro al utilizar la nueva propuesta.

Todo esto también se refleja en el tiempo de ejecución, notablemente inferior en todas las pruebas realizadas para el algoritmo modificado, hasta el extremo de tardar menos de 2 minutos en una ejecución en que el algoritmo original empleó más de 237 minutos (`T40I10D100K` con $minSup =$

Tabla 3.2: Reglas analizadas y encontradas (T40I10D100K)

			reglas		
			analizadas		encontradas
fichero	$minSup$	$minConf$	original	modificado	
T40I10D100K	5 %	50 %	30	30	0
(Cálculos)			()	()	
Tiempo			0 msg	0 msg	
	1 %	50 %	5 193 340	398 818	290 273
			(1 789.1 %)	(137.4 %)	
			30.3 sg	4.1 sg	
		0 %	5 912 979	401 096	401 096
			(1 474.2 %)	(100 %)	
			30.3 sg	4.1 sg	
	0.5 %	50 %	606 619 662	6 302 454	4 927 893
			(12 309.9 %)	(127.9 %)	
			4 772.7 sg	91.2 sg	
		0 %	846 495 481	6 453 924	6 453 924
			(13 116.0 %)	(100 %)	
			14 266.6 sg	106.1 sg	

$0.5\,\%$ y $minConf = 0\,\%$). Si se aplica esta propuesta a las transacciones realizadas por un único usuario de un portal web serán aún menores los tiempos, consiguiendo resultados en tiempo real para alimentar un sistema de recomendación web.

3.1.3. Datasets de reglas de asociación

El objetivo de cualquier análisis de ARM es descubrir las reglas de asociación que contiene un dataset $\mathcal{D}$. Una vez descubiertas, se han de ordenar siguiendo algún criterio que permita seleccionar la AR más adecuada para la investigación en estudio.

En el desarrollo de un sistema de recomendación web basado en ARs, como en la mayoría de experimentos que utilizan estas reglas, se ordenan primero por soporte–para tener preparadas antes las que se supone que se van a utilizar en más ocasiones– y en segundo lugar por confianza–para utilizar antes aquellas sobre las que se tiene mayor confianza.

Las ARs utilizadas en un WRS son del tipo "*Si un usuario visita la(s) página(s) X, entonces se le recomendará que visite la(s) página(s) Y*". Se han obtenido analizando el dataset $\mathcal{D}$, cuyas transacciones son las sesiones de navegación de los usuarios del portal web. Son reglas que relacionan las

páginas del portal web, obtenidas al analizar «qué» páginas están relacionadas entre sí desde la perspectiva de los usuarios.

Si se quiere conocer mejor el comportamiento de los usuarios y ya se han descubierto las ARs que relacionan las páginas del portal web, podrían usarse éstas para averiguar «cómo» utiliza el usuario el portal en estudio. Si se sustituyen las transacciones originales del dataset $\mathcal{D}$ por el conjunto de reglas de asociación que satisface cada una de ellas, se obtiene otro dataset, $\mathcal{R}$, en que las sesiones de navegación se han convertido en el conjunto de reglas que verifica un usuario cuando navega por el portal.

Aplicando Apriori a este nuevo dataset, se pueden encontrar familias de reglas que expliquen mejor el comportamiento de los usuarios del portal web. Se puede estudiar «cómo» se comportan los usuarios en lugar de estudiar «qué» páginas relacionan en una sesión de navegación.

Reescribiendo $\mathcal{D}$ para obtener $\mathcal{R}$

Para estudiar el comportamiento de un usuario es preferible analizar qué reglas «*verifica*» en vez de centrar el análisis en los ítems que forman dichas regla. Supóngase que un usuario visita los lunes un portal web solicitando las páginas A, B y C y un conjunto de páginas P_1. Los viernes este usuario visita las páginas A, G y H y un conjunto de páginas P_2. Supóngase también que las páginas de P_1 y P_2 no están relacionadas frecuentemente con la página A, y que el sistema de recomendación web implementado no considera variables temporales, por lo que no puede determinar si es lunes o viernes. Cuando el usuario entra en el portal web y solicita la página A es muy probable que se le recomiende visitar la página B o G si se están usando los métodos tradicionales de recomendación. Si el usuario visita a continuación G, el sistema recomendará la página H, a continuación alguna de las páginas de P_2 y por último la página B con menor probabilidad. De cualquier modo el hecho de que visite la página A no es tan significativo como la posibilidad de usar las reglas de comportamiento ya guardadas en nuestro sistema, las reglas que contienen la relación entre los k-itemsets AGH y P_2.

El planteamiento de esta investigación es descubrir si se pueden analizar las reglas que se verifican en un portal web de forma independiente al número de páginas que contiene cada regla. Proporcionar un método para medir la relación existente entre las reglas de comportamiento de los

usuarios de un portal web usando únicamente sus sesiones de navegación.

Estudiando las reglas verificadas por cada transacción del dataset original $\mathcal{D}$, se puede definir el conjunto de reglas $\mathcal{R}$, que contiene una línea por cada transacción de $\mathcal{D}$. Cada línea de $\mathcal{R}$ contiene las reglas que verifica la transacción original. El objetivo de esta conversión, $\mathcal{D} \hookrightarrow \mathcal{R}$, es analizar en la siguiente fase del análisis las reglas que contiene $\mathcal{R}$ utilizando el algoritmo Apriori.

Inicialmente, esta tarea es simplemente exploratoria y se necesitan muchas horas para convertir el dataset original $\mathcal{D}$ en grandes datasets de reglas $\mathcal{R}$ con diferentes umbrales de soporte y confianza. Los datasets de reglas tienden a ser mucho más grandes que los de transacciones. Si una transacción verifica una regla que contiene un determinado k-itemset l_k, también verificará todas las reglas de los itemsets contenidos en l_k. Por ejemplo, si la transacción $ABCD$ genera una regla, también generará las cincuenta reglas que se deducen de sus subconjuntos:

- 14 reglas de 4 ítems: $ABC \rightarrow D$, $ABD \rightarrow C$, $\ldots D \rightarrow ABC$
- 24 reglas de 3 ítems: $AB \rightarrow C$, $AB \rightarrow D$, $\ldots D \rightarrow BC$
- 12 reglas de 2 ítems: $A \rightarrow B$, $A \rightarrow C \ldots D \rightarrow C$

Fijando un umbral de confianza bajo, una transacción de $\mathcal{D}$ con dos ítems genera una línea con 2 reglas en $\mathcal{R}$. Una transacción con 3 ítems genera 12 reglas. Una transacción con 4 ítems genera 50 reglas. El dataset de reglas $\mathcal{R}$ crece exponencialmente en función de la longitud de las transacciones del dataset de transacciones $\mathcal{D}$.

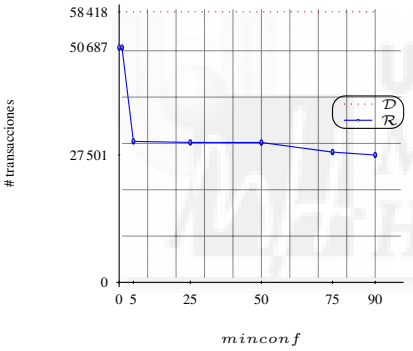
Antes de aplicar el algoritmo Apriori sobre un dataset mucho más grande que el original, es conveniente hacer un análisis previo sobre las diferencias existentes entre los datasets $\mathcal{D}$ y $\mathcal{R}$.

Análisis descriptivo de las diferencias existentes entre $\mathcal{D}$ y $\mathcal{R}$

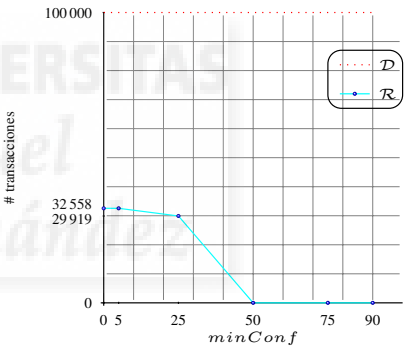
Para estudiar las diferencias entre $\mathcal{D}$ y $\mathcal{R}$, se transformaron los datasets `foodmart` y `T40I10D100K` en datasets de reglas. En la primera revisión de los datasets de reglas se observó que contienen muchas líneas en blanco. Las transacciones que no verifican ninguna de las reglas de asociación encontrada no generan información, por lo que la línea del archivo en que se guarda aparece vacía.

En la sección 3.1.2 se expone una modificación del algoritmo de generación de reglas de asociación, con la que se puede trabajar con umbrales de confianza muy pequeños, e incluso nulos. La razón por la que se buscó esta reducción, que en algunos casos consistía en la eliminación del umbral, se entiende mejor al revisar los resultados obtenidos (ver figura 3.5 y tabla 3.3). La proporción de transacciones originales que no generan reglas es mayor cuanto mayor es el umbral de confianza fijado, se ha de trabajar con valores pequeños de $minConf$ para utilizar la máxima cantidad de datos recogidos en el experimento.

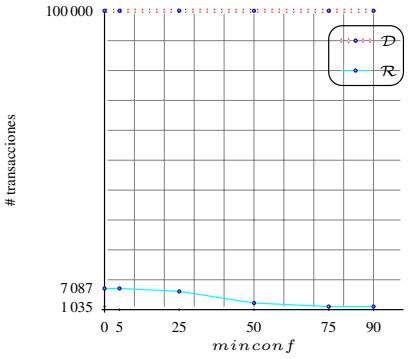
Figura 3.5: $\mathcal{D}$ vs $\mathcal{R}$



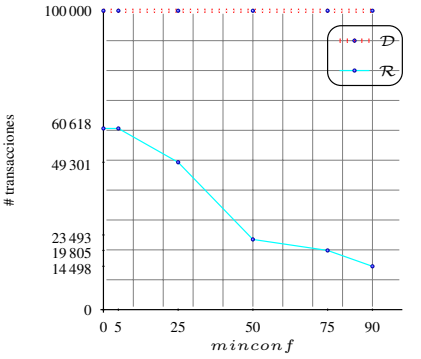
(a) foodmart 0.005 %



(b) T40I10D100K 5 %



(c) T10I4D100K 1 %



(d) T10I4D100K 0.5 %

El número de transacciones que contienen reglas, es decir, el número de transacciones que aportan información, es inversamente proporcional al soporte mínimo y a la confianza mínima fijados.

Por ejemplo, en la tabla 3.3 se observa que, al usar $minSup = 5\,\%$ y $minConf = 25\,\%$ en el dataset `T40I10D100K`, las reglas obtenidas se derivan únicamente de un 29.9 % de las transacciones del dataset $\mathcal{D}$, de lo que se deduce que más del 70 % de las transacciones originales son completamente ignoradas por el análisis. Utilizando $minSup = 0.5\,\%$ y $minConf = 0\,\%$ se aprovecha información de más del 60 % de las transacciones de $\mathcal{D}$.

Tabla 3.3: $\mathcal{D}$ vs $\mathcal{R}$

			transacciones		
fichero	$minSup$	$minConf$	$\mathcal{D}$	$\mathcal{R}$	$\mathcal{D}$ útil
foodmart	0.005 %	90 %	58 418	27 501	47.1 %
		75 %		28 125	48.1 %
		50 %		30 201	51.7 %
		25 %		30 212	51.7 %
		5 %		30 440	52.1 %
		1 %		50 687	86.8 %
		0 %		50 687	86.8. %
T40I10D100K	5 %	50 %	100 000	0	0 %
		25 %		29 919	29.9 %
		5 %		32 558	32.6 %
		0 %		32 558	32.6 %
	1 %	50 %		92 227	92.2 %
		25 %		99 956	100.0 %
		5 %		99 956	100.0 %
		1 %		99 956	100.0 %
		0 %		99 956	100.0 %
	0.5 %	50 %		99 835	99.8 %
T10I4D100K	5 %	50 %	100 000	0	0 %
		25 %		15 475	15.5 %
		5 %		32 558	32.6 %
	1 %	50 %		2 211	2.2 %
		25 %		6 109	6.1 %
		5 %		7 087	7.1 %
		1 %		7 087	7.1 %
		0 %		7 087	7.1 %
	0.5 %	50 %		23 493	23.5 %
		25 %		49 301	49.3 %
		5 %		60 618	60.6 %
		1 %		60 618	60.6 %
		0 %		60 618	60.6 %

Existen muchas aproximaciones en el estado del arte que buscan la reducción de las dimensiones del problema a tratar, utilizando técnicas estadísticas basadas en la frecuencia de los ítems. La más utilizada es el uso de soporte mínimo. Sin embargo, no se tienen en cuenta ciertos detalles, como el que descubren los resultados expuestos en esta sección. Al incrementar el soporte mínimo en un análisis de ARM, las reglas obtenidas sólo reflejan el comportamiento de un reducido grupo de individuos de la población en estudio.

En este trabajo se usaron valores para el soporte y confianza mínimos que pudieran ser gestionados con ordenadores personales convencionales, sin generar problemas de falta de recursos, de modo que se pueda atender al mayor número de individuos de la población en estudio. El tamaño de los datasets $\mathcal{R}$ obtenidos con estos parámetros es tan grande que no se pueden analizar directamente, por lo que era necesario encontrar una estrategia para dividir $\mathcal{R}$ en diferentes datasets que puedan ser analizados convenientemente.

Algoritmo Apriori2

Existen muchas investigaciones proponiendo métodos para dividir los datasets originales de transacciones, pero todas se basan en un alto control y clasificación del conjunto de ítems disponibles en el sistema. Si se quiere utilizar sólo la información proporcionada por los datos se han de buscar estrategias que no utilicen otro tipo de información.

Para lograr este objetivo se propone el uso del algoritmo Apriori2 . Se definen un conjunto de *familias de reglas* para dividir el dataset $\mathcal{R}$ y agrupar aquellas reglas que los usuarios relacionan entre sí cuando interactúan con el sistema. A continuación se describe el algoritmo propuesto.

1. Seleccionar la regla con mayor soporte, en caso de empate seleccionar la regla con mayor confianza y, si de nuevo hay empate, seleccionar la regla que contenga más ítems.

$$R_1 = max_i\{soporte(R_i)\} \wedge$$
$$(confianza(R_1) = max_j\{confianza(R_j) | soporte(R_j) = soporte(R_1)\})$$

   Esta regla será el elemento principal de $\mathcal{F}_1$, la primera familia de reglas.

2. Dividir $\mathcal{R}$ en subconjuntos de reglas. El primer dataset, $\mathcal{R}_1$, contendrá todas las líneas de $\mathcal{R}$ que tengan la regla R_1 y el segundo dataset, $\mathcal{R}_\infty$, contendrá el resto de líneas de $\mathcal{R}$.

3. Ejecutar el algoritmo Apriori sobre $\mathcal{R}_1$ y aplicar de nuevo el paso 1 para seleccionar R_2, la regla de $\mathcal{R}_1$ que aparece con mayor frecuencia junto a R_1.

4. Comprobar el soporte de R_2 en $\mathcal{R}_\infty$: si el soporte de R_2 en $\mathcal{R}_1$ es mayor que su soporte en $\mathcal{R}_\infty$ se añade R_2 a la familia $\mathcal{F}_1$; en otro caso se elimina R_2 de $\mathcal{R}_1$.

5. Volver al paso (3) mientras queden reglas no clasificadas en $\mathcal{R}_1$.

Cuando se ha completado la definición de la primera familia de reglas, se eliminan todas las reglas pertenecientes a $\mathcal{F}_1$ del dataset de reglas $\mathcal{R}$, y las transacciones que queden vacías. A continuación se utiliza el mismo procedimiento para generar $\mathcal{F}_2$, la segunda familia de reglas. El algoritmo finaliza cuando $\mathcal{R}_\infty$ no contiene reglas derivadas, es decir, cuando todas las transacciones de $\mathcal{R}_\infty$ contienen únicamente una regla.

Sistema de recomendación basado en Apriori2

Por último, propuse un sistema de recomendación web para aplicar la modificación en dos pasos del algoritmo Apriori.

El objetivo principal de un sistema de recomendación web es obtener recomendaciones personalizadas para los usuarios de un portal web, generando recomendaciones en tiempo real en forma de enlaces. Al aplicar el algoritmo Apriori2 sobre un dataset de sesiones de usuario del portal web, se obtiene $\mathcal{F}$, el conjunto de familias de reglas que verifican los usuarios del portal. Utilizando este conjunto, se puede adaptar la recomendación proporcionada por el sistema, comprobando qué reglas verifica el usuario en su sesión de navegación. El proceso se define a continuación:

1. Un usuario entra en el sistema y selecciona el ítem A.

2. El sistema sólo puede usar la información almacenada sobre los ítems directamente relacionados con A, es decir, las reglas originales cuyo antecedente es A – esta es la recomendación clásica basada en reglas

de asociación. La primera recomendación se basa únicamente en la confianza, el sistema recomendará los consecuentes de las reglas con mayor confianza que tengan al ítem A como antecedente.

3. El usuario selecciona un segundo ítem, B.

4. A partir de ahora se puede usar la nueva información que se ha recogido sobre el comportamiento de la población de usuarios, las reglas que verifican en su visita a nuestro portal web. Se obtienen las reglas derivadas del k-itemset l_k y se busca la familia a la que pertenecen, $\mathcal{F}_i$.

   - Si $\mathcal{F}_i$ contiene reglas derivadas de las reglas verificadas por el usuario en esta visita, entonces hay una confianza del 100 % entre las reglas descubiertas y las reglas verificadas por el usuario. En este caso el sistema recomendará los ítems de las reglas descubiertas con mayor soporte. A diferencia de los métodos clásicos, esta propuesta descubre reglas que no tienen los ítems visitados por el usuario como antecedentes; esto es importante ya que el análisis realizado ignora por completo el orden de selección de los ítems.
   - Si $\mathcal{F}_i$ no contiene reglas derivadas, se usarán las reglas con mayor confianza–y mayor soporte en caso de empate– en relación con las reglas verificadas por el usuario. Adicionalmente, para resolver el problema de encontrar recomendaciones basadas en el antecedente, es posible encontrar reglas con cualquier ítem seleccionado por el usuario. Esto no puede llevarse a cabo con el método clásico.

5. El usuario selecciona un tercer ítem, C.

6. Se repite el tercer paso, buscando una o más familias que contengan las reglas derivadas del k-itemset $l_k = \{A, B, C\}$.

3.1.4. Reglas de Oportunidad

Con la búsqueda de reglas de asociación aplicada a los datos de uso de un portal web, se pretende encontrar patrones de comportamiento que permitan mejorar la navegación a través del portal. Si un usuario solicita

una página web se le puede sugerir que también visite otras páginas que los propios usuarios han visitado en otras ocasiones junto a la solicitada. De este modo, son los propios usuarios quienes asocian las páginas que visitan en una misma sesión, como si las estuvieran agrupando por tener algo en común.

Cuando son muchos los usuarios que agrupan las mismas páginas web, se puede deducir que hay algún motivo para ello y se genera un patrón, pero cuando son pocos los que lo hacen se puede llegar a pensar que la agrupación se ha hecho al azar, sin que exista una relación real entre las páginas agrupadas. De esta idea surge el concepto de soporte, el número de veces que aparece repetida una asociación de páginas web en las sesiones de navegación registradas. Si este número es pequeño, con respecto a cierto umbral fijado por el analista, no se generará una regla de asociación entre esas páginas, según el enfoque clásico de la búsqueda de reglas de asociación.

En un sistema de recomendación de páginas web, el uso del criterio de soporte mínimo penaliza a las páginas web que han sido visitadas pocas veces, entre otras las páginas web de reciente incorporación, pues no tienen suficiente soporte para formar parte de una regla de asociación que genere una sugerencia. Y conforme funciona el sistema de recomendación web, el problema puede acrecentarse pues los enlaces sugeridos se basan siempre en páginas web que ya tienen suficiente soporte.

En minería de reglas de asociación, un ítem con soporte inferior al soporte mínimo se denomina ítem raro. Los algoritmos de ARM analizados en la sección 2.1.3 ignoran por completo los ítems raros, por lo que no pueden aportar información sobre las páginas web con bajo soporte y, por tanto, no ayudan a incorporarlas al sistema de recomendación. En la sección 2.1.4 se exponen trabajos que ayudan a resolver parcialmente el dilema del ítem raro. Sin embargo, las reglas de asociación que proporcionan los ítems raros añadidos al algoritmo siguen teniendo un soporte bajo y no serán incorporados al sistema de recomendación web.

Parte de la información que procesan los WRS provienen de la ejecución de algoritmos de ARM, que permiten extraer información de uso real del portal web. Estos algoritmos, encuentran primero todos los itemsets frecuentes y después extraen las reglas de asociación que superen el umbral de confianza mínima.

Esto da muy poco juego a las páginas nuevas que se incorporan al por-

tal, y a aquellas que son menos frecuentes, lo que no garantiza que sean de menor interés. Para incluir estas páginas entre las recomendaciones, se puede incorporar al análisis una ponderación de las páginas web del portal, de modo que se fuerce su incorporación al conjunto de itemsets frecuentes, $\mathcal{L}$. Sin embargo, esto implica introducir información al análisis que no procede de los datos obtenidos a partir del uso del portarl web.

En esta tesis se pretende extraer la máxima información de los datos obtenidos, sin añadir información artificial. Será el uso real de los usuarios quien determine qué ítems raros se pueden sugerir mediante el WRS. Para averiguar cómo se usan las páginas con poco soporte utilizando algoritmos de ARM, no se pueden usar las reglas de asociación que conocemos, como se entenderá con el siguientes ejemplo.

Supóngase que el dataset $\mathcal{D}$ tiene 10 sesiones de navegación. La página A aparece en 9 sesiones, en 2 de las cuales también está la página B, que aparece en 3 sesiones en total. La información que recoge el árbol $\mathcal{L}$ tras ejecutar el algoritmo Apriori es:

$\mathcal{L}_0$	$\mathcal{C}_1$
$A(9)$	$B(2)$
$B(3)$	

Si se fija el soporte mínimo en el 30 % y la confianza mínima en el 50 %, la regla de asociación $A \rightarrow B$ no se descubrirá porque el soporte observado, del 20 %, hace que desaparezca de $\mathcal{C}_1$ la página B al construir$\mathcal{L}_1$.

Si las 3 veces que se visita la página B fuera en una sesión en que también se visita la página A, el árbol $\mathcal{L}$ sería el siguiente:

$\mathcal{L}_0$	$\mathcal{C}_1$
$A(9)$	$B(3)$
$B(3)$	

En este caso sí que se mantiene B en $\mathcal{L}_1$, pero la regla $A \rightarrow B$ no supera la confianza mínima, tiene un 33.3 % de confianza, no se guardará como regla interesante y no se podrá usar en el WRS.

Sin embargo, cabe destacar que, en el primer caso, el 66.7 % de veces que aparece B lo hace en una transacción en la que está A. Y en el segundo caso ocurre el 100 % de las ocasiones. Con el planteamiento clásico de Apriori no se obtiene ninguna regla que tenga como antecedente la página

A. Sin embargo se ha detectado que en el uso del portal existe una fuerte relación entre la existencia de A y de B en la misma sesión:

> *Si en una transacción está el ítem A, no hay ni soporte ni confianza para deducir que es probable que también aparezca el ítem B.* Apriori *no descubre la regla de asociación $A \rightarrow B$. Pero si se quiere sugerir B en un WRS, éste es uno de los mejores momentos para anunciarlo.*

Esta medida puede ser útil en otras ocasiones. Supóngase que se analizan 15 transacciones fijando el soporte mínimo en el 20 %, observando:

$\mathcal{L}_0$	$\mathcal{C}_1$
$A(10)$	$B(5)$
	$C(3)$
$B(10)$	
$C(3)$	

En este caso, la confianza de $A \rightarrow B$ es del 50 %, muy superior al 30 % de confianza que proporciona la regla $A \rightarrow C$. Sin embargo, la página B sólo se solicita conjuntamente con A el 50 % de las veces que es visitada, frente al 100 % que muestra C. Si sólo se pudiera recomendar un enlace al usuario que ya ha solicitado la página A, el enfoque clásico de Apriori proporciona la página B como idónea, aunque se trate de un buen momento para sugerir la visita a la página C, pues siempre se ha visitado conjuntamente con A.

El uso de soporte mínimo en la búsqueda de reglas de asociación, aunque es necesario para evitar que sea inabordable el estudio mediante computadores, provoca la pérdida de información sobre un gran número de páginas del portal en estudio. La consecuencia más directa cuando se usa para alimentar un WRS, es que sólo se sugiere visitar las páginas que ya son frecuentes, con lo que su frecuencia de uso crece. Esto hace que decrezca la frecuencia relativa de visitas a las páginas menos frecuentes. Para ser capaces de recomendar cualquier página del portal web a partir de los datos de uso, se ha de ser más flexible con el uso del soporte mínimo.

Las reglas de asociación se generan sobre las páginas visitadas más frecuentemente, en función de la confianza que ofrecen. Supóngase que las páginas A y B de un portal web son visitadas con frecuencia, mientras que la página C no es de uso frecuente. Las reglas obtenidas al analizar esta situación se utilizan en un WRS del siguiente modo:

> *Si al menos el 50 % de los usuarios que visitan la página A, también visitan en la misma sesión la página B, se genera la regla $A \to B$. Al utilizar esta regla, cuando un usuario está visitando la página A se le sugerirá visitar la página B.*

Para que el WRS sugiera visitar páginas con pocas visitas, se definen las reglas de oportunidad:

> *Si al menos el 50 % de los usuarios que visitan la página C visitan en la misma sesión la página A, se genera la regla $A \rightarrowtail C$.*
> *Al usuario que está visitando la página A se le recomienda que visite la página C. La regla inversa, $C \to A$, sería una regla de asociación. Pero no se genera pues no tiene soporte mínimo. El objetivo de las reglas de oportunidad es generar reglas cuyo consecuente sea un ítem raro. Es poco probable que un usuario visite por sí mismo el consecuente y tendría poca utilidad si se utilizara como antecedente.*

Con esta nueva medida, se propone el algoritmo ORFind, capaz de detectar este nuevo tipo de reglas. Se incrementa considerablemente el porcentaje de ítems sobre el que se obtiene información para el WRS, sin incrementar apenas el uso de recursos del ordenador que realiza el análisis.

1. En primer lugar, el algoritmo debe ser capaz de recoger información de ítems que no superen el soporte mínimo, ítems raros. Si no se usara soporte mínimo se obtendría un árbol $\mathcal{L}$ extremadamente grande, y con información irrelevante que desaparecería al obtener las reglas de asociación con confianza mínima. El soporte mínimo debe tener cierta flexibilidad.

2. Las reglas de oportunidad con más de un antecedente no aportan mayor información al WRS, y generan muchos datos a almacenar por lo que se ignorarán. La explicación está en que si seguimos escribiendo en $\mathcal{L}_i, i > 1$, la frecuencia del ítem raro puede que la confianza de la regla clásica generada crezca pero nunca crecerá la oportunidad del ítem pues es una medida decreciente al avanzar por $\mathcal{L}$.

3. Debe informar de un nuevo tipo de reglas que pueden, o no, superar la confianza mínima.

Listado 3.5: Algoritmo ORFind

```
Input: D, sm (soporte mínimo) y om (oportunidad mínima)
Output: RO (Reglas de Oportunidad) y RA (Reglas de Asociación)

//En el primer nivel se cuenta el soporte de todos los ítems
foreach (transacción T_i ∈ D)
   foreach (item_1 ∈ T_i)
      L_0[item_1]++;

//Generar C_1
foreach (transacción T_i ∈ D)
   foreach (2−itemset ∈ T_i)
      L_0[item_1]−>C_1[item_2]++

//Extraer las reglas de oportunidad
foreach (item_1 ∈ L_0)
   foreach (item_2 ∈ L_0[item_1]−>C_1)
      if (L_0[item_1]−>C_1[item_2] / L_0[item_2] ≥ om)
         RO.add(item_1 ⇀ item_2)

//Purgar L_0 y L_1 y seguir con el algoritmo clásico
...
```

El algoritmo ORFind sólo modifica las dos primeras lecturas del dataset $\mathcal{D}$ del algoritmo Apriori, como se muestra en el listado 3.5.

El objetivo que tiene la introducción de reglas de oportunidad es obtener relaciones que cubran el mayor número de ítems posible. Un sistema de recomendación ha de tener información sobre todos los ítems del sistema para poder recomendar el más adecuado en cada momento. Según el enfoque clásico, esto supone reducir el soporte mínimo a 0, con lo que se descubre información sobre todos los ítems. Pero los datasets grandes o con un gran número de ítems distintos no pueden analizarse si no se aplica soporte mínimo.

Para observar la incidencia de la obtención de reglas de oportunidad sobre datos de diversa procedencia, se procesaron un dataset con datos reales –BMS-POS– y dos con datos sintéticos –T10I4D100K y T40I10D100K– . Se obtuvieron resultados similares en todos los casos por lo que se exponen sólo los del primer dataset.

El tiempo necesario para obtener las reglas de oportunidad es tan pequeño que no afecta al tiempo total necesario para buscar las reglas de asociación presentes en un dataset de transacciones. Siin embargo, si se

considera la diferencia de tiempo necesaria para ejecutar el algoritmo con distintos soportes mínimos, para lograr información sobre un gran número de ítems del repositorio, es notablemente más rápido trabajar con un soporte mínimo grande y añadir las reglas de oportunidad que trabajar sólo con reglas de asociación con un soporte mínimo más pequeño. Estas diferencias se han constatado al realizar los experimentos, pero no se han registrado los tiempos de ejecución pues no se buscaba dicha mejora en la experiencia realizada.

El menor de los repositorios puestos a prueba fue `T10I4D100K`. Contiene 870 ítems distintos en 100,000 transacciones con 10 ítems de promedio, y un total de 1,010,228 ítems.

La tabla 3.4 muestra los resultados obtenidos al fijar la confianza y oportunidad mínima en el 50 %. Se hizo el experimento reduciendo el soporte mínimo mientras se pudiera llevar a cabo el análisis. La ejecución del algoritmo sobre este dataset se abortaba, por falta de memoria RAM, al utilizar soporte mínimo inferior al 0.3 %. En la primera columna se muestra el umbral de soporte utilizado. Las dos columnas siguientes muestran el número de reglas de asociación encontradas y el porcentaje de ítems de $\mathcal{D}$ que se utilizan en esas reglas. En las dos siguientes se muestra la misma información respecto a las reglas de oportunidad. La última columna, el porcentaje de mejora obtenido al descubrir las reglas de oportunidad, obtenido a partir del número total de ítems de $\mathcal{D}$ de los que se obtiene información, el 83.6 % en todos los análisis mostrados en la tabla.

Se puede observar que al reducir el soporte mínimo con el enfoque clásico, se obtiene información sobre un número mayor de ítems con un aumento exponencial del número de reglas encontradas, lo que dificulta enormemente su análisis.

En la séptima fila, por ejemplo, que usa el soporte mínimo más utilizado en la bibliografía revisada, el 5 %, se obtienen 761,644 reglas de asociación que utilizan el 76.9 % de los ítems de $\mathcal{D}$. Las 141 reglas de oportunidad descubiertas utilizan sólo el 17.8 % de los ítems de $\mathcal{D}$, ítems raros para el algoritmo de ARM. La mejora en este caso es pequeña, del 8.0 %, debido a la gran cobertura que tiene el algoritmo clásico.

Para obtener información sobre un número mayor de ítems se han de reducir los umbrales de confianza y oportunidad mínima. En la tabla 3.5 se observa este efecto y se comprueba que la mejora aportada por las reglas de oportunidad es ligeramente menor pero aún importante. Al fijar la

Tabla 3.4: `T10I4D100K` con un 50 % de confianza y oportunidad mínima

Soporte mínimo (%)	Reglas de Asociación encontradas	Ítems cubiertos (%)	Reglas de Oportunidad encontradas	Ítems cubiertos (%)	Mejora aportada por las ROs (%)
100.0	7	0.6	1,125	83.4	99.3
50.0	1,145	7.5	1,071	81.1	91.1
40.0	4,861	16.7	990	77.8	80.1
30.0	20,775	32.6	800	70.2	60.9
20.0	176,883	50.3	560	55.1	39.8
10.0	333,757	66.0	339	36.7	21.0
5.0	761,644	76.9	141	17.8	8.0
1.0	3,611,429	82.9	20	2.9	0.8
0.5	17,590,740	83,4	6	0.8	0.1
0.3	107,561,757	83,4	6	0.8	0.1

Tabla 3.5: `T10I4D100K` con un 25 % de confianza y oportunidad mínima

Soporte mínimo (%)	Reglas de Asociación encontradas	Ítems cubiertos (%)	Reglas de Oportunidad encontradas	Ítems cubiertos (%)	Mejora aportada por las ROs (%)
100.0	17	1.4	2,913	99.9	98.6
50.0	1,535	21.0	2,710	99.0	78.9
40.0	5,923	34.0	2,482	97.7	65.9
30.0	23,199	53.8	2.004	93.3	46.1
20.0	189,205	72.8	1,328	81.4	27.2
10.0	354,998	88.7	596	53.2	11.2
5.0	808,183	95.5	217	24.8	4.4
1.0	3,895,609	99.4	25	3.3	0.5
0.5	19,642,125	99.8	10	1.5	0.1
0.3	126,446,840	99.8	10	1.5	0.1

confianza y la oportunidad mínima en el 25 %, se obtienen reglas de ambos tipos que utilizan prácticamente todos los ítems del dataset, el 99.9 %. Se obtiene información sobre casi todos los ítems de $\mathcal{D}$, lo que resulta muy útil en un sistema de recomendación.

Las pruebas realizadas con el algoritmo ORFind (ver listado 3.5) descubren información sobre casi el 100 % de los ítems de un dataset sólo si

se usan umbrales bajos de confianza. Esto genera demasiadas reglas de asociación, muchas de ellas con poca confianza, por lo que se implementó el uso de múltiples soportes en base a las propuestas de B. Liu, W. Hsu y Ma (1999) y Kiran y Reddy, 2009. Se realizaron los mismos experimentos sobre datasets medianos y grandes. Con `T10I4D100K` y `T40I10D100K`, que contienen 870 y 942 ítems diferentes respectivamente, se obtuvo información sobre el 100 % de sus ítems, utilizando un umbral de confianza del 50 %. La mejora se debía a que estos métodos no recogen información sobre relaciones infrecuentes de ítems frecuentes.

Con datasets más grandes, se comprueba que no es posible abordar el estudio de todos sus ítems utilizando algoritmos basados en Apriori, como ocurre con el repositorio `kosarak`, que contiene 41,270 ítems distintos. Apriori descubre reglas de asociación que relacionan a 1,544 de sus ítems, mientras que usando soporte múltiple se obtiene información sobre 4,636 ítems. Una gran mejora, pero aún quedan un 89 % de ítems en el dataset sobre los que no se obtiene ningún tipo de información. Este porcentaje sería peor si el número de ítems en estudio fuera mayor, lo que no es anormal en muchos de los portales web existentes.

Para obtener información sobre todos los ítems de un dataset $\mathcal{D}$ con muchos ítems distintos, se ha de prescindir inicialmente del concepto de soporte mínimo. Pero no se debe obviar la capacidad de los ordenadores en que se ejecutará el algoritmo. En algoritmos basados en Apriori, el número de candidatos a 2-itemset es exponencial en función del número de ítems distintos de $\mathcal{D}$. La estructura `FP-Tree`, utilizada en el algoritmo FP-Growth (J. Han, Pei et al., 2000), no genera candidatos, guardando en memoria sólo los 2-itemsets que encuentra en $\mathcal{D}$.

El algoritmo FP-ORFind (ver listado 3.6) muestra cómo leer el dataset $\mathcal{D}$ sin generar candidatos, y obteniendo todas sus reglas de oportunidad. Los grandes datasets de transacciones pueden contener tantos ítems diferentes que el enfoque de Apriori impediría la simple generación de $\mathcal{C}_2$. Los 41,270 ítems de datos `kosarak` generan 851,627,085 candidatos a 2-itemsets, lo que augura un tamaño intratable para la estructura $\mathcal{L}$. La función `Incrementa` de FP-ORFind comprueba si existe el ítem, y lo crea en caso necesario. De este modo se emplea mayor tiempo en la obtención de los 2-itemsets, pero se evita una explosión de candidatos que podría abortar la ejecución del algoritmo por falta de recursos.

Una vez conocidas las reglas de oportunidad que proporciona el algo-

Listado 3.6: Algoritmo FP-ORFind

```
Input: 𝒟, sm (soporte mínimo) y om (oportunidad mínima)
Output: RO (Reglas de Oportunidad)

// Obtener frecuencia de todos los ítemes y 2−itemsets de 𝒟
foreach (transacción T_i ∈ 𝒟)
   foreach (item_1 ∈ T_i)
   {
      Incrementa(FP_1[item_1]);
      foreach ((item_1, item_2) ∈ T_i)
         Incrementa(FP_1[item_1]−>FP_2[item_2]);
   }

// Extraer las Reglas de Oportunidad
foreach (item_1 ∈ FP_1)
   if (FP_1[item_1] ≥ sm) then
      foreach (item_2 ∈ FP_1[item_1]−>FP_2)
         if (FP_1[item_2] < sm y FP_2[item_2] / FP_1[item_2] ≥ om) then
            RO.add(item_1 ⇀ item_2);
```

ritmo 3.6), se ejecutó el algoritmo Apriori clásico a los datasets `T10I4D100K`, `T40I10D100K` y `kosarak`, descubriendo reglas que involucraban a todos los ítems de cada dataset, en menos tiempo del empleado con la ejecución de las propuestas de B. Liu, W. Hsu y Ma y Kiran y Reddy.

Las reglas de asociación obtenidas entre ítems frecuentes tienen el suficiente soporte como para ser utilizadas como patrones de comportamiento del colectivo estudiado. Las reglas de oportunidad descubiertas vinculan a todos los ítems infrecuentes con algún ítem frecuente. Con esta información, un sistema de recomendación puede aprovechar el momento más oportuno para recomendar un ítem infrecuente de cuyo uso no tenemos aún información suficiente.

3.2. Minería de Reglas de Clasificación

En esta sección se exponen las aportaciones más relevantes obtenidas al investigar sobre minería de reglas de clasificación (CRM).

La sección 3.2.1 presenta una transformación aplicable a datasets de clasificación, con la que se reducen significativamente las dimensiones del dataset a analizar. Aplicada sobre dos datasets de clasificación, `mushroom`

y chess, utilizados en los workshops FIMI'03[8] y FIMI'04[9], esta reducción elimina el dilema del ítem raro, con lo que se descubren todas las reglas de clasificación que contienen los datasets de clasificación. Además, el análisis propuesto descubre las reglas de asociación negativas que contiene el dataset, en un tiempo de ejecución significativamente inferior al empleado al analizar directamente el dataset de clasificación original.

En la sección 3.2.2 se introduce la mayor aportación de esta tesis, el análisis de catálogos, desarrollada en los capítulos 4, 5 y 6, que presenta una nueva metodología para tratar la información contenida en los datasets de clasificación.

3.2.1. Reducción de datasets de clasificación

La definición de transacción (véase def. 2.3 en pág. 20) es muy elemental: «conjunto de ítems observados en determinadas circunstancias». Esta simple definición permite modelar muchas colecciones de datos mediante transacciones, desde la clásica «cesta de la compra» hasta el estudio de una sesión de navegación web, pasando por la observación de las características de individuos que tienen enfermedades similares o la detección de fraude en el uso de tarjetas de crédito.

Agrawal, Imielinski et al. (1993a) definieron las transacciones a partir de la «cesta de la compra», y plantearon las bases de la minería de reglas de asociación (ARM). En su artículo, también hacían referencia a las *reglas de clasificación*, planteando que se podían obtener del mismo modo que las reglas de asociación si se les añade una restricción sintáctica: el consecuente no puede ser cualquier ítem, ha de ser una de las *clases* en que se divide la población.

En un experimento de clasificación, el dataset a analizar no contiene transacciones tipo «cesta de la compra», contiene tuplas etiquetadas que son tratadas como transacciones. Las reglas que proporciona el análisis de estos datasets son un subconjunto de las reglas de asociación que proporciona cualquier algoritmo de ARM, son reglas de clasificación. Ya existe un área de conocimiento de ML que aborda el problema de clasificación, y utiliza el término *regla de clasificación* para describir las mismas reglas. Pero los datasets de clasificación que utilizan y el modo en que se utilizan

[8] http://fimi.ua.ac.be/fimi03/

[9] http://fimi.ua.ac.be/fimi04/

difiere del planteamiento propuesto por Agrawal, Imielinski et al. (1993a), por lo que B. Liu, W. Hsu y Ma, 1998 propusieron el término *regla de clasificación asociativa* para diferenciar ambos planteamientos.

Muchos investigadores sobre ARM utilizan datasets de este tipo, que a pesar de ser relativamente pequeños presentan el dilema del ítem raro. El dataset `mushroom` (FIMI) no se puede analizar usando Apriori si se utilizan soportes mínimos bajos, en torno al 1 %. Los investigadores de hace un par de décadas no tenían tecnología para analizar este dataset más a fondo, pero en la actualidad tenemos muchos más recursos, es difícil pensar que un dataset tan «pequeño» no se pueda analizar a fondo con la tecnología actual.

Borgelt (2004) analiza este dataset aplicando un soporte mínimo del 1.23 %. Sólo obtiene información sobre los ítems que aparecen en 100 transacciones o más. Dong y M. Han (2007) aplican entre el 4 % y el 20 % de soporte mínimo. Luo y X.-M. Zhang (2007) usan valores entre el 5 % y el 30 %. Todos ellos renuncian a conocer gran parte de la información que pueden proporcionar algunos de los ítems de un dataset relativamente pequeño.

Si se observa el contenido del dataset `mushroom` (FIMI), de tan solo 570KB, se descubre que no contiene el tipo de transacciones con las que se trabaja en ARM. Sus transacciones presentan una estructura concreta, todas tienen el mismo número de ítems. Además, hay ítems excluyentes, si una transacción tiene el ítem 1 no tiene el ítem 2. Lo mismo ocurre con los conjuntos de ítems $\{3, 4, 5, 6, 7, 8\}$, $\{9, 10, 11\}$ …, $\{113, \ldots, 119\}$. Esta información es inherente a un dataset de clasificación, si un individuo toma el valor x en un atributo, no puede tomar otro valor en el mismo atributo.

Pero los algoritmos de ARM no conocen esta información. Gran parte de la eficiencia que presentan estos algoritmos se debe a su simplicidad, se limitan a comprobar qué ítems están relacionados en las transacciones de un dataset. Sólo filtran las relaciones que tienen poco soporte, y las que tienen poca confianza antes de finalizar el análisis. Se podría modificar el algoritmo para introducir esta información en el análisis de datasets de clasificación, pero esto restaría mucha eficiencia a algoritmos de ARM que sabemos que funcionan muy bien. Otra solución consiste en aplicar un pre-proceso a estos datasets para incorporar al análisis la información estructural de los datasets de clasificación, en concreto la propiedad de unicidad (ver propiedad 2.2).

En todas las transacciones de `mushroom` (FIMI) aparece el ítem 1 o el ítem 2, si se elimina el ítem 1 de todas ellas, se obtiene un dataset con un ítem menos, pero con la misma información. El nuevo dataset de clasificación ya no contiene transacciones de tamaño fijo, las que contenían al ítem 1 tienen un elemento menos, pero esto es algo habitual en cualquier dataset de transacciones. Cuando se interprete el árbol $\mathcal{L}$ será fácil incorporar la información que se ha eliminado del dataset de clasificación original. Si el dataset original tiene M transacciones y en el análisis del dataset de clasificación reducido se observa que el soporte del ítem 2 es m_2, es inmediato descubrir que el soporte del ítem 1 en el dataset de clasificación original es $m_1 = M - m_2$.

Aplicando la misma transformación al resto de atributos representados en el dataset original, se obtiene un dataset de clasificación reducido sin pérdida alguna de información. Eliminando los ítems 1, 3, 9 . . . , 113 de `mushroom` (FIMI), se obtiene un dataset con menos reglas de asociación que el dataset de clasificación original. Pero no se ha perdido la información proporcionada por esos ítems, todas las reglas de asociación del dataset de clasificación original se pueden obtener a partir de las descubiertas en el dataset de clasificación reducido. Bastan unas simples operaciones matemáticas.

Supóngase que se está estudiando una población con tres atributos, cuyos valores se codifican con $\{1, 2\}$ para el primer atributo, $\{3, 4, 5\}$ para el segundo y $\{6, 7\}$ para el tercero. Suponiendo que el dataset de clasificación $\mathcal{D}$ obtenido tiene la máxima variabilidad, el algoritmo Apriori genera el árbol $\mathcal{L}$ mostrado en la figura 3.6, en el que se recogen todas las relaciones de co-ocurrencia encontradas en $\mathcal{D}$. Contiene 28 itemsets con más de un ítem, que son los que pueden generar reglas de asociación. Suponiendo que todos son frecuentes, pueden generar hasta 68 reglas de asociación diferentes.

Si se reduce el dataset, suprimiendo los ítems 1, 3 y 6, se obtiene el árbol $\mathcal{L}$ mostrado en la figura 3.7. Este árbol contiene 6 itemsets con más de un ítem, que pueden generar como máximo 16 reglas de asociación diferentes. La reducción de datos a procesar por el algoritmo de ARM es significativa, lo que puede ayudar a eliminar el dilema del ítem raro.

La información relativa a los ítems 1, 3 y 6 se obtiene restando al número de transacciones del dataset de clasificación original, M, alguno de los valores del árbol $\mathcal{L}$ obtenido a partir del dataset de clasificación redu-

Figura 3.6: Árbol $\mathcal{L}$ del dataset de clasificación original

$1(m_1)$	–	$3(m_{13})$	–	$6(m_{136})$
			–	$7(m_{137})$
	–	$4(m_{14})$	–	$6(m_{146})$
			–	$7(m_{147})$
	–	$5(m_{15})$	–	$6(m_{156})$
			–	$7(m_{157})$
	–	$6(m_{16})$		
	–	$7(m_{17})$		
$2(m_2)$	–	$3(m_{23})$	–	$6(m_{236})$
			–	$7(m_{237})$
	–	$4(m_{24})$	–	$6(m_{246})$
			–	$7(m_{247})$
	–	$5(m_{25})$	–	$6(m_{256})$
			–	$7(m_{257})$
	–	$6(m_{26})$		
	–	$7(m_{27})$		
$3(m_3)$	–	$6(m_{36})$		
	–	$7(m_{37})$		
$4(m_4)$	–	$6(m_{46})$		
	–	$7(m_{47})$		
$5(m_5)$	–	$6(m_{56})$		
	–	$7(m_{57})$		
$6(m_6)$				
$7(m_7)$				

Figura 3.7: Árbol $\mathcal{L}$ del dataset de clasificación reducido

$2(m_2)$	–	$4(m_{24})$	–	$7(m_{247})$
	–	$5(m_{25})$	–	$7(m_{257})$
	–	$7(m_{27})$		
$4(m_4)$	–	$7(m_{47})$		
$5(m_5)$	–	$7(m_{57})$		
$7(m_7)$				

cido. Para obtener el soporte de cualquier itemset de la figura 3.6 basta con tener en cuenta que si sumamos el soporte de los ítems de cada atributo, siempre obtendremos M, el número de transacciones del dataset de clasificación.

$$\begin{array}{rcl} m_1 & = & M - m_2 \\ m_3 & = & M - m_4 - m_5 \\ m_6 & = & M - m_7 \\ & \cdots & \\ m_{23} & = & m_2 - m_{24} - m_{25} \\ m_{13} & = & m_3 - m_{23} \end{array} \tag{3.1}$$

Agrawal, Imielinski et al. (1993a) especifican que para resolver el problema de clasificación usando estos datasets, es necesario añadir una restricción sintáctica, indicando que sólo se generen reglas de asociación cuyo consecuente sea uno de los valores correspondientes a la clase.

Para añadir una restricción sintáctica hay que modificar el algoritmo Apriori en sus dos fases. En la fase de minería de itemsets frecuentes se ha de forzar que sólo se generen itemsets que contengan una etiqueta de clase. Los itemsets que no contienen etiqueta de clase no pueden generar una regla de clasificación. En la fase de generación de reglas de asociación se ha de forzar que sólo se generen las reglas con una etiqueta de clase en su consecuente.

Figura 3.8: Árbol $\mathcal{L}$ para clasificación del dataset de clasificación reducido

$2(m_2)$	$-$	$4(m_{24})$	$-$	$7(m_{247})$
	$-$	$5(m_{25})$	$-$	$7(m_{257})$
	$-$	$7(m_{27})$		

El árbol $\mathcal{L}$ del ejemplo se reduce al mostrado en la figura 3.8, que sólo tiene 5 itemsets con más de un ítem y generará, como máximo, 7 reglas de clasificación, reglas de asociación con la clase como consecuente.

La transformación propuesta reduce significativamente los requisitos de memoria RAM de los algoritmos de ARM al analizar datasets de clasificación. En el ejemplo expuesto, para analizar el dataset de clasificación reducido se necesita sólo un 18 % de la memoria RAM utilizada en el análisis del dataset de clasificación original.

Esta reducción es aplicable sólo si se fija un umbral de soporte mínimo del 0 %, de otro modo no se podrían utilizar las ecuaciones 3.1. Por este motivo, esta propuesta sólo es aplicable a datasets de clasificación cuyos atributos no tengan rangos demasiado extensos.

Refinamiento

Si en lugar de suprimir los ítems 1,3 y 6 del dataset original se eliminan, para cada atributo, el ítem de mayor soporte, se minimiza el tamaño del dataset reducido. El algoritmo aplicado sobre el nuevo dataset, en consecuencia, necesita menos tiempo de ejecución.

Reglas de Asociación Negativas

Para llevar a cabo el análisis de ARM sobre los datasets reducidos con esta metodología, no se puede imponer un umbral de soporte mínimo. Se ha de contar el soporte de todos los k-itemsets de $\mathcal{D}$ para poder aplicar las ecuaciones 3.1 Con datasets que tengan un gran número de ítems distintos no se puede aplicar esta propuesta porque los datasets reducidos suprimen sólo 1 valor por cada atributo del experimento de clasificación.

Los datasets de dimensiones similares a las de `mushroom` (FIMI) se pueden analizar sin problemas aunque se fije el soporte mínimo en el 0 %, sin renunciar a ningún dato del dataset original. En *UCI Machine Learning Repository* (2013) hay muchos datasets de estas características. En estos casos, además de obtener todas las reglas de asociación que contienen, se pueden deducir todas las *reglas de asociación negativas* del dataset original. Basta con obtener un soporte nulo en un itemset de $\mathcal{L}$ para deducir que los ítems que contiene no aparecen relacionados en ninguna transacción del dataset analizado.

Definición 3.1 (Regla de asociación negativa)**.** *Si los ítems A y B no coexisten en ninguna transacción del dataset $\mathcal{D}$, se obtendrá $m_{AB} = 0$ en un análisis con $minSup = 0$. Esto se recoge en la regla de asociación negativa $A \nleftrightarrow B$, que informa al investigador que la presencia de uno de los dos ítems garantiza la ausencia del otro en la misma transacción.*

Las reglas de asociación negativas son muy importantes en ciertos experimentos, pero no han recibido mucha atención de la comunidad científica debido a que en cualquier dataset, el número de reglas negativas que contiene suele ser muy superior al número de reglas de asociación que contiene y, sobre todo, a que no existe ninguna métrica con la que cuantificar la información proporcionada por una regla de asociación negativa.

La regla de asociación negativa "$A \nleftrightarrow B$" se puede expresar como la combinación de reglas "$A \rightarrow \neg B$" $\wedge$ "$B \rightarrow \neg A$". Si se trabaja con un

dataset de clasificación, la información puede ser más útil ya que la regla "$\{A_1 = 1\} \nrightarrow \{A_2 = 4\}$ informa que cuando un individuo toma el valor 1 en el primer atributo, ya se sabe que en el segundo atributo no puede tomar el valor 4, y viceversa. En el ejemplo descrito en esta sección, donde $A_2 = \{3, 4, 5\}$, esta regla informa que si se comprueba en un individuo que $A_1 = 1$, entonces A_2 será, seguro, o 3 o 5.

Validación

Esta propuesta se validó con los datasets `chess` y `mushroom`, publicados en *FIM Data Repository* (2004), codificados para ARM como se indica en la definición 2.25. Se ejecutó el algoritmo Apriori sobre ambos datasets ajustando el soporte mínimo al 0 %. En ambos casos se abortó la ejecución del programa debido a falta de recursos. Se incrementó el soporte mínimo hasta lograr una ejecución completa del algoritmo, al alcanzar el 30 % en el caso de `chess` y el 1 % en el caso de `mushroom`.

A continuación se crearon sus datasets de clasificación reducidos y se volvió a aplicar el algoritmo Apriori con $minSup = 0\,\%$. En ambos casos terminó la ejecución del programa sin problemas de memoria y en tiempos aceptables. La tabla 3.6 muestra los resultados obtenidos. La primera columna contiene el nombre del dataset de clasificación. La segunda muestra el soporte mínimo fijado en el análisis del dataset original. La tercera columna indica el porcentaje de ítems raros que supone haber aplicado ese soporte mínimo, es decir, el porcentaje de ítems del dataset de clasificación original que desaparecen del análisis, sobre los que no se obtendrá ningún tipo de información. En la siguiente columna se indica el tiempo empleado en la ejecución del análisis realizado sobre el dataset original. Las dos siguientes columnas indican el tiempo empleado en la ejecución del análisis utilizando la versión reducida del dataset de clasificación, en primer lugar habiendo suprimido el primer ítem del rango de cada atributo (1), y en segundo lugar habiendo suprimido el ítem de máximo soporte en cada atributo (2).

En la tabla 3.6 se observa que el 23 % de los ítems de `mushroom` y más del 33 % de los de `chess` son totalmente ignorados en el análisis clásico de ARM, si no se utilizan algoritmos que introduzcan en el análisis algún ítem raro. Al reducir el dataset de clasificación, se obtiene información sobre el 100 % de los datos del experimento, empleando significativamente menos

Tabla 3.6: Análisis de un dataset original vs su versión reducida

	Dataset original		(1)	(2)
	Soporte mínimo (%)	ítems raros (%)	Tiempo (sg)	
mushroom	1	23.0	2,130	183	140
chess	30	33.3	7,660	192	117

tiempo para llevar a cabo el análisis.

En el caso de mushroom, los investigadores que utilizan un soporte mínimo del 1 % o superior, obtendrán información sobre un máximo de 119-27=92 de los ítems que contiene el dataset de clasificación original, ya que contiene al menos 27 ítems raros en estas circunstancias. La reducción sin pérdidas propuesta permite obtener información sobre los 119 ítems de mushroom, además de proporcionar información sobre las reglas de asociación negativas que contiene el dataset de clasificación, todo ello empleando menos recursos y tiempo que con el análisis del dataset de clasificación original.

3.2.2. Catálogos

En *KEEL Standard Dataset Repository* (2004) hay una selección de 75 datasets de clasificación, la mayoría de ellos procedentes de *UCI Machine Learning Repository* (2013). Antes de publicarlos, los investigadores responsables del repositorio realizaron una transformación de los datasets originales para adaptar su formato a la herramienta con que llevaban a cabo sus investigaciones, publicada bajo licencia GPLv3 en http://sci2s.ugr.es/keel/download.php.

La transformación aplicada a los datasets de clasificación de UCI consiste en:

1. añadir metadatos al dataset de clasificación con el nombre de los atributos, su tipo –categórico, entero o real– y su rango,
2. suprimir los registros con datos desconocidos,
3. situar la clase en la última columna, y
4. en algún caso, eliminar los registros duplicados.

La segunda transformación tienen consecuencias en el resultado final del análisis. En clasificación, es habitual diferenciar los algoritmos que trabajan con datos desconocidos de los que no tratan esta información incompleta. Al aplicar algoritmos de ARM sobre datasets con datos desconocidos, como `mushroom` (FIMI), se obtienen resultados diferentes a los obtenidos al tratar el mismo dataset sin registros incompletos, como `mushroom` (KEEL).

La cuarta transformación no parece muy severa, ya que sólo la documentan en el dataset `abalone` y sólo se han eliminado 3 registros de un total de 4,177. Sin embargo, es una transformación utilizada en datasets de clasificación de diferentes repositorios, sin documentar, como si se tratara de una transformación aceptable sin necesidad de justificación alguna.

A continuación se profundiza en los efectos que pueden producir en el análisis cada una de las transformaciones.

1. La primera transformación es puramente descriptiva, los metadatos que contiene el dataset pueden usarse para que la aplicación pueda informar al analista sobre los descubrimientos hechos tras el análisis utilizando los nombres de los atributos, haciendo más comprensibles los resultados. También permiten saber si un atributo es numérico, y tomar decisiones de discretización en su caso.

   Los metadatos están en las primeras líneas del archivo, de texto plano, y comienzan con el carácter '@'. Si sólo se quieren leer los datos que contiene el dataset basta con ignorar estas primeras líneas.

2. La segunda transformación está motivada porque existen algoritmos que analizan datasets de clasificación con datos desconocidos y otros que no. En los datasets de UCI, cuando no se conoce el valor de un atributo en un registro se indica usando el carácter '?'.

   Cuando se preparó el dataset de clasificación `mushroom` (FIMI), convirtiendo los datos originales en códigos numéricos para que pudieran tratarse como transacciones, se codificaron también los datos desconocidos. Si el analista no conoce esta transformación puede obtener, por ejemplo, la regla de asociación $112 \rightarrow 2$, que podría interpretarse como la regla de clasificación «*Si* el atributo `cap-surface` toma el valor '?', *entonces* la seta es `comestible`». Si se hiciera caso a

esta regla, antes de observar una seta en el trabajo de campo ya sabríamos que es comestible, debido a que desconocemos el valor de ese atributo.

Una regla de clasificación con datos desconocido en su antecedente no aporta conocimiento, más bien puede confundir al analista. Por este motivo, en el análisis de datasets de clasificación con datos desconocidos se ha de aplicar alguna transformación que evite la situación planteada en el párrafo anterior.

- Suprimir el registro. Si no se está bien preparado para interpretar la información descubierta es mejor no descubrirla, centrando el análisis en el resto de información que contiene el dataset de clasificación.
- Sustituir el valor desconocido por el valor de mayor soporte en el atributo en cuestión, la moda.
- Sustituir el valor desconocido por una estimación basada en el resto de atributos. Esto supondría realizar otro análisis de clasificación en que intervengan sólo los atributos del análisis original, y el atributo con datos desconocidos actúa como clase.

Los tres tipos de transformación son agresivos, cambian la información que contiene el dataset de clasificación original. En esta investigación se adopta la primera estrategia, para poder utilizar directamente los datasets de clasificación publicados en KEEL. En la sección 5.2.4 se propone un post-análisis con el que se puede incorporar la información eliminada una vez analizado el dataset sin datos desconocidos.

3. Es fundamental conocer la posición que ocupa la clase en cada registro antes de ejecutar cualquier algoritmo sobre el dataset de clasificación. La clase es otro atributo más del problema, pero todas las reglas de clasificación generadas tienen a la clase como consecuente, por lo que si no se sabe cuál es su posición se podrían obtener reglas de clasificación con la clase en el antecedente y otro atributo del problema en el consecuente.

   Los datasets de clasificación de UCI suelen situarla en la primera primera posición, un estándar de facto en las registros para clasifi-

cación, aunque no lo hacen así en todos los datasets que publican. En KEEL han hecho el esfuerzo de transformar todos los datasets de clasificación que publican para que tengan la clase en la misma posición.

4. La eliminación de registros duplicados modifica el soporte de los ítems eliminados. Esto modifica la validez de los resultados obtenidos mediante algoritmos de ARM, de naturaleza probabilística y con una base justificada en el concepto de soporte,

Todos los datasets de clasificación de KEEL se pueden guardar en memoria RAM utilizando matrices, con el objetivo de hacer sólo una lectura de los datos en disco. Al utilizar la estructura `std::set` de la librería *C++ Standard Template Library (STL)* 2011, se descubre que casi la mitad de los datasets del repositorio no contienen duplicados.Los datasets de clasificación tienen más propiedades de las que se han utilizado para desarrollar la investigación de la sección 3.2.1.

Mediante un exhaustivo análisis sobre los 75 datasets de KEEL se descubre que todos los datasets de clasificación se pueden reducir eliminando los duplicados que contienen, y que al hacer esto se ha de cambiar el modo de analizarlos, ya que se elimina la información que el dataset original tiene sobre el soporte de cada uno de sus ítems y de las reglas de clasificación que contiene. El conjunto de registros distintos que contiene un dataset de clasificación es un *catálogo*, con propiedades que permiten un análisis eficiente de las reglas de asociación que contiene.

El resto de esta memoria formaliza una nueva metodología para analizar los datasets de clasificación a partir de los catálogos.

4

Catálogos

El dataset de clasificación `mushroom` (FIMI), utilizado en la sección 3.2.1, tiene una curiosa historia. En primer lugar, hay cinco versiones del dataset, procedentes todas de la misma fuente.

Jeff Schlimmer preparó un dataset de clasificación a partir de una guía sobre setas de América del norte, "*The Audubon Society Field Guide to North American Mushrooms* (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf". El 27 de abril de 1987, donó una copia a *UCI Machine Learning Repository* (2013), codificando los datos del dataset original para que ocupara menos espacio en disco y fuera más fácil su distribución online, detalle muy importante en esa época. Entregó el archivo `mushroom` (UCI)[1], con la siguiente descripción

> *This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500–525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poisonous Oak and Ivy.*

[1] `http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data` En el `DVD` adjunto es el archivo `/datos/UCI/mushroom.dat`.

Este dataset aparece en gran cantidad de publicaciones[2], ha servido para comparar la eficiencia de muchas propuestas sobre clasificación. Y. Wang y Wong (2003), Suzuki (2004), Thabtah et al., 2006, H. Li y H. Chen, 2009, H. Li y N. Zhang, 2010, Malik y Raheja, 2013 y Ritu, 2014 usan `mushroom` (UCI).

El Workshop on Frequent Itemset Mining Implementations (FIMI'03)[3] utilizó una versión codificada de `mushroom` (UCI), preparada para que los participantes en el workshop la usaran con las implementaciones de algoritmos de minería de itemsets frecuentes. Se trata del archivo `mushroom` (FIMI)[4].

En la introducción de las actas de FIMI'03[5] se destaca que es un dataset denso, lo que provoca que se aborte la ejecución de los programas que intentan analizarlo con soporte mínimo bajo. Las pruebas realizadas en el workshop con este dataset de clasificación se ajustan con soportes mínimos entre el 4 % y el 20 %. Borgelt (2004), Luo y X.-M. Zhang, 2007, Jin et al., 2009 y Sahoo et al., 2014 usan `mushroom` (FIMI).

Ambos datasets de clasificación, `mushroom` (UCI) y `mushroom` (FIMI), tienen 8,124 registros. El de UCI utiliza el carácter '?' para indicar que no se conoce el valor de un atributo en un registro. En la descripción del dataset se aclara este dato, indicando que contiene 2,480 valores desconocidos, todos asociados al atributo `stalk-root`. El de FIMI no indica nada al respecto. Supóngase que el valor codificado con el número 70 es realmente un dato desconocido en UCI.

- El análisis del dataset de FIMI puede generar resultados del tipo "*Si* (`stalk-root = 70`) y..., *entonces* la seta es `comestible`". En este caso, el analista no sabe qué valores son desconocidos y no es consciente de que se está generando una regla de clasificación incorrecta.

- En el análisis del dataset de UCI, el analista entiende que el resultado "*Si* (`stalk-root = ?`) y..., *entonces* la seta es `comestible`" significa que "*Si* no conocemos el valor del atributo `stalk-root` y..., *entonces*

[2] En `http://archive.ics.uci.edu/ml/datasets/Mushroom` hay un listado incompleto de artículos científicos que utilizan este dataset.

[3] `http://fimi.ua.ac.be/fimi03/`

[4] `http://fimi.ua.ac.be/data/mushroom.dat` En el `DVD` adjunto es el archivo `/datos/FIMI/mushroom.dat`.

[5] `http://www.ceur-ws.org/Vol-90/`

la seta es comestible". Y sabe que no es una regla de clasificación válida.

KEEL Standard Dataset Repository (2004) incorpora este dataset a su repositorio, indicando que tiene 8,124 registros, 5,644 sin datos desconocidos. En este repositorio no se utilizan los registros con datos desconocidos, por lo que el dataset de clasificación mushroom (KEEL)[6] tiene 5,644 registros.

LUCS-KDD discretised/normalised ARM and CARM Data Library (2003) publica el dataset de clasificación mushroom (LUCS-KDD), una versión discretizada y preparada para ARM de mushroom (UCI). Y. J. Wang et al. (2008), Hernández León, Carrasco et al., 2010 y Hernández León, 2011 usan mushroom (LUCS-KDD).

El 25 de junio de 1990, Jeff Schlimmer envió a UCI el dataset de clasificación original, junto al siguiente mensaje:

```
From: Jeff.Schlimmercs.cmu.edu
Date: Mon, 25 Jun 90 15:24:07 EDT

David, here is the mushroom file with the longer names.
I regenerated it from the original files that I fed to
Stagger, and I don't know where there are more records
than are in the 1-char version.  (8416 versus 8124.
Sigh.)
```

En el mensaje se muestra extrañado porque el dataset original tiene 292 registros más que el primero que se publicó en UCI. No le da mayor importancia, pero se despide con un suspiro.

La explicación de la desaparición de esos 292 registros está en el preproceso que se dio al dataset, que tenía 8,416 registros como este:

```
EDIBLE,CONVEX,SMOOTH,WHITE,BRUISES,ALMOND,FREE,CROWDED. . .
```

Para reducir sus dimensiones se acortaron los nombres usados en la descripción de cada valor. EDIBLE se convirtió en e, CONVEX en x... El registro anterior se convirtió en:

```
e,x,s,w,t,a,f,w. . .
```

[6] http://sci2s.ugr.es/keel/dataset.php?cod=178 En el DVD adjunto es el archivo /datos/KEEL/mushroom.dat.

Y además se eliminaron los registros duplicados. No se ha documentado el proceso, ninguno de los investigadores involucrados dan importancia a esta situación, pero el hecho constatado es que se partió de un dataset de clasificación que tenía 292 registros duplicados y se obtuvo un dataset sin ningún registro duplicado.

En el primer dataset de KEEL, (*KEEL abalone* 1995), se había hecho la misma operación partiendo del dataset de UCI (*UCI abalone* 1995). Esta transformación sí que está documentada, aunque no justificada:

> *In this version, 3 duplicated instances have been removed from the original UCI dataset.*

Efectivamente, el dataset de clasificación publicado en UCI tiene 4,177 registros, 3 de ellos duplicados, por lo que el dataset de KEEL tiene 4,174 registros diferentes.

Hay 37 de los 75 datasets de clasificación de KEEL sin registros duplicados. Muchos de los datasets de UCI tampoco tienen duplicados, como se ha comentado en el caso de `mushroom`. No he encontrado justificación científica para esta transformación en todo el estado del arte revisado sobre clasificación. Como estadístico no conozco ninguna justificación para llevar a cabo esta acción. Al eliminar registros duplicados se cambia la información que contiene el soporte de las reglas de asociación obtenidas. El soporte es la métrica más utilizada en los algoritmos de minería de reglas de asociación, directa o indirectamente es la base de todas las métricas usadas en esta disciplina.

La minería de reglas de asociación está fundamentada en la teoría de la probabilidad. Cuando se usa sobre los datos obtenidos en una muestra representativa de la población en estudio, debido a las muchas y bien estudiadas propiedades de estas colecciones de datos, puede extenderse a la población estudiada el conocimiento adquirido sobre la muestra analizada. Los soportes, en una muestra representativa, son buenos estimadores de las frecuencias poblacionales a las que representan. Pero si se suprimen registros de una muestra sólo por estar duplicados, la minería de reglas de asociación no tiene fundamento probabilístico.

Llamando *catálogo* al dataset obtenido tras eliminar los duplicados de un dataset de clasificación, se puede plantear la siguiente hipótesis.

> *Si se suprimen los duplicados de un dataset de clasificación se obtiene un catálogo, con la misma información estructural que la muestra*

contenida en el dataset original, pero sin información probabilística sobre las frecuencias poblacionales de los datos que contiene.

Si se han eliminado registros de una muestra y no se tiene información sobre qué registros han sido eliminados, es posible que un ítem con mucho soporte en el dataset de clasificación original, con mucha información para los algoritmos de ARM, pase a tener soporte inferior al fijado como mínimo, convirtiéndose en un ítem raro que no será analizado. Este razonamiento permite plantear la siguiente hipótesis.

El análisis de catálogos no se puede basar en el concepto de soporte. Las reglas de asociación que contiene un catálogo sólo se pueden interpretar expresando que se ha descubierto la relación entre dos valores de un dataset, no informan sobre si esta relación es o no frecuente.

Para obtener un catálogo a partir de un dataset de clasificación sólo hay que eliminar los duplicados que contiene. Es un preproceso trivial, por lo que se aplicó sobre los 75 datasets de KEEL para conocer mejor las características de este nuevo tipo de colecciones de datos.

En la sección 4.1 se explica, mediante un ejemplo, por qué no se debe usar el soporte para medir la calidad de las reglas de clasificación obtenidas en el análisis de un catálogo. En la sección 4.2 se presenta una descripción de algunas características de los datasets de clasificación, al saber que muchos de ellos son realmente catálogos, se comprueba que su análisis directo mediante metodologías de minería de reglas de asociación no es aconsejable.

Al analizar las reglas de asociación que contienen los catálogos, y las características que los diferencian de otros tipos de datasets, surgen nuevos conceptos que permiten descubrir y formular un nuevo modelo matemático con el que analizar un catálogo. En la sección 5.1 se presentan estos conceptos teóricos. En la sección 5.2 se presenta una metodología con la que aplicar el modelo teórico propuesto.

En el capítulo 6 se muestran los resultados obtenidos al aplicar esta metodología sobre los 75 catálogos que se obtienen a partir de todos los datasets de clasificación de KEEL.

4.1. Significado del soporte en catálogos

Supóngase que en una región hay 101 variedades de setas con características bien definidas, una variedad es venenosa y las 100 restantes son comestibles. Las variedades comestibles tienen características similares, mientras que la venenosa se diferencia del resto en muchas características. Supóngase que la población de setas tiene 1,000,000 de ejemplares de la variedad venenosa, mientras que de cada variedad de seta comestible hay 10,000 ejemplares. En estas circunstancias, el 50 % de las setas de la población son venenosas y el otro 50 % son comestibles. Cada variedad de setas comestibles tiene una frecuencia poblacional del 0.5 %.

Si se toma una *muestra representativa* de ejemplares de esta población, es de esperar que la proporción de setas venenosas recogida sea próxima al 50 %. El soporte de cada una de las variedades de seta comestibles será próximo al 0.5 %. Si se aplica un algoritmo de ARM fijando el soporte mínimo en el 0.5 %, la variedad venenosa será considerada *frecuente*, mientras que algunas variedades de seta comestibles se considerarán *frecuentes* y el resto *raras*, en función de que el azar haya proporcionado un soporte superior o inferior a su frecuencia poblacional.

Si se eliminan los duplicados de la muestra obtenida, el catálogo resultante tendrá exactamente 101 registros, 100 representando a setas comestibles y sólo una venenosa. Si se fija el soporte mínimo en el 1 %, la variedad venenosa será considerada como *rara* porque no comparte sus características con el resto de variedades. No aparecerá en ninguna de las reglas de clasificación encontradas. Las variedades comestibles, con muchas características comunes, proporcionarán reglas frecuentes. Las conclusiones del análisis afirmarán que todas las setas estudiadas son comestibles, algo que es totalmente incorrecto, y peligroso en el ejemplo planteado.

Si se eliminan los registros duplicados de un dataset de clasificación, el soporte de los ítems del catálogo obtenido no son buenos estimadores de su frecuencia poblacional. No se puede utilizar esta métrica para decidir la importancia que tiene un dato concreto y, mucho menos, para eliminar los datos que estén poco representados en el catálogo. Los conceptos de ítem frecuente o de ítem raro no tienen sentido en el análisis de este tipo de datasets.

4.2. Datasets de clasificación

Un dataset de clasificación, $\mathcal{D}_0$, es una colección de datos recogidos sobre individuos ya clasificados para un experimento de clasificación. Los datos usados en un experimento de clasificación se suelen almacenar con formato de matriz en archivos de texto plano. Son los formatos que usan *UCI Machine Learning Repository* (2013) y *KEEL Standard Dataset Repository* (2004), en algunos casos incluyendo metadatos con información sobre los atributos y las clases (nombre, tipo, rango…).

Los datasets de clasificación son matrices de $\mathcal{N}$ +1 columnas y $\mathcal{M}_0$ filas, siendo N el número de atributos del experimento y $\mathcal{M}_0$ el número de registros recogidos en el dataset. Cuando se toma una muestra de individuos clasificados, $\mathcal{M}_0$ es el número de individuos clasificados muestreados, son *evidencias*. Es posible que dos individuos compartan las mismas características, e incluso que ambos pertenezcan a la misma clase. En este último caso habrá una evidencia duplicada en $\mathcal{D}_0$. Un modo de reducir las dimensiones de $\mathcal{D}_0$ es la eliminación de duplicados, al fin y al cabo es información que ya se conoce. El problema surge cuando se quiere usar el soporte en estos datasets de clasificación reducidos.

El uso de soporte mínimo fue el primer recurso de los algoritmos de ARM para poder analizar grandes datasets sin obtener problemas de desbordamiento de memoria que abortaran el proceso. En datasets de transacciones con un gran número de ítems distintos –100 ya son muchos ítems si están muy relacionados entre sí– no se puede gestionar correctamente tanta relación inter-ítem, por lo que se han de reducir las dimensiones del problema.

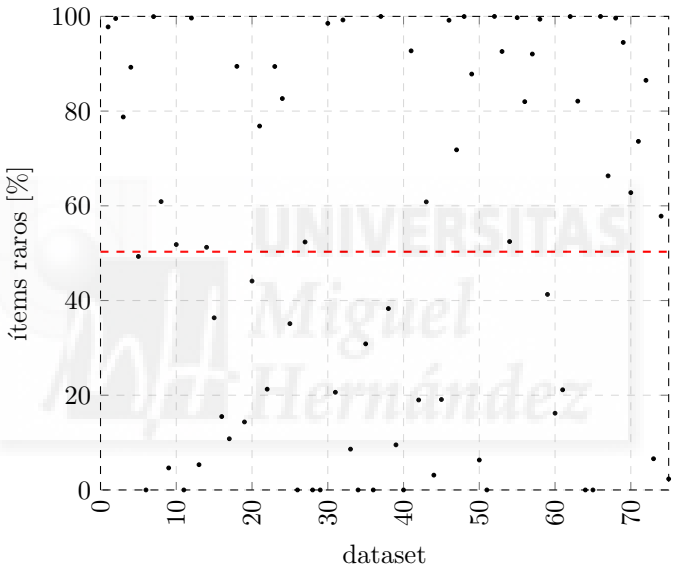
Coenen y Leng, 2007 hacen un estudio sobre el efecto que producen los umbrales de soporte y confianza sobre la precisión de los clasificador obtenidos mediante reglas de clasificación. Utilizan un soporte estándar del 1 %, argumentando que produce buenos clasificadores en general, pero en casos como `mushroom` (UCI) tienen que usar un valor superior, 1.8 %, para conseguir buenos resultados.

En la figura 4.1 se muestra el porcentaje de ítems raros que contienen los 75 datasets de clasificación de KEEL cuando se fija el soporte mínimo en el 1 %.

- En 40 de los 75 datasets, más de la mitad de sus ítems son raros.

- Hay 15 datasets de clasificación en los que casi el 100 % de sus ítems son raros. Los resultados obtenidos con este umbral de soporte ofrecen información sobre un porcentaje muy pequeño de los ítems en estudio, ya que la mayoría de ítems son ignorados en estos casos.

Figura 4.1: Porcentaje de ítems con soporte inferior a $minSup = 1\,\%$ –ítems raros– contenidos en los 75 datasets de KEEL, en orden alfabético



Cabe destacar que en un proceso no supervisado de este tipo, no se obtiene información detallada sobre los ítems que han sido ignorados en el análisis. Con el único propósito de poder llevar a cabo el análisis, se ha perdido el control sobre los datos que serán realmente analizados.

Los ítems raros, en el problema de clasificación planteado mediante técnicas de ARM, son los valores que toma un atributo en un individuo concreto. Ignorar uno de los valores de un atributo puede ser algo muy grave si el atributo sólo toma dos valores. Si el atributo es numérico y tiene un rango más extenso, la pérdida de información es menos grave, pero sigue siendo pérdida de información.

En las siguientes secciones se describen tres circunstancias que afectan a los datasets de clasificación cuando son analizados con técnicas de minería de reglas de asociación. Estos aspectos se han descubierto a

Al hacer un análisis descriptivo de los 75 datasets de clasificación publicados en KEEL, se han descubierto tres aspectos que sugieren un estudio diferente al ARM cuando se quiere obtener información sobre ciertos datasets, concretamente sobre los catálogos.

1. En 17 de los 75 datasets de clasificación analizados se pierden todos los ítems de algún atributo. Se trata de atributos continuos con un rango muy grande, en los que se usa mucha precisión para medir los valores de los atributos, lo que provoca que apenas se repitan los valores observados en ese atributo. El efecto de eliminar un atributo completo es bueno para el análisis ya que este tipo de atributos no se puede gestionar correctamente con técnicas de ARM como si fuera un atributo categórico, sin embargo el analista desconoce qué atributo ha desaparecido.

2. Hay 12 datasets de clasificación con clases poco frecuentes, con soporte inferior al 1 %, lo que provoca que en los registros etiquetados con estas clases desaparezca la etiqueta de clase, quedando un registro sin clasificar que no aporta absolutamente nada al experimento de clasificación. De esto tampoco es informado el analista, y los registros sin clasificar permanecen en el análisis consumiendo recursos de memoria que no darán ningún tipo de conocimiento al análisis.

3. Un aspecto fundamental al analizar datasets de clasificación utilizando técnicas de ARM es la presencia de atributos constantes.

   Las reglas de asociación (ARs) se crearon para descubrir las relaciones existentes entre los ítems de un gran conjunto de transacciones. Modelizan con gran acierto la llamada *cesta de la compra*. Debido al gran número de ARs que puede contener un conjunto de transacciones, no es operativo obtenerlas todas, y en muchas ocasiones no es posible debido al desbordamiento de recursos de los equipos utilizados para el análisis.

   La medida más importante en los algoritmos de ARM es el soporte de las reglas descubiertas. Todas las medidas de calidad de las ARs

utilizan de algún modo el soporte de una regla. Cuanto más soporte tenga una regla mayor será la información que puede proporcionar al investigador.

Es difícil pensar que en una enorme *cesta de la compra* exista un ítem presente en todas sus transacciones. Por ejemplo, hay mucha gente que compra pan a diario, pero no todos los clientes lo hacen, o incluso un cliente que ha comprado pan por la mañana no vuelve a hacerlo aunque acuda de nuevo al comercio el mismo día. De hecho, en ninguno de los trabajos mencionados en los capítulos 2 y 3 se menciona esta circunstancia.

Sin embargo, los datasets de clasificación son susceptibles de contener atributos constantes, atributos sobre los que sólo se recoge un valor en la muestra observada, aunque teóricamente pueda tomar más valores. En la sección 6.1 se muestra que no es tan extraño encontrar atributos constantes en un dataset de clasificación.

- Al recoger datos en una población en que sólo existen individuos de raza caucásica, sólo aparecerá un valor en el atributo *raza*.
- Si se dispone de un gran dataset de clasificación y se quieren analizar los datos de un cierto periodo de tiempo, es posible que algún atributo con datos sobre el clima quede reducido a un único valor.

Los ítems de un atributo constante aparecen en todas las líneas de un dataset de clasificación, es decir, tienen un soporte del 100 %. Si utilizamos este criterio para decidir qué ítem contiene más información para el estudio, pensaremos que este ítem es el más importante del dataset. Y no lo es en un experimento de clasificación, no tiene sentido que una regla de asociación incluya la información "*Si* $A_i = v_i \Rightarrow \ldots$", cuando no tenemos evidencias de que A_i pueda tomar un valor distinto a v_i.

En la sección 6.2 se muestra el efecto de no tener en cuenta este aspecto característico de los datasets de clasificación y atípico en los conjuntos de transacciones, para los que se diseñaron originalmente los algoritmos de ARM.

5

Modelización

En este capítulo se formaliza el concepto de *catálogo*. El estado del arte revisado en la sección 3.2 muestra que los datasets de clasificación se analizan siempre desde una perspectiva probabilística. En el capítulo 4 se demuestra que existen dos tipos bien diferenciados de datasets de clasificación:

1. Muestras representativas
2. Catálogos

Las muestras representativas poseen unas propiedades estadísticas que permiten su análisis mediante técnicas de minería de reglas de asociación. Todas las investigaciones realizadas sobre minería de reglas de clasificación asociativa repiten la misma conclusión: los algoritmos que se proponen en este área producen clasificadores más precisos que los obtenidos con técnicas de minería de reglas de clasificación. Sin embargo, en todas estas investigaciones se utilizan indistintamente muestras representativas y catálogos.

Al usar un enfoque probabilístico para analizar catálogos, se acepta la creencia de que los ítems con soporte menor que un valor prefijado son «raros», son poco representativos en la población en estudio. En la sección 4.1 se explica, mediante un ejemplo, que esta creencia no está científicamente justificada. En este capítulo se definen conceptos y estrategias que permiten analizar un catálogo sin utilizar el concepto de soporte.

En la sección 5.1 se propone un modelo teórico en el que se definen los elementos de un catálogo y sus propiedades. A partir de estos conceptos, se postulan lemas y teoremas que permiten tratar a los datasets de clasificación utilizando operaciones básicas de teoría de conjuntos. Estas operaciones proporcionan un conjunto de datasets de clasificación reducidos, con la misma información sobre clasificación que el dataset original.

En la sección 5.2 se utiliza este modelo teórico y se presentan funciones y estructuras que permiten construir el algoritmo $\mathcal{ACDC}$. Este algoritmo trata los datasets de clasificación como elementos individuales, no analiza todo su contenido separando los datos que contiene, por lo que los recursos de memoria que utiliza son los necesarios para alojar una matriz en memoria. En los algoritmos de minería de reglas de asociación se produce desbordamiento de memoria cuando se buscan las relaciones que existen entre los datos de una matriz, debido a la gran cantidad de datos que hay que relacionar y el número exponencial de relaciones que se deben almacenar para ser evaluadas.

5.1. Modelo teórico

Partiendo de las definiciones de la sección 2.2, en esta sección se presentan nuevas definiciones que permiten estudiar las propiedades de un catálogo y definir formalmente el concepto de *colección de catálogos robustos* (def. 5.18), un conjunto de catálogos de diferentes dimensiones que contienen la misma información para clasificación que el catálogo inicial.

Sea $\mathcal{C} = \{c_1, \ldots, c_Q\}$ una partición de la población en estudio. Sean $\mathcal{A} = \{A_1, \ldots, A_\mathcal{N}\}$ un conjunto de $\mathcal{N}$ atributos medibles en los individuos de la población. Sea $\mathcal{D}_0$ el dataset de clasificación que contiene los $\mathcal{M}_0$ registros recogidos para el experimento de clasificación, $|\mathcal{D}_0| = \mathcal{M}_0$. Si se separan de $\mathcal{D}_0$:

1. Los registros duplicados, dejando un representante de cada uno de ellos.

2. Los registros con datos desconocidos.

Se obtiene el catálogo $\mathcal{D}$, con $|\mathcal{D}| = \mathcal{M}$ registros, y dos matrices con los duplicados y los datos desconocidos. Estas dos matrices se incorporarán en un análisis posterior al análisis del catálogo obtenido.

El dataset de clasificación original se convierte en un dataset de clasificación de $\mathcal{M} \times (\mathcal{N}+1)$ elementos. Sólo se sabe que el soporte no es una métrica adecuada para analizar este dataset de clasificación..

Para desarrollar el modelo teórico con el que se pueden analizar estos datasets de clasificación, los catálogos, se definen a continuación los elementos que forman un catálogo.

Definición 5.1 (Caracterización)**.** *Una* caracterización x *es un conjunto ordenado de valores correspondientes al conjunto de atributos* $\mathcal{A} = \{A_1, \ldots, A_\mathcal{N}\}$ *de un experimento de clasificación* $\mathcal{E}$*.*

$$x = (A_1 = x_1, \ldots, A_\mathcal{N} = x_\mathcal{N}) = (x_1, \ldots, x_\mathcal{N}) \tag{5.1}$$

donde x_j *es un valor del rango de* A_j*.*

Definición 5.2 (Caracterización completa)**.** *Una caracterización es completa para el experimento de clasificación* $\mathcal{E}$ *cuando contiene el valor de los* $\mathcal{N}$ *atributos del experimento.*

Definición 5.3 (Caracterización incompleta)**.** *Una caracterización es incompleta para el experimento de clasificación* $\mathcal{E}$ *cuando se desconoce el valor de alguno de los* $\mathcal{N}$ *atributos del experimento.*

Una caracterización es el conjunto de valores observados en un individuo de la población en estudio. Al obtener la caracterización de un individuo ya clasificado se obtiene una evidencia empírica.

Definición 5.4 (Evidencia empírica)**.** *Una* evidencia empírica*,* e*, es el par formado por la caracterización* x *de un individuo de la población en estudio y la clase* c *a la que pertenece.*

$$e = (x, c) \tag{5.2}$$

La evidencias basadas en una caracterización completa son evidencias completas. Los registros de $\mathcal{D}_0$ con datos desconocidos son evidencias incompletas.

Definición 5.5 (Dataset de clasificación)**.** *Un* dataset de clasificación *es un conjunto de evidencias.*

Un dataset de clasificación es, por ejemplo, una muestra de individuos clasificados procedentes de una población. Si se eliminan de un dataset de clasificación todos los valores correspondientes a cualquier atributo, el conjunto obtenido también es un dataset de clasificación.

Definición 5.6 (Catálogo)**.** *Un* catálogo *es un dataset de clasificación sin valores desconocidos ni evidencias duplicadas.*

En un catálogo sólo se recoge una instancia de cada una de las evidencias completas del dataset.

Lema 5.1 (Existencia de catálogos en los datasets de clasificación)**.** *Todo dataset de clasificación con caracterizaciones completas contiene al menos un catálogo.*

Demostración. Sea $\mathcal{D}_?$ el conjunto de evidencias con datos desconocidos. Sea $\mathcal{D}_d$ el conjunto de evidencias duplicadas. Si se eliminan del dataset las evidencias de $\mathcal{D}_?$ y se deja sólo un representante de las evidencias de $\mathcal{D}_d$ se obtiene el conjunto $\mathcal{D}_0$, que es un catálogo.

$$dataset = \mathcal{D}_0 \cup \mathcal{D}_? \cup \mathcal{D}_d \tag{5.3}$$

□

Lema 5.2 (Interpretación del soporte de los ítems de un catálogo)**.** *Un catálogo $\mathcal{D}$ no contiene información sobre la frecuencia muestral de sus evidencias, por lo que el soporte no es un buen estimador de la frecuencia poblacional de un ítem, de un k-itemset o de una regla.*

Demostración. Supóngase que la población en estudio sólo tiene individuos con las caracterizaciones x e y, los primeros pertenecientes a la clase c y y los segundos a la clase c'. Sean X e Y el número de individuos de la población de las clases c y c' respectivamente, $X \neq Y$. Si se toma una muestra representativa de la población, es de esperar que el soporte de la evidencia (x, c) sea un buen estimador de la frecuencia poblacional de la clase c, $P(x) = \frac{X}{X+Y}$. Si se eliminan duplicados de la muestra se obtiene un catálogo con sólo dos evidencias, $e_1 = (x, c)$ y $e_2 = (y, c')$, ambas con soporte del 50 %. Estos soportes no pueden ser usados para estimar $P(x) = \frac{X}{X+Y}$ y $P(y) = \frac{Y}{X+Y}$ pues $X \neq Y \Rightarrow \frac{X}{X+Y} \neq \frac{Y}{X+Y}$. □

Definición 5.7 (Clasificador ingenuo)**.** *Se denotará con $\mathcal{C}_{\mathcal{D}}(x)$ el* clasificador ingenuo *de la caracterización x en el catálogo $\mathcal{D}$, el conjunto de clases asociado en $\mathcal{D}$ con la caracterización x.*

$$\mathcal{C}_{\mathcal{D}}(x) = \{c \in \mathcal{C} \ / (x, c) \in \mathcal{D}\} \tag{5.4}$$

Se trata de un clasificador *ingenuo* y sólo se puede usar para clasificar un nuevo individuo si:

1. se obtiene la caracterización completa del individuo, x, y
2. x está registrada en $\mathcal{D}$.

El conjunto de todos los clasificadores ingenuos de $\mathcal{D}$ es una partición bayesiana sin discretización, con un exceso de expresividad (Orallo et al., 2004). El conjunto de todos los clasificadores ingenuos de $\mathcal{D}$, $\{\mathcal{C}_{\mathcal{D}}(x) \ / x \in \mathcal{D}\}$, contiene la misma información que $\mathcal{D}$ sobre el experimento de clasificación. Si la caracterización x está relacionada en $\mathcal{D}$ con más de una clase, $\mathcal{C}_{\mathcal{D}}(x)$ ocupará menos espacio en memoria que $\mathcal{D}$ pues sólo contendrá una vez la caracterización x. Al no utilizar el soporte de las caracterizaciones de $\mathcal{D}$, este conjunto sólo proporciona la información de que "*existen individuos con la caracterización x*" –cuando $|\mathcal{C}_{\mathcal{D}}(x)| \geq 1$– y que "todos los individuos conocidos con la caracterización x están clasificados del mismo modo, cuando $|\mathcal{C}_{\mathcal{D}}(x)| = 1$. Daré nombre a estos conceptos.

Definición 5.8 (Caracterización desconocida)**.** *La caracterización x es* desconocida *cuando no está registrada en el catálogo $\mathcal{D}$.*

$$x \text{ es desconocida } \Leftrightarrow |\mathcal{C}_{\mathcal{D}}(x)| = 0 \tag{5.5}$$

Al iniciar un experimento de clasificación, todas las combinaciones posibles de los valores de cada atributo (características) son caracterizaciones desconocidas. Conforme se obtienen caracterizaciones completas de individuos clasificados, se descubre qué caracterizaciones existen en la población en estudio. En la DM sólo se usan los datos recogidos, desde la perspectiva de clasificación, las combinaciones de características que realmente existen.

En el caso hipotético de que se pudieran obtener las caracterizaciones de todos los individuos de una población, las caracterizaciones desconocidas son combinaciones de características que no existen, y pueden ser

interpretadas como "*la naturaleza de los atributos estudiados hace imposible esta combinación*". Como se suele trabajar con muestras y no con poblaciones completas, esta interpretación se ha de relajar: "*si nunca se ha encontrado esta caracterización, se puede estimar que la naturaleza de los atributos estudiados hace imposible esta combinación de características*". Como ocurre con cualquier estimación, la obtención de nuevos datos para el experimento podría modificarla.

Definición 5.9 (Caracterización robusta)**.** *La caracterización x es* robusta *cuando está relacionada con una única clase en el catálogo $\mathcal{D}$.*

$$x \text{ es robusta } \Leftrightarrow |\mathcal{C}_{\mathcal{D}}(x)| = 1 \tag{5.6}$$

Una *evidencia robusta* es aquella cuya caracterización es robusta. Un *clasificador robusto* es el que relaciona a una caracterización con exactamente una clase. Un catálogo puede contener muchas evidencias robustas, incluso puede contener sólo evidencias robustas si el experimento de clasificación está bien diseñado, i.e., si los atributos seleccionados contienen la información suficiente para discriminar la clase de pertenencia de todos los individuos de la población. Los catálogos robustos son el eje de esta investigación.

Definición 5.10 (Catálogo robusto)**.** *Un catálogo es robusto cuando todas sus caracterizaciones son robustas.*

$$\mathcal{D} \text{ es robusto } \Leftrightarrow |\mathcal{C}_{\mathcal{D}}(x)| = 1, \forall x \in \mathcal{D} \tag{5.7}$$

Las caracterizaciones robustas permiten clasificar a cualquier individuo de la población de forma unívoca. Mientras no se compruebe lo contrario, es decir, si no se demuestra que un individuo con la caracterización robusta x pertenece a otra clase, las caracterizaciones robustas son reglas de clasificación determinísticas:

> *Si (x, c) es una evidencia robusta,* sabemos *que todos los individuos con la caracterización x pertenecen a la clase c.*

Definición 5.11 (Caracterización con incertidumbre)**.** *La caracterización x contiene* incertidumbre *cuando está relacionada con más de una clase en el catálogo $\mathcal{D}$.*

$$x \text{ contiene incertidumbre } \Leftrightarrow |\mathcal{C}_{\mathcal{D}}(x)| > 1 \tag{5.8}$$

Aunque la incertidumbre es una característica de las caracterizaciones, el concepto puede trasladarse a catálogos, sin embargo no es aplicable a evidencias.

Definición 5.12 (Catálogo con incertidumbre)**.** *Un catálogo contiene incertidumbre cuando alguna de sus caracterizaciones tienen incertidumbre.*

$$\mathcal{D} \text{ contiene incertidumbre } \Leftrightarrow \exists x \in \mathcal{D} \,/\, |\mathcal{C}_{\mathcal{D}}(x)| > 1 \tag{5.9}$$

La incertidumbre refleja que los atributos seleccionados para el experimento de clasificación, o la precisión usada para medirlos, no discriminan unívocamente a la clase en estudio. Supóngase que el catálogo $\mathcal{D}$ contiene la caracterización $\{A_1 = 1.1\}$ asociada a las clases c y c'.

1. Se podría eliminar la incertidumbre añadiendo un atributo al experimento, A_2, si se obtuvieran las evidencias $(1.1, v_2^1, c)$ y $(1.1, v_2^2, c')$, siendo $v_2^1 \neq v_2^2$..
2. Si se pueden utilizar herramientas con mayor precisión para medir el atributo A_1, podría desaparecer la incertidumbre si se observan las evidencias robustas $(1.12, c)$ y $(1.14, c')$.

Sólo se puede eliminar la incertidumbre añadiendo nuevos atributos al experimento o usando mayor precisión al medir alguno(s) de ellos. La discretización usada en CARM, entonces, no reduce la incertidumbre de un experimento de clasificación, más bien puede acrecentarla. Cuando hay muchas evidencias asociando la caracterización x con la clase c, y pocas que la asocien con la clase c', se puede eliminar la incertidumbre de $\mathcal{D}$ eliminando las evidencias (x, c') del dataset de clasificación. Pero esto creará el falso descubrimiento de que la caracterización x es robusta. Y si el dataset analizado es realmente un catálogo, se estarán usando propiedades probabilísticas inexistentes.

Las caracterizaciones de un dataset son hechos observados. Las evidencias robustas de un dataset son "*hechos que, a partir de los datos conocidos, son incuestionables*". Lo que dicen los datos es que todas las caracterizaciones robustas conocidas están correctamente clasificadas (hasta la fecha). En palabras de ARM, una evidencia robusta es una regla de asociación con el 100 % de confianza. Si, antes de hacer ninguna transformación o eliminación de datos, se buscan las evidencias robustas de un dataset de

clasificación, se pueden conocer mejor las características estructurales del experimento de clasificación. Las siguientes definiciones ayudarán a hacerlo de forma eficiente.

Definición 5.13 (Conjunto de caracterizaciones)**.** *$\mathcal{D}^{-\mathcal{C}}$ representa el conjunto de caracterizaciones que contiene $\mathcal{D}$. Se obtiene eliminando de $\mathcal{D}$ toda la información sobre la clase, y eliminando las caracterizaciones duplicadas.*

$$\mathcal{D}^{-\mathcal{C}} = \{x \ / \ (x, c) \in \mathcal{D}\} \tag{5.10}$$

Las definiciones 5.9 y 5.13 permiten plantear el teorema más importante de este trabajo, con el que se averigua si un catálogo es robusto con sólo obtener su conjunto de caracterizaciones y comparar las dimensiones de ambas matrices.

Teorema 5.1 (Catálogo robusto)**.** *Sea $\mathcal{D}^{-\mathcal{C}}$ el conjunto de caracterizaciones del catálogo $\mathcal{D}$.*

$$\mathcal{D} \text{ es robusto } \Leftrightarrow \ |\mathcal{D}^{-\mathcal{C}}| = |\mathcal{D}| \tag{5.11}$$

($\Rightarrow$). Si $\mathcal{D}$ es robusto $\Rightarrow \ \forall x \in \mathcal{D}, |\mathcal{C}_{\mathcal{D}}(x)| = 1 \ \Rightarrow$ sean $e = (x, c), e' = (x', c')$ dos evidencias de $\mathcal{D}$, si $e \neq e' \Rightarrow x \neq x' \Rightarrow$ cada evidencia del catálogo $\mathcal{D}$ se corresponde con una caracterización de $\mathcal{D}^{-\mathcal{C}} \Rightarrow \ |\mathcal{D}^{-\mathcal{C}}| = |\mathcal{D}|$. □

($\Leftarrow$). Si $|\mathcal{D}^{-\mathcal{C}}| = |\mathcal{D}| \ \Rightarrow$ todas las evidencias de $\mathcal{D}$ se convierten en una caracterización de $\mathcal{D}^{-\mathcal{C}} \Rightarrow$ todas las evidencias de $\mathcal{D}$ son robustas. □

Corolario 5.1 (Catálogo con incertidumbre)**.** *$\mathcal{D}$ contiene incertidumbre si, y sólo si, $|\mathcal{D}^{-\mathcal{C}}| < |\mathcal{D}|$*

El lema 5.1 permite descomponer cualquier dataset de clasificación en tres conjuntos independientes, uno recogiendo todas las evidencias incompletas, otro recogiendo las evidencias duplicadas, y otro con un catálogo, el catálogo más grande que contiene el dataset de clasificación. El siguiente lema nos permitirá separar este catálogo en dos matrices, separando las evidencias robustas de las que tienen incertidumbre.

Lema 5.3 (Descomposición de datasets de clasificación)**.** *Todo dataset de clasificación puede descomponerse en cuatro conjuntos.*

$$dataset = \mathcal{D} \cup \mathcal{D}_{\mathcal{E}} \cup \mathcal{D}_{?} \cup \mathcal{D}_{d} \tag{5.12}$$

Donde $\mathcal{D}_{\mathcal{E}}$ contiene todas las caracterizaciones completas con incertidumbre y $\mathcal{D}$ es un catálogo robusto.

El lema 5.3 divide la información contenida en un dataset de clasificación en cuatro matrices que pueden ser tratadas de forma independiente.

1. $\mathcal{D}$ es un catálogo robusto. Estos catálogos tienen características que permiten un análisis profundo de todos sus datos sin presentar los desbordamientos de memoria producidos por los algoritmos de ARM.

2. $\mathcal{D}_{\mathcal{E}}$ contiene la incertidumbre del dataset de clasificación original. La incertidumbre no se puede eliminar sin realizar cambios en el diseño del experimento de clasificación. Si el dataset de clasificación original contiene incertidumbre, se separa para analizar un catálogo robusto. Una vez terminado el análisis, se debe comprobar si el conocimiento descubierto en el análisis del catálogo robusto se contradice con la información que contiene $\mathcal{D}_{\mathcal{E}}$.

3. $\mathcal{D}_?$ contiene las evidencias incompletas del dataset de clasificación original. Tras el análisis del catálogo robusto se debe comprobar si los datos que contiene contradicen o confirman el conocimiento descubierto.

4. $\mathcal{D}_d$ contiene información sobre el soporte de las evidencias del dataset de clasificación original. En este trabajo no se utiliza esta información. Sólo si se tuviera la certeza de que el dataset recoge una muestra representativa de la población en estudio, se podría incorporar al análisis para obtener información probabilística a la solución del problema.

$\mathcal{D}$ contiene todas las evidencias completas sin incertidumbre recogidas en el experimento de clasificación. Sea $\mathcal{M} = |\mathcal{D}|$, $\mathcal{D}$ es un conjunto de $\mathcal{M}$ clasificadores robustos. Un clasificador robusto es, desde la perspectiva de ARM, una regla de asociación con el 100 % de confianza.

Lema 5.4 (Confianza del clasificador ingenuo unitario)**.** *Si $\mathcal{C}_{\mathcal{D}}(x) = \{c\}$, la regla de asociación $\{x \to c\}$ tiene un 100 % de confianza.*

Demostración. Si $\mathcal{C}_{\mathcal{D}}(x) = \{c\}$, x no pertenece a $\mathcal{D}_{\mathcal{E}}$ ni a $\mathcal{D}_?$ por definición, si pertenece a $\mathcal{D}_d$ siempre aparece con la misma asociación $\{x \to c\}$, luego todas las apariciones de x en el dataset original forman la misma regla de asociación. Su confianza, aunque no se conozca el soporte, tiene en numerador y denominador el mismo valor, el número de veces que aparece x en el dataset, por lo que el cociente será siempre la unidad. □

En la sección 2.2.1 se revisa el estado del arte sobre minería de reglas de clasificación asociativa. Las conclusiones de todos los artículos revisados concluyen que las mejores reglas de clasificación asociativa contenidas en un dataset de clasificación proporcionan clasificadores de un modo más eficiente y preciso que las reglas de clasificación obtenidas mediante otros métodos de clasificación. Para cuantificar la bondad de las reglas de clasificación asociativa se utilizan las métricas soporte y confianza, y otras basadas en el soporte.

El lema 5.2 resta credibilidad al uso del soporte cuando se trabaja con catálogos, por lo que sólo se puede usar la confianza para determinar la calidad de las reglas obtenidas en un catálogo. Además, no se puede determinar la confianza de todas las reglas de asociación que contiene el catálogo si se desconoce su soporte. Pero se sabe que las evidencias robustas tienen el 100 % de confianza.

Hasta el momento sólo se ha descubierto que el catálogo $\mathcal{D}$ es un conjunto de $\mathcal{M}$ clasificadores con máxima confianza. No siempre se pueden medir todos los atributos en un individuo sin clasificar, por lo que no se pueden utilizar los $\mathcal{M}$ clasificadores descubiertos. Con operaciones elementales sobre $\mathcal{D}$ se puede descubrir si las evidencias incompletas que contiene también son robustas, para lo que es necesario definir nuevos conceptos.

En un experimento de clasificación $\mathcal{E}$ con $\mathcal{N}$ atributos, cualquier caracterización en la que no se conozca el valor de un atributo es una caracterización incompleta. Si no se conoce el valor del atributo A_i, $x = (x_1, \ldots, x_{i-1}, ?, x_{i+1}, \ldots, x_{\mathcal{N}})$ es una caracterización incompleta para el experimento $\mathcal{E}$.

Definición 5.14 (Experimento incompleto)**.** *Si se elimina el atributo A_i del experimento de clasificación $\mathcal{E}$, se obtiene $\mathcal{E}^{-i}$, otro experimento de clasificación. Si se elimina el conjunto de atributos $\mathcal{I} \subsetneq \mathcal{A}$ del experimento de clasificación $\mathcal{E}$, se obtiene $\mathcal{E}^{-\mathcal{I}}$, otro experimento de clasificación.*

La caracterización $x^{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{\mathcal{N}})$ es una caracterización completa del experimento $\mathcal{E}^{-i}$. La caracterización $x^{-\mathcal{I}}$, incompleta en el experimento de clasificación $\mathcal{E}$, es completa para el experimento $\mathcal{E}^{-\mathcal{I}}$.

Un experimento de clasificación $\mathcal{E}$ con $\mathcal{N}$ atributos contiene $(2^{\mathcal{N}} - 1)$ experimentos incompletos. Todos ellos contienen información sobre el problema de clasificación que se quiere resolver. Utilizando cualquier algorit-

mo de clasificación se obtienen clasificadores con caracterizaciones completas de alguno de esos sub-experimentos. El problema que se encuentran los investigadores de CARM es que hay demasiadas reglas de asociación en los datasets de clasificación con los que trabajan. Para descubrir las mejores reglas de clasificación asociativa han de obtener antes todas las reglas de asociación que contiene el dataset y, para ello, han de reducir los rangos de los atributos tratados y reducir el número de ítems a analizar usando el umbral de soporte mínimo o discretizando los atributos en estudio. Los algoritmos de ARM son muy eficientes, pero también consumen un exceso de recursos.

Si ya se ha descubierto que el catálogo $\mathcal{D}$ asociado al experimento $\mathcal{E}$ contiene $\mathcal{M}$ evidencias robustas, i.e., $\mathcal{M}$ reglas de asociación con confianza del 100 %, no se necesitan recursos para guardar esta información, basta con saber que $\mathcal{D}$ es un catálogo robusto. Los algoritmos de ARM comienzan obteniendo los ítems frecuentes, a continuación buscan los 2-itemsets frecuentes y siguen incrementando el número de elementos del itemset a tratar hasta que ya no quedan más en el dataset original, lo que ocurrirá cuando se encuentren los $\mathcal{N}$-itemsets frecuentes. Una vez se conocen todos los k-itemsets frecuentes que contiene el dataset, se calcula la confianza de las reglas que generan, siendo siempre la clase el consecuente, y se ordenan todas las reglas según alguna función de utilidad que siempre incorpora información referente al soporte o a la confianza de la regla.

La propuesta desarrollada en este capítulo comienza con esta última información, partiendo de los datos crudos, sin transformaciones ni supresión de dato alguno. Un catálogo robusto contiene $\mathcal{M}$ reglas de clasificación asociativa con un 100 % de confianza. No se puede añadir información al catálogo $\mathcal{D}$, pero sí estudiar los sub-catálogos que contiene, los catálogos asociados a los experimentos incompletos del experimento $\mathcal{E}$. Son necesarios nuevos conceptos para estudiar qué información contienen los sub-catálogos, y métodos que ayuden a descubrir esta información.

Definición 5.15 (Catálogo reducido)**.** *El catálogo reducido $\mathcal{D}^{-\mathcal{I}}$, en el experimento de clasificación $\mathcal{E}$, es el catálogo completo asociado al experimento incompleto $\mathcal{E}^{-\mathcal{I}}$. Para obtenerlo se ha de eliminar la información de los atributos de $\mathcal{I}$, y eliminar los duplicados que pudiera contener $\mathcal{D}^{-\mathcal{I}}$.*

El catálogo $\mathcal{D}^{-\mathcal{I}}$ representa al experimento $\mathcal{E}^{-\mathcal{I}}$, diseñado con el conjunto de atributos $\mathcal{A}^{-\mathcal{I}} = \mathcal{A} - \mathcal{I}$ y las clases $\mathcal{C}$. Una caracterización completa

de $\mathcal{D}^{-\mathcal{I}}$, $x^{-\mathcal{I}}$, representa a una caracterización incompleta de $\mathcal{D}$.

Cuando $\mathcal{I}$ sólo contiene un atributo, $\mathcal{I} = \{A_i\}$, el catálogo reducido $\mathcal{D}^{-\mathcal{I}} = \mathcal{D}^{-i}$ representa al experimento $\mathcal{E}^{-i}$, diseñado con los atributos $\mathcal{A}^{-i} = \mathcal{A} - \{A_i\}$ y las clases $\mathcal{C}$, con caracterizaciones completas x^{-i}. Esta es la base del algoritmo recursivo planteado en la sección 5.2.3, con el que se obtiene información sobre los $(2^{\mathcal{N}} - 1)$ sub-experimentos que contiene el experimento original $\mathcal{E}$.

Definición 5.16 (Atributo redundante)**.** A_i *es* redundante *cuando la información sobre clasificación que proporciona al experimento $\mathcal{E}$ está ya explicada por el resto de atributos.*

Si A_i es redundante, el conjunto de atributos $\mathcal{A}^{-i} = \mathcal{A} - \{A_i\}$ contiene la misma información sobre la clase $\mathcal{C}$ que el conjunto $\mathcal{A}$. Puede deberse a:

1. A_i no está relacionado con la clase $\mathcal{C}$.
2. A_i se puede obtener a partir $\mathcal{A}^{-i}$.

Sea cual sea el motivo, si A_i es un atributo redundante, al eliminarlo del experimento $\mathcal{E}$ se obtiene el experimento de clasificación $\mathcal{E}^{-i}$, con la misma información sobre la clase $\mathcal{C}$ en estudio que el experimento original. Pero con menos datos que procesar, datos que no aportaban nada al problema de clasificación.

Definición 5.17 (Atributo esencial)**.** A_i *es* esencial *cuando la información sobre clasificación que proporciona al experimento $\mathcal{E}$ no está explicada por el resto de atributos, $\mathcal{A}^{-i}$.*

Si A_i es un atributo esencial, el valor que toma en un individuo ayuda a discriminar la clase a la que pertenece. Si se elimina la información correspondiente a A_i del catálogo robusto $\mathcal{D}$, aparecerá incertidumbre en el catálogo obtenido.

Lema 5.5 (Caracterizacion de atributo)**.** *Sea $\mathcal{D}$ un catálogo robusto. Sea $\mathcal{D}^{-i}$ el catálogo obtenido al suprimir la información sobre el atributo A_i y eliminar duplicados. Entonces*

$$\text{Si } \mathcal{D}^{-i} \text{ es robusto } \Rightarrow \ A_i \text{ es redundante} \tag{5.13}$$

$$\text{Si } \mathcal{D}^{-i} \text{ no es robusto } \Rightarrow \ A_i \text{ es esencial} \tag{5.14}$$

Dem. 5.13. Si $\mathcal{D}^{-i}$ es robusto $\Rightarrow$ $|\mathcal{C}_{\mathcal{D}^{-i}}(x^{-i})| = 1\ \forall x^{-i} \in \mathcal{D}^{-i}$ $\Rightarrow$ al eliminar A_i no aparece incertidumbre $\Rightarrow$ $\mathcal{A}^{-i}$ contiene la misma información sobre $\mathcal{C}$ que $\mathcal{A}$ $\Rightarrow$ A_i es redundante. □

Dem. 5.14. Si $\mathcal{D}^{-i}$ no es robusto $\Rightarrow$ $\exists x^{-i} \in \mathcal{D}^{-i}$ / $|\mathcal{C}_{\mathcal{D}^{-i}}(x^{-i})| > 1$ $\Rightarrow$ Como $\mathcal{D}$ es robusto, $|\mathcal{C}_{\mathcal{D}}(x)| = 1\ \forall x \in \mathcal{D}$ $\Rightarrow$ $\exists x_i, x_i' \in \mathcal{A}_i \wedge x, x' \in \mathcal{D}$ / $x = (x^{-i}, x_i, c) \wedge x' = (x^{-i}, x_i', c')$, siendo $c \neq c'$ $\Rightarrow$ si se prescinde de la información que contiene A_i aparece incertidumbre en el experimento $\mathcal{E}^{-i}$ $\Rightarrow$ $\mathcal{A}_i$ es esencial . □

Para comprobar si un atributo es redundante o esencial, sólo son necesarias algunas operaciones elementales sobre matrices:

1. Eliminar la columna de $\mathcal{D}$ con los valores correspondientes al atributo A_i, obteniendo $\mathcal{D}_{temp}^{-i}$.
2. Eliminar de $\mathcal{D}_{temp}^{-i}$ las filas repetidas, obteniendo $\mathcal{D}^{-i}$.
3. Eliminar de $\mathcal{D}^{-i}$ la columna con los valores de la clase $\mathcal{C}$, obteniendo $\mathcal{D}_{temp}^{-i-\mathcal{C}}$.
4. Eliminar de $\mathcal{D}_{temp}^{-i-\mathcal{C}}$ las filas repetidas, obteniendo $\mathcal{D}^{-i-\mathcal{C}}$.
5. Si $|\mathcal{D}^{-i-\mathcal{C}}| < |\mathcal{D}^{-i}|$ se deduce que A_i es un atributo *redundante*, en otro caso, A_i es *esencial* (teorema 5.1).

Lema 5.6 (Herencia)**.** *Si $A_i \in \mathcal{A}$ es un atributo esencial en el experimento de clasificación $\mathcal{E}$, también lo es en todos los sub-experimentos $\mathcal{E}^{-\mathcal{I}}$ que contiene, $\forall \mathcal{I} \subsetneq \mathcal{A}$.*

Demostración. Como A_i es esencial en $\mathcal{E}$ $\Rightarrow$ $\exists x_i, x_i' \in A_i \wedge c, c' \in \mathcal{C} \wedge x, x' \in \mathcal{D}$ / $x = (x^{-i}, x_i, c), x' = (x^{-i}, x_i', c')$ $\wedge$ $c \neq c'$.
Sea $\mathcal{D}^{-\mathcal{I}}$, $\mathcal{I} \subsetneq \mathcal{A}$ / $i \notin \mathcal{I}$, un catálogo robusto $\Rightarrow$ $\exists x^{-\mathcal{I}}, x'^{-\mathcal{I}}$ / $x^{-\mathcal{I}} = (x^{-\mathcal{I}-i}, x_i, c)$ $\wedge$ $x'^{-\mathcal{I}} = (x^{-\mathcal{I}-i}, x_i', c')$ $\Rightarrow$ $\mathcal{C}_{\mathcal{D}^{-\mathcal{I}}}(x^{-\mathcal{I}}) = \{c, c'\}$ $\Rightarrow$ $\mathcal{D}^{-\mathcal{I}-i}$ no es robusto $\Rightarrow$ A_i es esencial para $\mathcal{D}^{-\mathcal{I}}$. □

El algoritmo $\mathcal{ACDC}$, expuesto en la sección 5.2.3, utiliza el teorema 5.1, el lema 5.3 y la definición 5.15 para encontrar todos los catálogos robustos que contiene el catálogo robusto $\mathcal{D}$.

Definición 5.18 (Colección de catálogos robustos)**.** *$\mathcal{CCR}_{\mathcal{D}}$ es la colección de catálogos robustos que contiene el catálogo robusto $\mathcal{D}$.*

$$\mathcal{CCR}_{\mathcal{D}} = \{\mathcal{D}^{-\mathcal{I}} \subset \mathcal{D} \;/\; \mathcal{D}^{-\mathcal{I}} \text{ es robusto}\} \tag{5.15}$$

Cuando el dataset a analizar es de grandes dimensiones, bien por tener atributos con rangos muy extensos o bien por tener muchos atributos, la obtención del $\mathcal{CCR}_{\mathcal{D}}$ puede requerir demasiado tiempo e incluso puede desbordarse la memoria RAM si contiene muchos catálogos robustos.

Si se obtienen sólo los atributos redundantes de $\mathcal{D}$, se descubre qué catálogos robustos contiene al eliminar sólo un atributo de $\mathcal{D}$, $\mathcal{CCR}_{\mathcal{D},1} = \{\mathcal{D}^{-i} \subset \mathcal{D} \;/\; \mathcal{D}^{-i} \text{ es robusto}\}$. El lema 5.6 permite ahorrar cálculos al obtener $\mathcal{CCR}_{\mathcal{D},2} = \{\mathcal{D}^{-i-j} \subset \mathcal{D} \;/\; \mathcal{D}^{-i-j} \text{ es robusto}\}$, ya que no será necesario comprobar de nuevo que los atributos esenciales del catálogo $\mathcal{D}$ siguen siendo esenciales en cualquier sub-catálogo $\mathcal{D}^{-\mathcal{I}}$. Aplicando el lema 5.6 se puede gestionar la memoria RAM para obtener recursivamente los conjuntos $\mathcal{CCR}_{\mathcal{D},k} = \{\mathcal{D}^{-\mathcal{I}} \subset \mathcal{D} \;/\; |\mathcal{I}| = k \;\wedge\; \mathcal{D}^{-\mathcal{I}} \text{ es robusto}\}$.

Lema 5.7 (Utilidad del $\mathcal{CCR}_{\mathcal{D}}$)**.** *Dado un experimento de clasificación $\mathcal{E}$ con el catálogo robusto asociado $\mathcal{D}$, todos los catálogos de su $\mathcal{CCR}_{\mathcal{D}}$ tienen la misma información que el catálogo $\mathcal{D}$ para clasificar sobre las clases $\mathcal{C}$.*

El lema 5.7 abre las puertas al investigador a decidir qué conjunto de atributos utilizar en su trabajo de campo, utilizando sólo los atributos que pueda medir, según las circunstancias.

En el lema 5.3 se ha descompuesto el dataset de clasificación original en cuatro conjuntos, el catálogo robusto $\mathcal{D}$, el conjunto de evidencias no robustas $\mathcal{D}_{\mathcal{E}}$, el conjunto de evidencias con datos desconocidos $\mathcal{D}_?$ y el conjunto de evidencias duplicadas $\mathcal{D}_d$.

1. La información que contiene $\mathcal{D}_d$ no es útil en el análisis propuesto. Puede ser incorporada si se sabe que el dataset es una muestra representativa de la población en estudio.

2. $\mathcal{D}$ permite obtener su $\mathcal{CCR}_{\mathcal{D}}$, que ha de ser contrastado con la información que contienen los conjuntos del dataset de clasificación aún no analizados.

3. $\mathcal{D}_?$ contiene evidencias incompletas. Cuando ya se conoce $\mathcal{CCR}_{\mathcal{D}}$ se puede comprobar si las sub-evidencias completas que contiene contradicen o refuerzan la información descubierta en $\mathcal{CCR}_{\mathcal{D}}$.

4. $\mathcal{D}_{\mathcal{E}}$ contiene las evidencias completas con incertidumbre de $\mathcal{E}$. Con ellas no se puede hacer nada, se deben al diseño del experimento y sólo informan de la imposibilidad de hacer una estimación exacta con este diseño. Sin embargo todas las sub-evidencias que contienen deben ser contrastadas con las evidencias robustas que hay en $\mathcal{CCR}_{\mathcal{D}}$.

Aunque en ARM no es habitual encontrar datasets en los que un ítem aparezca en todas sus transacciones, en clasificación se presenta esta circunstancia en muchas ocasiones. Por ejemplo, cuando se quiere estudiar un subconjunto de un dataset geolocalizado en una región en que un atributo siempre toma el mismo valor, o bien cuando se hace con un subconjunto perteneciente a un periodo de tiempo en el que todos los valores de un atributo son el mismo. Esta situación no está prevista en las investigaciones de CARM ya que no lo está en ARM, pero su efecto es muy negativo para la eficiencia de los algoritmos utilizados. En ARM se da mayor importancia a los ítems con mayor soporte, luego un atributo que sólo tome un valor en $\mathcal{D}$ tendrá la máxima importancia a pesar de no ofrecer ninguna información relevante para el análisis. La siguiente proposición ayudará a tenerlos en cuenta en el momento en que se tenga constancia de ello, tras la primera lectura del dataset de clasificación a procesar.

Proposición 5.1 (Atributos constantes)**.** *Si A_i sólo toma un valor en el catálogo $\mathcal{D}$, entonces A_i es redundante en todos los sub-experimentos del experimento de clasificación $\mathcal{E}$ y debe ser eliminado, pues no proporciona información para clasificación.*

El árbol $\mathcal{L}$ usado en el algoritmo Apriori es una estructura que aprovecha el orden lexicográfico de los ítems que contiene. Si se adapta este árbol para almacenar los índices de los atributos eliminados para obtener los catálogos robustos de $\mathcal{CCR}_{\mathcal{D}}$, contendrá nodos con subconjuntos de catálogos que ya se han analizado.

Propiedad 5.1 (Superconjuntos de catálogos robustos)**.** *Sean $\mathcal{I}, \mathcal{J} \subsetneq \mathcal{A}$, dos conjuntos de atributos tales que $\mathcal{I} = \mathcal{J} \cup \{\mathcal{A}_i\}$. Si $\mathcal{D}^{-\mathcal{I}}$ es catálogo robusto, entonces $\mathcal{D}^{-\mathcal{J}}$ también lo es.*

Demostración. Como $\mathcal{D}^{-\mathcal{I}}$ es robusto y $A_i \in \mathcal{I}$, si se añade a $\mathcal{D}^{-\mathcal{I}}$ el atributo A_i se está añadiendo un atributo que era redundante en $\mathcal{D}^{\mathcal{I}-\{A_i\}} = \mathcal{D}^{-\mathcal{J}}$. □

La propiedad 5.1 permite ahorrar cálculos que ya se han hecho de forma indirecta. Por ejemplo, si se ha descubierto que $\mathcal{I} = \{1, 2, 3, 4, 5\}$ proporciona el catálogo robusto $\mathcal{D}^{-\mathcal{I}}$, no es necesario comprobar que el catálogo $\mathcal{D}^{-\mathcal{J}}$, con $\mathcal{J} = \{2, 3, 4, 5\}$, también es robusto. Lo mismo ocurrirá con los catálogos $\mathcal{D}^{-2-3-4}$, $\mathcal{D}^{-2-3-5}$, $\mathcal{D}^{-2-3}$, ... Esta propiedad dota de mucha eficiencia al algoritmo $\mathcal{ACDC}$.

5.1.1. Análisis de datasets de clasificación

Esta tesis propone una metodología para analizar catálogos. El lema 5.3 permite dividir cualquier dataset de clasificación en cuatro conjuntos:

$$dataset = \mathcal{D} \cup \mathcal{D}_{\mathcal{E}} \cup \mathcal{D}_{?} \cup \mathcal{D}_{d}$$

El catálogo $\mathcal{D}$ proporciona el conjunto de catálogos robustos del dataset de clasificación, $\mathcal{CCR}_{\mathcal{D}}$, mientras que $\mathcal{D}_d$ sólo contiene información probabilística que no se utiliza en este trabajo. El objetivo de cualquier proceso de KDD es extraer el máximo de información de los datos conocidos. Si los conjuntos $\mathcal{D}_{\mathcal{E}}$ y $\mathcal{D}_{?}$ no están vacíos, para analizar el dataset de clasificación original se ha de contrastar la información que contienen estos conjuntos con la descubierta en $\mathcal{CCR}_{\mathcal{D}}$.

Análisis de $\mathcal{D}_{\mathcal{E}}$

El conjunto $\mathcal{D}_{\mathcal{E}}$ contiene las caracterizaciones completas con incertidumbre del experimento de clasificación. Ya se ha apuntado que la incertidumbre es inherente al diseño del experimento: 1) la selección del conjunto de atributos $\mathcal{A}$ no discrimina bien las clases en estudio, o 2) la precisión de las mediciones de algún atributo es insuficiente, cuando el atributo sí discrimina las clases $\mathcal{C}$.

Aunque no se trate de evidencias robustas, también son evidencias empíricas. Son caracterizaciones que no discriminan totalmente la clase de pertenencia, pero son datos que representan a la población en estudio. En un análisis basado en los datos conocidos no se puede ignorar esta información.

Una vez descubierto $\mathcal{CCR}_{\mathcal{D}}$, si $\mathcal{D}_{\mathcal{E}} \neq \emptyset$, se ha de comprobar qué subcaracterizaciones de las evidencias de $\mathcal{D}_{\mathcal{E}}$ aparecen como robustas en el sub-catálogo robusto correspondiente.

La metodología propuesta en esta tesis trabaja con catálogos completos, no con evidencias específicas, para evitar el dilema del ítem raro. Si se descubre que un sub-catálogo robusto de $\mathcal{D}$ contiene una evidencia no robusta y se elimina el sub-catálogo del conjunto $\mathcal{CCR}_{\mathcal{D}}$, se perdería la información sobre el resto de evidencias robustas descubiertas en el sub-catálogo.

Las sub-evidencias de $\mathcal{D}_{\mathcal{E}}$ que hayan sido descubiertas como robustas en el análisis de $\mathcal{D}$ se han de guardar como información adicional a la proporcionada por $\mathcal{CCR}_{\mathcal{D}}$. $\mathcal{D}_{\mathcal{E}}$ no tiene las propiedades de los catálogos robustos.

Lema 5.8 (Catálogos de $\mathcal{D}_{\mathcal{E}}$)**.** *Todos los sub-catálogos de $\mathcal{D}_{\mathcal{E}}$ contienen incertidumbre.*

Demostración. Todas las caracterizaciones de $\mathcal{D}_{\mathcal{E}}$ tienen incertidumbre, luego todas sus caracterizaciones tienen un clasificador $\mathcal{C}_{\mathcal{D}}(x)$ que relaciona a x con más de una clase, de lo que se deduce que cualquier sub-conjunto $x^{-\mathcal{I}}$ de x tiene un clasificador $\mathcal{C}_{\mathcal{D}}(x^{-\mathcal{I}})$ que lo relaciona con más de una clase. □

La incertidumbre recogida en $\mathcal{D}_{\mathcal{E}}$ no se puede "corregir", pero tampoco se debe ignorar. Supóngase que en uno de los catálogos robustos de $\mathcal{CCR}_{\mathcal{D}}$, $\mathcal{D}^{1,2}$, se ha registrado la evidencia (x_1, x_2, c). Supóngase que en $\mathcal{D}_{\mathcal{E}}$ está registrada la evidencia $(x_1, x_2, x_3, \{c, c'\})$. Existe una contradicción entre la información que contiene $\mathcal{CCR}_{\mathcal{D}}$ y la que contiene $\mathcal{D}_{\mathcal{E}}$. Al analizar este último conjunto se descubre que la caracterización (x_1, x_2) no es robusta, a pesar de formar parte del catálogo robusto $\mathcal{D}^{1,2}$.

No se debe eliminar la caracterización (x_1, x_2) de $\mathcal{D}^{1,2}$. La metodología propuesta en esta tesis se basa en el análisis de catálogos robustos completos. Un dataset de clasificación puede contener miles de millones de evidencias, como demuestran los resultados obtenidos en el capítulo 6. Sólo se guarda el conjunto de atributos que contiene cada catálogo robusto descubierto, y se puede reconstruir a partir del dataset de clasificación original sin ningún problema. Si se añade la caracterización con incertidumbre (x_1, x_2) al conjunto $\mathcal{D}_{\mathcal{E}}^{1,2}$, inicialmente vacío, queda registrada la excepción encontrada, que cuestiona la información descubierta en el catálogo robusto $\mathcal{D}^{1,2}$.

Para clasificar a un individuo con la caracterización (x_1, x_2) se utiliza la siguiente estrategia.

1. Se comprueba que hay un catálogo robusto que sólo usa los atributos A_1 y A_2, $\mathcal{D}^{1,2}$.
2. Se comprueba que la caracterización es robusta, $(x_1, x_2) \in \mathcal{D}^{1,2}$, y que está clasificada como c.
3. Se comprueba si $\mathcal{D}^{1,2}$ tiene excepciones, encontrando que $\mathcal{D}_{\mathcal{E}}^{1,2}$ contiene la caracterización (x_1, x_2).
4. Se concluye que no hay información con el 100 % de confianza para clasificar al individuo en estudio.

Como no se ha utilizado soporte, no se pueden dar estimaciones sobre qué probabilidad tiene el individuo de pertenecer a la clase c o a la clase c', sólo se puede informar que existe incertidumbre en este caso. Para dar estas estimaciones se ha de trabajar con una muestra representativa de la población en estudio, y contar el número de veces en que (x_1, x_2) está clasificado como c y cuántas como c'. Esto se puede hacer con una simple consulta al dataset de clasificación, que puede ejecutarse en tiempo real incluso en grandes colecciones de datos.

Definición 5.19 (Caracterizaciones con incertidumbre de $\mathcal{CCR}_{\mathcal{D}}$)**.** *$\mathcal{CC}_{\mathcal{D}_{\mathcal{E}}}$ es el conjunto de todas las caracterizaciones que contiene $\mathcal{D}_{\mathcal{E}}$ y contradicen la información descubierta en los catálogos robustos de $\mathcal{CCR}_{\mathcal{D}}$.*

Conocido $\mathcal{CCR}_{\mathcal{D}}$, sea $\mathcal{N}^-$ el número mínimo de atributos que contienen sus catálogos. Para obtener $\mathcal{CC}_{\mathcal{D}_{\mathcal{E}}}$ se han de extraer todas las caracterizaciones que contiene $\mathcal{D}_{\mathcal{E}}$ con, al menos, $\mathcal{N}^-$ atributos. Las que tengan menos atributos no se podrán comparar con la información de los catálogos robustos de $\mathcal{CCR}_{\mathcal{D}}$.

$\mathcal{CC}_{\mathcal{D}_{\mathcal{E}}}$ es, inicialmente, un conjunto vacío. Si los atributos del catálogo $\mathcal{D}_{\mathcal{E}}^{-\mathcal{I}}$ coinciden con los atributos de un catálogo robusto de $\mathcal{CCR}_{\mathcal{D}}$, se añade $\mathcal{D}_{\mathcal{E}}^{-\mathcal{I}}$ al conjunto $\mathcal{CC}_{\mathcal{D}_{\mathcal{E}}}$.

Una vez finalizado este proceso, para clasificar una caracterización se ha de buscar primero en los catálogos robustos de $\mathcal{CCR}_{\mathcal{D}}$. Una vez comprobado si se puede clasificar en base a $\mathcal{CCR}_{\mathcal{D}}$, se ha de buscar en $\mathcal{CC}_{\mathcal{D}_{\mathcal{E}}}$ por si hubieran evidencias con incertidumbre para esta caracterización.

$\mathcal{CCR}_{\mathcal{D}}$ y $\mathcal{CC}_{\mathcal{D}_\mathcal{E}}$ contienen toda la información que hemos descubierto en las evidencias completas del dataset de clasificación original. Ya sólo falta analizar el conjunto $\mathcal{D}_?$, si no es un conjunto vacío, para obtener toda la información que contiene el dataset de clasificación original.

Análisis de $\mathcal{D}_?$

Una vez descubierta la colección de catálogos robustos, $\mathcal{CCR}_{\mathcal{D}}$, y la incertidumbre de los catálogos de este conjunto, se ha de contrastar el conocimiento adquirido con la información que contiene $\mathcal{D}_?$. Para hacerlo, se han de estudiar todos los catálogos completos que contiene este conjunto.

Definición 5.20 (Mayor evidencia completa)**.** *Sea $\mathcal{I}$ el conjunto de valores desconocidos de la evidencia $e_? = (x_?, c)$, la mayor evidencia completa contenida en $e_?$ es $e_?^{-\mathcal{I}} = (x_?^{-\mathcal{I}}, c)$.*

Para contrastar la información que tiene $e_?$ se ha de comprobar si 1) $e_?^{-\mathcal{I}}$ es una evidencia robusta y 2) $\mathcal{D}^{-\mathcal{I}}$ está en $\mathcal{CCR}_{\mathcal{D}}$.

1. Si $|\mathcal{C}_{\mathcal{D}_?^{-\mathcal{I}}}(x_?^{-\mathcal{I}})| > 1 \Rightarrow x_?^{-\mathcal{I}}$ es una caracterización con incertidumbre.
2. Si $|\mathcal{C}_{\mathcal{D}_?^{-\mathcal{I}}}(x_?^{-\mathcal{I}})| = 1 \Rightarrow e_?^{-\mathcal{I}}$ es una evidencia robusta.

Si $x_?^{-\mathcal{I}}$ es una caracterización con incertidumbre se ha de aplicar el análisis propuesto en la sección 5.1.1. Si $e_?^{-\mathcal{I}}$ es una evidencia robusta, se puede buscar en $\mathcal{D}^{-\mathcal{I}}$ sólo si este catálogo es robusto, en otro caso no se dispone de información sobre las evidencias de $\mathcal{E}^{-\mathcal{I}}$ y se han de estudiar las sub-evidencias de $e_?^{-\mathcal{I}}$, $e_?^{-\mathcal{I}-\mathcal{J}}$, para todos los conjuntos $\mathcal{J} \subsetneq \mathcal{A} - \mathcal{I}$.

Se puede dividir $\mathcal{D}_?$ en dos conjuntos con información homogénea. $\mathcal{D}_{?,\mathcal{E}}$ contiene sus caracterizaciones con incertidumbre, y $\mathcal{D}_{?,+}$ sus evidencias robustas.

$$\mathcal{D}_? = \mathcal{D}_{?,\mathcal{E}} \cup \mathcal{D}_{?,+} \tag{5.16}$$

Definición 5.21 (Conjunto de caracterizaciones incompletas con incertidumbre de $\mathcal{D}_0$)**.** *$\mathcal{D}_{?,\mathcal{E}}$ es el conjunto de caracterizaciones incompletas con incertidumbre del dataset original.*

Definición 5.22 (Conjunto de evidencias incompletas robustas de $\mathcal{D}_0$)**.** *$\mathcal{D}_{?,+}$ es el conjunto de evidencias incompletas robustas del dataset original.*

El análisis de estos conjuntos completa el conocimiento descubierto a través de los catálogos robustos de $\mathcal{CCR}_{\mathcal{D}}$ y de los catálogos de $\mathcal{CC}_{\mathcal{D}_{\mathcal{E}}}$.

A continuación se desarrolla el modelo aplicado que se deriva de los conceptos teóricos expuestos en esta sección.

5.2. Modelo aplicado

En esta sección se desarrolla el algoritmo *Análisis de Caracterizaciones en Datasets de Clasificación* ($\mathcal{ACDC}$), que utiliza el modelo teórico descrito en la sección 5.1 para descubrir la colección de catálogos robustos que contiene el dataset de clasificación analizado.

Los datasets de clasificación son conjuntos de $\mathcal{M}_0$ evidencias, vectores de $\mathcal{N}+1$ datos. Se asume que la clase en estudio es el último valor de estos vectores, por lo que cada evidencia se puede expresar como un vector de caracterizaciones clasificadas, (x, c). El algoritmo trabaja con conjuntos homogéneos de evidencias, no las trata individualmente. Los datos pueden ser categóricos o numéricos, pero no se usan las propiedades de los datos numéricos por lo que se tratan todos como categóricos.

El algoritmo trabaja con conjuntos de evidencias, es decir, con matrices, partiendo del mayor catálogo del dataset de clasificación analizado, $\mathcal{D}$. Los resultados obtenidos son también matrices. Si se descubre que $\mathcal{D}^{-\mathcal{I}}$ es un catálogo robusto, no es necesario guardar la matriz $\mathcal{D}^{-\mathcal{I}}$ en memoria RAM, basta con guardar los índices de $\mathcal{I}$ en un árbol $\mathcal{L}$ como el utilizado en el algoritmo Apriori.

El modelo expuesto en esta sección se ha formulado para que pueda implementarse con cualquier herramienta informática que gestione matrices eficientemente. Puede usarse con gestors de bases de datos o con cualquier lenguaje de programación que permita la eliminación de filas o columnas de una matriz.

A continuación se desarrollan las funciones que, utilizando las estructuras de la sección 5.2.2, se utilizan en el algoritmo Apriori, expuesto en la sección 5.2.3. En la sección 5.2.4 se propone un análisis que contrasta el conocimiento descubierto por el algoritmo Apriori con la información que contienen las evidencias separadas del dataset de clasificación en la primera fase del algoritmo.

5.2.1. Funciones

La primera tarea consiste en obtener el mayor catálogo que contiene el dataset (véase listado 5.1) usando la definición de catálogo (def. 5.6) y el lema 5.1, que afirma que todo dataset de clasificación con evidencias completas contiene algún catálogo.

Listado 5.1: Obtención del catálogo $\mathcal{D}_0$

```
ObtenerCatalogo(dataset, N, M0, D0, D?, A, C)
{
   foreach (evidencia e ∈ dataset)
   {
      M0++;

      if (e tiene valores desconocidos)
      {
         D?.add(e);
         continue;
      }

      if (e ∉ D0)          //No se guardan duplicados
         D0.add(e);

      for (i = 1...N)
         if (ei ∉ Ai)
            Ai.add(ei);

      if (eN+1 ∉ C)
         C.add(eN+1);
   }
}
```

En la primera lectura del dataset de clasificación se separan las evidencias incompletas, se crea el mayor catálogo que contiene el dataset y se obtienen sus principales características:

1. el número de evidencias del dataset, $\mathcal{M}_0$,

2. el conjunto de evidencias incompletas $\mathcal{D}_?$,

3. el mayor catálogo que contiene el dataset de clasificación, $\mathcal{D}_0$, de este conjunto se deduce el número de evidencias completas diferentes, $\mathcal{M} = |\mathcal{D}_0|$,

4. el rango de los atributos de $\mathcal{A}$, y

5. el conjunto de clases $\mathcal{C}$.

El rango de los atributos da al investigador información básica sobre el problema a resolver. Rangos con muchos valores denotan, generalmente, atributos numéricos. Los algoritmos de ARM no trabajan bien con este tipo de atributos pues suelen contener muchos valores con soportes pequeños que desaparecen del análisis al usar el criterio del soporte mínimo. El algoritmo $\mathcal{ACDC}$ no suprime ningún dato por su soporte ya que los atributos se observan como un todo, no como un conjunto de valores distintos.

Listado 5.2: Obtención de $\mathcal{D}^{-\mathcal{I}-i}$ a partir de $\mathcal{D}^{-\mathcal{I}}$

```
EliminaColumna(i, 𝒩, ℐ, 𝒟^{-ℐ})
{
    i = ObtenColumnaAEliminar(i, 𝒩, ℐ) //Función 5.3

    𝒟^{-ℐ-i} = ∅;

    foreach (evidencia e^{-ℐ} ∈ 𝒟^{-ℐ})
    {
        e^{-ℐ-i} = e^{-ℐ} sin su i–ésimo valor;

        if (e^{-ℐ-i} ∉ 𝒟^{-ℐ-i}) //No se guardan duplicados
            𝒟^{-ℐ-i}.add(e^{-ℐ-i});
    }

    return 𝒟^{-ℐ-i};
}
```

El análisis propuesto se basa en la supresión de atributos, de uno en uno, y la comprobación de las características del catálogo obtenido. Para suprimir un atributo se necesita un conjunto auxiliar, $\mathcal{I}$, que recoge los atributos que ya han desaparecido en el catálogo $\mathcal{D}^{-\mathcal{I}}$. La función 5.2 crea el catálogo $\mathcal{D}^{-\mathcal{I}-i}$ a partir del catálogo robusto $\mathcal{D}^{-\mathcal{I}}$. En la primera llamada

a la función, $\mathcal{I}$ es el conjunto vacío, por lo que $\mathcal{D}^{-\mathcal{I}}$ es realmente el catálogo $\mathcal{D}$.

La función auxiliar 5.3 determina la posición real de la columna a eliminar. Una vez eliminada cualquier columna en $\mathcal{D}$, las columnas de $\mathcal{D}^{-\mathcal{I}}$ no ocupan la posición indicada por el valor i, referido siempre al conjunto total de atributos, $\mathcal{A}$. Si no se hiciera así, no se podría saber realmente a qué atributos del problema original se refieren las columnas de la matriz $\mathcal{D}^{-\mathcal{I}}$.

Listado 5.3: Obtención de la columna a eliminar

```
ObtenerColumnaAEliminar(i, 𝒩, ℐ)
{
    for (columna_a_eliminar = 0...𝒩) //𝒩 es la clase
    {
        if (columna_a_eliminar ∈ ℐ)
            continue;
        if (columna_a_eliminar == i)
            break;
    }

    return columna_a_eliminar;
}
```

Los atributos con rango unitario son atributos constantes. Según la proposición 5.1 no aportan información sobre el experimento de clasificación y consumen gran cantidad de recursos por lo que, si tras la primera lectura del dataset de clasificación se observa que algún atributo tiene rango unitario, se elimina utilizando la función 5.2.

Listado 5.4: Comprobar si un catálogo es robusto

```
EsRobusto(𝒩, ℐ, 𝒟^-ℐ)
{
    𝒟^-ℐ-𝒞 = EliminaColumna(𝒩,𝒩,ℐ); //Se elimina la última columna

    return (|𝒟^-ℐ-𝒞| == |𝒟^-ℐ|);
}
```

Una vez eliminados los atributos constantes, se ha de comprobar si $\mathcal{D}_0$ es un catálogo robusto (realmente $\mathcal{D}_0^{-\mathcal{I}}$ si el dataset contenía algún atributo constante, pero se usará $\mathcal{D}_0$ por simplicidad). Esta prueba se realizará con submatrices de todos los catálogos robustos que forman el $\mathcal{CCR}_{\mathcal{D}}$, por lo que se define como una función (ver listado 5.4) basada en el teorema 5.1.

Si el catálogo $\mathcal{D}_0$ no es robusto, se ha de separar la incertidumbre presente en el dataset de clasificación, guardándola en $\mathcal{D}_{\mathcal{E}}$ para incorporarla al análisis una vez descubierto el conjunto de catálogos robustos $\mathcal{CCR}_{\mathcal{D}}$.

Sólo es necesario aislar la incertidumbre del primer catálogo obtenido, por lo que se usará una función específica para ello (véase listado 5.5). Esta función separa cada evidencia en el par (x, c), donde x es la caracterización y c su clase asociada. Para descubrir qué caracterizaciones tienen incertidumbre, $\mathcal{D}_{\mathcal{E}}$ utiliza una estructura especial que guarda los clasificadores ingenuos $\mathcal{C}_{\mathcal{D}}(x)$ (ver definición 5.7), es decir, una matriz de $\mathcal{N}$ columnas en que cada fila representa a la caracterización x y está enlazada con el conjunto de clases con las que está relacionada x en el dataset.

Listado 5.5: Aislar incertidumbre del catálogo inicial

```
AislarIncerticumbre(D_0)
{
    D_E = ∅;

    foreach (evidencia e ∈ D_0)
    {
        x = (e_1 ... e_N);
        c = e_{N+1};

        D_0.remove(e);
        D_E.add(x, c); //Se guarda separando caracterización y clase
    }

    foreach (caracterización x ∈ D_E)
        if (|D_E[x]| == 1|) //D_E[x] es C_D(x), clasificador ingenuo de x
        {
            e = (x, c)
            D_E.remove(x, c);
            D_0.add(e)
        }
}
```

Para descubrir todos los catálogos robustos que contiene cada catálogo robusto encontrado en el análisis, se usa una función recursiva (ver listado 5.6).

Listado 5.6: Obtener el $\mathcal{CCR}_{\mathcal{D}}$ de un catálogo robusto

```
ObtenerCCR( ultimo_atributo_eliminado, 𝒩, ℐ, 𝒟^{-ℐ})
{
    for (a =(ultimo_atributo_eliminado+1)...𝒩)
    {
        if (a ∈ ℐ || {-ℐ - a} ∈ 𝒞𝒞ℛ_𝒟)
            continue;

        𝒟^{-ℐ-a} = EliminarColumna(a, 𝒩, ℐ, 𝒟^{-ℐ}); //Función 5.2

        if (EsRobusto(𝒟^{-ℐ-a})) //Función 5.4
        {
            𝒞𝒞ℛ_𝒟.add(𝒟^{-ℐ-a});
            ObtenerCCR(a, 𝒩, ℐ ∪ a, 𝒟^{-ℐ-a});
        }
    }
}
```

Al implementar la función 5.6 se ganará en eficiencia usando la propiedad 5.1 ya que si se ha descubierto que $\mathcal{I} = \{1, 2, 3, 4, 5\}$ proporciona el catálogo robusto $\mathcal{D}^{-\mathcal{I}}$, no es necesario analizar el conjunto de atributos $\mathcal{J} = \{2, 3, 4, 5\}$ pues ya se sabe que $\mathcal{D}^{-\mathcal{J}}$ también es catálogo robusto.

5.2.2. Estructuras

Como se comprueba en la sección 6.3, hay datasets de clasificación con miles de millones de catálogos robustos. Cada catálogo robusto de $\mathcal{CCR}_{\mathcal{D}}$ es una matriz, por lo que manejar en memoria todos los catálogos es inabordable. Lo que se guarda en memoria es el modo de reconstruir todos los catálogos robustos que contiene $\mathcal{CCR}_{\mathcal{D}}$. Para ello basta con guardar en un vector los índices de los atributos eliminados para obtener cada $\mathcal{D}^{-\mathcal{I}}$. $\mathcal{CCR}_{\mathcal{D}}$ se puede definir usando la ecuación 5.17.

$$\mathcal{CCR}_{\mathcal{D}} = \{\mathcal{I} \ / \ \mathcal{D}^{-\mathcal{I}} \text{ es robusto}\} \tag{5.17}$$

La estructura usada para $\mathcal{CC}_{\mathcal{D}_\mathcal{E}}$, la colección de catálogos con incertidumbre, es similar a la utilizada con $\mathcal{CCR}_\mathcal{D}$.

Para gestionar los catálogos robustos se pueden utilizar matrices de valores individuales, conjuntos de evidencias completas, o conjuntos de pares (caracterización, clase). Según la estructura utilizada se necesitarán más o menos recursos y se podrán usar métodos con mayor o menor eficiencia.

5.2.3. Algoritmo ACDC

Este algoritmo no busca evidencias robustas aisladas como hacen los algoritmos de ARM. Su número puede ser muy grande y provocar desbordamiento de memoria. El algoritmo extrae del dataset el mayor catálogo robusto que contiene, separando los conjuntos $\mathcal{D}_?$ y $\mathcal{D}_\mathcal{E}$, con información que será analizada al terminar la ejecución del algoritmo, cuando se haya descubierto la colección de catálogos robustos $\mathcal{CCR}_\mathcal{D}$ que contiene el dataset original.

Listado 5.7: Algoritmo $\mathcal{ACDC}$

```
//Se abre el dataset y se obtiene el número de atributos
𝒩 = (número de datos de la primera evidencia) − 1;
ℳ0 = 0;
𝒟0 = 𝒟? = 𝒞 = ℐ = ∅;
for (i = 1...𝒩)
    Ai = ∅;

ObtenerCatalogo(dataset, 𝒩, ℳ0, 𝒟0, 𝒟?, 𝒜, 𝒞); //Función 5.1

for (i = 1...𝒩) //Se eliminan atributos constantes
    if (rango(Ai) == 1)
    {
        EliminaColumna(i, 𝒩, ℐ, 𝒟^−ℐ); //Ver función 5.2
        ℐ = ℐ ∪ i;
    }

AislarIncerticumbre(𝒟0); //Función 5.5

ObtenerCCR(0, 𝒜 − ℐ, ℐ, 𝒟^−ℐ); //Función 5.6
```

El algoritmo se resume en la instrucciones mostradas en el listado 5.7. Para usarlo se indica la ubicación del dataset, se abre y se leen sus metadatos –si los tuviera– hasta obtener la primera evidencia, de la que extraemos el número de atributos contando el número de datos que contiene y considerando que el último de ellos se corresponde con la clase.

Para analizar datasets de clasificación de grandes dimensiones. puede ser conveniente restringir el número de catálogos robustos a encontrar. Para conseguirlo basta con añadir un parámetro a la función 5.6, indicando el número máximo de llamadas recursivas. Si se indica el valor k, se eliminan como máximo k atributos del catálogo recibido como parámetro.

5.2.4. Post-análisis

Una vez se ha obtenido el conjunto de catálogos robustos $\mathcal{CCR}_{\mathcal{D}}$, si $\mathcal{D}_{\mathcal{E}}$ y $\mathcal{D}_{?}$ no son conjuntos vacíos, se debe contrastar la información que contienen con la descubierta por el algoritmo $\mathcal{ACDC}$.

Las caracterizaciones con incertidumbre de $\mathcal{D}_{\mathcal{E}}$ reflejan un diseño incompleto del experimento de clasificación, bien por la falta de atributos que realmente discriminen las clases en estudio, bien debido al uso de herramientas sin la suficiente precisión para captar el poder de discriminación de algún atributo. Ninguna de sus caracterizaciones están en $\mathcal{D}$, por lo que no existe ninguna contradicción entre la información que proporcionan directamente ambos conjuntos. Sin embargo, las sub-caracterizaciones de $\mathcal{D}_{\mathcal{E}}$ pueden contradecir la información descubierta en el conjunto de catálogos robustos $\mathcal{CCR}_{\mathcal{D}}$.

Sobre los registros de $\mathcal{D}_{?}$ sólo se sabe que todas sus evidencias contienen algún valor desconocido. Al suprimir sus valores desconocidos, se obtiene una sub-evidencia completa, que puede contradecir o reforzar la información descubierta en el conjunto de catálogos robustos $\mathcal{CCR}_{\mathcal{D}}$.

El post-análisis consiste en comprobar qué sub-evidencias de $\mathcal{D}_{\mathcal{E}}$ y $\mathcal{D}_{?}$ contradicen los clasificadores ingenuos obtenidos, generando un conjunto de excepciones a $\mathcal{CCR}_{\mathcal{D}}$. Si se guarda esta información, cuando se quiera clasificar a un individuo utilizando una sub-caracterización, primero se comprueba en el conjunto $\mathcal{CCR}_{\mathcal{D}}$ si la caracterización es robusta. Si es robusta, se busca en el conjunto de excepciones, si se encuentra en este conjunto es porque no es robusta, en caso contrario sí que sería robusta y se puede clasificar al individuo usando una regla con un 100 % de confianza.

UNIVERSITAS
Miguel
Hernández

6

Validación

Ningún modelo teórico es realmente útil mientras no se demuestre que se puede poner en práctica, lo que no siempre es posible con la tecnología disponible por la mayoría de investigadores. Para aplicar la propiedad 5.1, por ejemplo, se ha de buscar el conjunto de atributos $\mathcal{I}$ dentro de todos los conjuntos de atributos guardados en $\mathcal{CCR}_{\mathcal{D}}$. Al ejecutar las primeras versiones del algoritmo $\mathcal{ACDC}$ se descubrió que el número de conjuntos de atributos que contiene es tan grande que se tardaría demasiado tiempo en descubrir si contiene a $\mathcal{I}$, por lo que no se implementó la propiedad en el algoritmo. Sólo en casos extremos, y en los primeros pasos del algoritmo, podría ser útil su implementación.

Todos los experimentos realizados para validar los modelos presentados en el capítulo 5 se han desarrollado y ejecutado sobre un ordenador de gama media, con procesador Intel Core i5 y 8GB de memoria RAM corriendo sobre el sistema operarivo MacOS. Las características del equipo y el código utilizado están a disposición de cualquier equipo de investigación, lo que garantiza que pueda ser probado y mejorado por la comunidad científica. Para implementar el algoritmo $\mathcal{ACDC}$ se ha usado la *C++ Standard Library (libc++ 5.0)* (2011) y `clang` con la intención de obtener un código estándar y compatible con la colección de compiladores `gcc`.

Esta fase de la investigación se ha desarrollado utilizando los 75 datasets de clasificación publicados en *KEEL Standard Dataset Repository* (2004). Algunos datasets podrían ser considerados muestras representativas, otros no tienen evidencias duplicadas y tienen el aspecto de catálogos. Todos

han sido tratados sin aplicar el criterio de soporte mínimo, y en ningún caso se ha detenido la ejecución del algoritmo debido a los problemas de desbordamiento típicos de ARM.

Los algoritmos de minería de datos pueden generar más resultados de los que se pueden gestionar con el ordenador en el que se ejecutan. Es recomendable realizar un análisis descriptivo preliminar sobre el dataset antes de aplicar un algoritmo como $\mathcal{ACDC}$. Como se indicó en la sección 5.2.3, la función que genera la colección de catálogos robustos puede ajustarse para que sólo ejecute k iteraciones. En la sección 6.1 se muestra el resultado de aplicar la primera iteración del algoritmo $\mathcal{ACDC}$ sobre los 75 datasets de clasificación de KEEL. Este análisis preliminar proporciona información sobre cada uno de los datasets, como el número de atributos que contiene y cuántos de ellos son redundantes. Esta información permite diseñar la ejecución del algoritmo completo sin que se aborte debido a problemas de desbordamiento de memoria.

Según la proposición 5.1, los atributos constantes no contienen información sobre el problema de clasificación en estudio. En la sección 6.2 se demuestra que la presencia de estos atributos en un dataset de clasificación duplica las necesidades de recursos de memoria y el tiempo empleado, lo que justifica su eliminación del dataset en la primera fase del análisis.

En la sección 6.3 se muestran los resultados obtenidos al ejecutar el algoritmo $\mathcal{ACDC}$ completo sobre algunos de los datasets de clasificación de KEEL.

En la sección 6.4 se analizan datasets de clasificación de grandes dimensiones. Debido a la explosión de resultados que proporcionan estos datasets, no se puede ejecutar el algoritmo de forma no supervisada. Se aplica ajustando el número de iteraciones del algoritmo, de modo que no aparezca desbordamiento de memoria. Con los resultados obtenidos, se selecciona uno de los catálogos robustos reducidos para aplicarle el algoritmo $\mathcal{ACDC}$, restringiendo sus iteraciones si las dimensiones de este dataset son todavía grandes.

6.1. Primera lectura de un dataset de clasificación

En el capítulo 2 se expuso que la mayor parte del tiempo empleado en un proceso de KDD consiste en tareas de pre-proceso de los datos ob-

tenidos para el experimento. En la investigación revisada sobre CARM, el pre-proceso aplicado es, a lo sumo, una discretización de los atributos numéricos para reducir su rango. También se aplica soporte mínimo para reducir el número de ítems a procesar por los algoritmos de ARM. Ya se ha discutido que estos pre-procesos suponen una pérdida de datos obtenidos durante el trabajo de campo llevado a cabo para el experimento.

Los datasets de clasificación utilizados en este capítulo ya han sido pre-procesados, están preparados para ser analizados mediante algún algoritmo de DM. El algoritmo $\mathcal{ACDC}$ descubre todas las reglas de asociación con 100 % de confianza que contiene un dataset de clasificación. Si un dataset tiene un número excesivo de reglas, al analizarlo sin usar soporte mínimo aparece el dilema del ítem raro. El hecho de que un ítem sea «raro» no garantiza que contenga poca información sobre clasificación. Es necesario hacer un pre-proceso de los datasets de clasificación antes de aplicar el algoritmo $\mathcal{ACDC}$.

A modo de pre-proceso, para obtener una descripción de las dimensiones reales del problema que se quiere resolver, se puede aplicar sólo la primera iteración del algoritmo $\mathcal{ACDC}$. Esta iteración descubre cuántos de los $\mathcal{N}$ atributos del experimento de clasificación son redundantes, información que es útil para estimar cuántas combinaciones de estos atributos pueden generar un catálogo robusto. Se ejecutó la primera iteración de $\mathcal{ACDC}$ sobre los 75 datasets de clasificación de KEEL, obteniendo la descripción que se muestra en la tabla 6.7.

Los resultados de la tabla 6.7 (ver página 172) se han obtenido al ejecutar el código del apéndice D.3, en menos de un minuto y utilizando 170.2MB de memoria RAM. Lo más interesante no es el poco tiempo empleado para analizar los 75 datasets publicados en *KEEL Standard Dataset Repository* (2004). Sin prescindir de ningún dato en ningún dataset y sin problemas de desbordamiento se pueden analizar todos los datasets de un repositorio público muy completo y heterogéneo.

El contenido de la tabla se ilustra con la información obtenida para el primer dataset de clasificación del repositorio, *KEEL abalone* (1995), experimento de clasificación realizado para estimar la edad del *abulón* u *oreja de mar*. El método más utilizado para estimar su edad consiste en contar los anillos que se han formado en su cáscara, método destructivo que quiere sustituirse por la medición de otros atributos. Con los valores obtenidos se quiere estimar el número de anillos que contiene, y su edad a partir

de este dato. Al estimar un atributo numérico, el número de anillos de un abulón, sería más apropiado un análisis de regresión. Sin embargo, el uso de reglas de asociación ha demostrado gran eficiencia y precisión en este tipo de experimentos.

> Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone and counting the number of rings through a microscope. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

Este dataset de clasificación se publicó originalmente en *UCI abalone* (1995). Al prepararlo para el repositorio KEEL se suprimieron tres registros duplicados. La única descripción encontrada sobre el hecho de eliminar duplicados está en *KEEL abalone* (1995):

> In this version, 3 duplicated instances have been removed from the original UCI dataset.

No hay ninguna justificación científica para este pre-proceso. Si se eliminan los duplicados de un dataset de clasificación, no debe considerarse ni analizarse como una muestra representativa de la población en estudio. De este razonamiento y de las dificultades encontradas en la sección 3.2 para analizar colecciones no excesivamente grandes de datos usando ARM, surgió la investigación presentada en esta memoria.

A continuación se describe el contenido de las columnas de la tabla 6.7, utilizando todos los valores de su primera fila. Se comentan los resultados más destacados del resto de filas de la tabla.

***col*1** Número de orden del dataset, ordenados alfabéticamente.

`[abalone]` 1

***col*2** Nombre del dataset.

`[abalone]` abalone

col3 Tamaño que ocupa el dataset en disco. Hay datasets extremadamente pequeños, como `appendicitis` que ocupa tan solo 4.73KB, y datasets de hasta 71.89MB, como `kddcup`. En la sección 6.4 se pone a prueba otra versión de este dataset obtenida en *UCI Machine Learning Repository* 2013, `kddcup99`, que ocupa 708.18MB.

`[abalone]` 224.58KB Es un archivo pequeño si se compara con los que se usan en los experimentos realizados en el capítulo 3. El problema que presenta su análisis mediante técnicas de ARM no está en su tamaño, se debe al número de ítems distintos que contiene y la alta relación entre ellos.

col4 Número de atributos del dataset, seguido de dos valores, el número de atributos redundantes y el de atributos constantes, que también son redundantes según la proposición 5.1.

63 de los 75 datasets de clasificación tienen atributos redundantes, siendo 44 los que no tienen ningún atributo esencial. Esto da mucho juego al algoritmo $\mathcal{ACDC}$ pues se basa en reducir las dimensiones del catálogo robusto inicial, eliminando de uno en uno los atributos redundantes y comprobando si puede ser reducido aún más. Los 9 datasets que no tienen atributos redundantes están marcados en color rojo para su rápida identificación.

En cuanto al número de atributos constantes, sorprende encontrar 6 datasets con atributos de este tipo. En la sección 6.2 se mostrarán los beneficios obtenidos al utilizar esta información en el análisis completo de un dataset de clasificación.

`[abalone]` $8_{(8)(\text{-})}$ Tiene 8 atributos, todos redundantes y ninguno constante. Está marcado en color verde para destacar que no tiene atributos esenciales, es decir, si sólo se tiene información sobre 7 de los 8 atributos del experimento, se obtendrá la misma clasificación que si se tuviera información sobre los 8 atributos.

col5 Número de valores diferentes que hay en las caracterizaciones del dataset, a los que hay que sumar el número de clases reflejado en la siguiente columna. Es un dato importante para conocer el número de ítems con que trabajan los algoritmos de ARM al tratar el dataset

sin aplicar transformaciones ni reducciones. Se obtiene a partir del rango de los atributos del experimento, sumando su cardinal.

En esta columna se aprecia que hay muchos atributos numéricos con la suficiente precisión como para generar atributos con rangos muy amplios, atributos que provocarán la aparición del dilema del ítem raro si se tratan como variables categóricas.

[abalone] 6,047 Usando sólo 8 atributos se obtienen más de 6,000 valores diferentes, lo que nos descubre el uso de atributos numéricos. Cuantos más valores distintos tiene un atributo menor será el soporte de cada valor en el dataset.

Si se usa el umbral de soporte mínimo, la mayoría de estos valores tendrán un soporte muy bajo y serán completamente ignorados, dejando para el análisis sólo algún valor, o ninguno, del atributo en cuestión. Sin embargo, el investigador no es informado de esta circunstancia al utilizar algoritmos de ARM.

col6 Número de clases.

[abalone] 28 Se trata del dataset con más clases de la colección. Es un experimento que se podría analizar mejor utilizando técnicas de regresión, que tienen en cuenta las propiedades numéricas de los valores del dataset.

Si se aplica el umbral de soporte mínimo, alguna de las clases puede desaparecer en el análisis, dejando caracterizaciones sin clasificar en el dataset finalmente analizado. Las caracterizaciones sin clasificar no son evidencias, no contienen información para resolver el problema de clasificación.

col7 Evidencias completas que contiene el dataset. En el algoritmo *ACDC* no se consideran las evidencias incompletas, las caracterizaciones con datos desconocidos se separan para un análisis posterior. Ninguno de los datasets de clasificación de *KEEL Standard Dataset Repository* 2004 contiene evidencias con datos desconocidos, pero alguno de ellos procede de una colección con datos desconocidos, como `mushroom` (UCI).

[abalone] 4,174 El número de evidencias completas recogida para el experimento de clasificación. Sólo puede garantizar que este dato

sea correcto el investigador encargado del experimento y de la toma y tratamiento de datos. Las transformaciones aplicadas a los datos no modifican este valor, pero si se decide suprimir duplicados para reducir las dimensiones del dataset y no se documenta adecuadamente, cualquier usuario de este dataset desconocerá el número real de evidencias completas recogidas para el experimento. Este dato no se utiliza en el algoritmo $\mathcal{ACDC}$, por lo que no le afecta el hecho de que pueda ser un dato realmente desconocido.

***col*8** Evidencias completas distintas del dataset. El objetivo de esta tesis es el análisis del catálogo que contiene cualquier dataset de clasificación (ver lema 5.1), por lo que no se analizan los duplicados. Esto elimina la necesidad de utilizar el *soporte* como medida de calidad. 37 de los datasets de clasificación de KEEL no tienen duplicados, es decir, la mitad de los datasets analizados son realmente catálogos. Se ha marcado el dato de esta columna en color verde para una rápida identificación de los catálogos del repositorio KEEL.

`[abalone]` 4,174 Todas las evidencias completas de este dataset de clasificación son únicas, no tiene duplicados. Lo sabemos porque está marcado con color verde y coincide con el número de evidencias completas del dataset. Es el número de evidencias completas distintas que forman el catálogo con el que trabaja el algoritmo $\mathcal{ACDC}$.

Desde la perspectiva clásica, un catálogo con tantas evidencias no es útil. Una guía formada por más de 4,000 fichas es poco manejable. Sin embargo, la tecnología actual hace pequeño este valor, son muchos los dispositivos informáticos capaces de gestionar eficientemente millones de evidencias completas. Este es uno de los motivos que permitió completar la investigación de esta tesis.

***col*9** Caracterizaciones completas con incertidumbre. Este dato es uno de los que más me sorprende. Sólo 19 de los 75 datasets analizados contienen incertidumbre, lo que indica que hay 56 experimentos basados en atributos que explican, sin ninguna duda, la clase de pertenencia de un individuo de la población en estudio.

La incertidumbre no se utiliza en el algoritmo $\mathcal{ACDC}$, pero es necesario conocerla para hacer un análisis posterior a la ejecución del algoritmo. Se ha de contrastar la información obtenida sobre las evidencias robustas del catálogo analizado con la que contienen las caracterizaciones con incertidumbre almacenadas en el conjunto $\mathcal{D}_{\mathcal{E}}$.

[abalone] [-] No hay incertidumbre en este dataset. Todas sus caracterizaciones son únicas, no hay dos individuos de distinta clase con la misma caracterización.

Si se quiere clasificar a un nuevo individuo y su caracterización ya están registrada, se usará un clasificador ingenuo $\mathcal{C}_{\mathcal{D}}(x)$ con una confianza del 100 %, no habrá ninguna duda en esta clasificación. Si una caracterización no están registrada, no se pueden utilizar directamente $\mathcal{C}_{\mathcal{D}}(x)$, pero se pueden estudiar sus subcaracterizaciones con el objetivo de clasificar al individuo con el clasificador $\mathcal{C}_{\mathcal{D}^{-\mathcal{I}}}(x^{-\mathcal{I}})$, e incorporar una nueva evidencia al dataset.

***col*10** Si el dataset de clasificación tiene atributos redundantes, se muestra el atributo redundante, A_i, que reduce más el número de evidencias en $\mathcal{D}^{-i}$. A continuación se indican las características básicas del dataset reducido cuando se elimina del dataset original el atributo A_i: el número de valores diferentes que tienen sus atributos, $\mathcal{A}^{-i}$, y el de evidencias de $\mathcal{D}^{-i}$.

Es un buen recurso si se quiere aplicar el algoritmo $\mathcal{ACDC}$ de un modo supervisado, por ejemplo, para analizar datasets de grandes dimensiones se eliminan primero los atributos redundantes que más reducen el número de evidencias en el catálogo reducido.

[abalone] A_1 (6,044 / 4,174) Si se elimina el atributo A_1 (`sexo`, con tres valores: `M`, `F` e `I`-infante) se obtiene el dataset $\mathcal{D}^{-1}$, con 4,174 evidencias robustas completas, las mismas que el catálogo robusto $\mathcal{D}$, pero con un atributo menos.

En este caso no se reduce el número de evidencias al eliminar sólo uno de los atributos redundantes del experimento. El mejor atributo a eliminar para reducir el número de evidencias a analizar es el primero porque ninguno de los restantes ofrece

mejores resultados, lo que sólo depende del orden utilizado para guardar los atributos en el dataset.

Los datasets `appendicitis`, `balance`, `bands`... y hasta un total de 30 datasets de la colección, tienen esta característica.

En otros casos, eliminando sólo un atributo se reduce notablemente el tamaño del catálogo, como ocurre con `mushroom`, al eliminar su atributo A_9 (`gill-color`, con 9 valores diferentes) se reduce el número de evidencias a analizar en un 70.5 %, siendo $|\mathcal{D}| = 5,644$ y $|\mathcal{D}^{-9}| = 1,662$, teniendo ambos la misma información para clasificación. Algo similar ocurre con los datasets `monk-2` y `shuttle`.

Estos datos se refieren únicamente al efecto de eliminar un único atributo del dataset, en la siguiente sección se comprobará que, si se puede seguir eliminando atributos redundantes, el número de evidencias de los datasets reducidos será cada vez menor en la mayoría de situaciones.

***col*11** En los datasets con atributos redundantes, se indica el atributo A_i que minimiza el número de valores distintos de $\mathcal{A}^{-i}$. Entre paréntesis el número de valores distintos que tienen los atributos de $\mathcal{A}^{-i}$ y el número de evidencias de $\mathcal{D}^{-i}$.

`[abalone]` $\boxed{A_5 \text{ (3,619 / 4,174)}}$ Como el dataset de clasificación original tiene 6.047 valores distintos, al eliminar A_5 (`Whole weight`) quedan sólo 3,619 valores distintos en $\mathcal{D}^{-5}$. Se debe a que A_5 es un atributo numérico, concretamente el peso en gramos del individuo, con 6,047 - 3,619 = 2,428 valores distintos. Si se cruza este dato con el número total de evidencias, 4,174, se deduce que la mayoría de valores de este atributo sólo está una o dos veces en el dataset, es decir, serían tratados como ítems raros en los algoritmos de CARM.

***col*12** Tiempo, en segundos, empleado en descubrir esta información. Sólo en 7 de los 75 datasets se ha necesitado más de un segundo para averiguar las características que tiene el dataset en el experimento de clasificación para el que se ha obtenido.

`[abalone]` $\boxed{0.069}$ En menos de una décima de segundo, se ha descubier-

to que el dataset de clasificación `abalone` se puede reducir sin pérdida alguna de la información que contienen sus datos para resolver el problema de clasificación. En la sección 6.3 se comprueba que usando sólo 3 atributos, los datos recogidos dan la misma información para este experimento de clasificación.

Vale la pena aplicar sólo la primera iteración del algoritmo $\mathcal{ACDC}$ sobre cualquier dataset de clasificación antes de proceder a un análisis más severo. La información descubierta proporciona al investigador la oportunidad de diseñar mejor un análisis más profundo. Una revisión detallada de la tabla 6.7 (pg. 172) proporciona información como:

- Entre los 75 datasets de clasificación analizados, 63 tienen algún atributo redundante y 43 de ellos tienen todos sus atributos redundantes.
- 37 de los 75 datasets de clasificación son catálogos robustos.
- Sólo 19 de los 75 datasets analizados tienen incertidumbre.
- Si se elimina el atributo A_9 del dataset de clasificación `mushroom`, se necesitan únicamente 1,662 evidencias robustas de las 5,644 que contiene el catálogo robusto original.

Solo hay 12 datasets de clasificación que no tienen atributos redundantes: `balance`, `banana`, `car`, `contraceptive`, `haberman`... El algoritmo $\mathcal{ACDC}$ no puede reducir las dimensiones de estos datasets. Muchos de ellos son datasets sintéticos preparados para poner a prueba diferentes algoritmos de minería de datos.

`balance`, por ejemplo, es un catálogo robusto que contiene todas las combinaciones de los 5 valores de cada uno de sus 4 atributos, está formado por $5^4 = 625$ evidencias. Es un ejemplo típico de catálogo basado en pocas propiedades numéricas. Se utiliza en experimentos de clasificación, pero en la documentación que muestra en *UCI Machine Learning Repository* 2013[1] se indica que la solución para clasificación se obtiene comparando el producto de los dos primeros valores con el producto de los dos restantes, si son iguales se ha de clasificar como «balanceado».

[1] `https://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/balance-scale.names`

banana contiene sólo dos atributos numéricos. Tiene 5,291 combinaciones diferentes de los valores de estos dos atributos, y sólo una de ellas presenta incertidumbre. Es un dataset que se analiza mejor con regresión que técnicas aplicables a variables categóricas.

Estos ejemplos recuerdan a los investigadores que no se pueden aplicar técnicas de ARM a cualquier tipo de datos, ya que estas técnicas trabajan con variables categóricas que no tienen las propiedades de las variables numéricas.

6.2. Atributos constantes

En la tabla 6.7 se descubre que 6 de los 75 datasets que forman *KEEL Standard Dataset Repository* 2004 tienen atributos constantes: automobile, census, kddcup, mushroom, optdigits y segment. Tres de estos datasets de clasificación –census, kddcup y optdigits– son de grandes dimensiones. Tienen muchos atributos redundantes, lo que obliga a revisar un gran número de combinaciones de estos atributos. Se analizarán en la sección6.4.

La proposición 5.1 postula la necesidad de eliminar los atributos constantes antes de caracterizar al resto de atributos, y el algoritmo $\mathcal{ACDC}$ la incorpora en sus primeros pasos, después de leer el dataset y antes de hacer cualquier otra operación con sus datos.

Para comprobar el efecto de los atributos constantes en el análisis de datasets de clasificación y documentar esta sección, se ha ejecutado una modificación del algoritmo $\mathcal{ACDC}$ sobre automobile, mushroom y segment. La modificación consiste en no dar importancia a este tipo de atributos y dejarlos en el análisis, como se hace en todas las aportaciones mostradas en la sección 2.2. A continuación se ha ejecutado el algoritmo $\mathcal{ACDC}$ sin modificaciones. Los resultados de ambas ejecuciones se muestran en la tabla 6.1.

La primera columna muestra el nombre del dataset analizado. La segunda muestra el número de evidencias robustas que tiene el dataset original. La tercera columna muestra el número de atributos y la cuarta cuántos de ellos son redundantes.

En las siguientes columnas hay dos filas para cada dataset de clasificación, en la superior se indica los resultados obtenidos cuando no se suprimen los atributos constantes, mientras que la inferior refleja el com-

Tabla 6.1: Efecto de la supresión de atributos constantes

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{A}\rvert$	Atributos redundantes	$\lvert\mathcal{CCR}_{\mathcal{D}}\rvert$	Evidencias robustas	Tiempo (sg.)
automobile	159	25	25	18,182,384	2,788,517,384	3,768
				9,091,192	1,394,258,692	1,839
mushroom	5,644	22	22	2,093,424	1,204,584,784	2,077
				1,046,712	602,292,392	1,026
segment	2,086	19	18	261,952	545,253,360	889
				130,976	272,626,680	413

portamiento del algoritmo $\mathcal{ACDC}$. La quinta columna muestra el número de catálogos robustos que contiene el dataset, la siguiente columna muestra el número de evidencias robustas que contienen en total, y la última columna muestra el tiempo empleado en obtener estos resultados.

Es fácil comprobar que si no se aplica la proposición 5.1 serán necesarios más de el doble de recursos para obtener, en más del doble de tiempo, los mismos resultados. Si A_i es un atributo constante que sólo toma el valor v_i en el dataset de clasificación original, la única diferencia entre los catálogos robustos $\mathcal{D}^{-\mathcal{I}}$ y $\mathcal{D}^{-\mathcal{I}-i}$ es una columna con el valor v_i en la columna correspondiente.

Los atributos constantes no deben eliminarse del experimento de clasificación sin una justificación. Si el experimento está bien diseñado, es de esperar que el atributo aporte información al experimento cuando se tomen muestras en que el atributo en cuestión no sea constante. Sin embargo, si el dataset de clasificación que se va a analizar contiene atributos constantes, se deben suprimir para mejorar la eficiencia del análisis.

6.3. El algoritmo $\mathcal{ACDC}$

En la sección 6.1 se ha ejecutado la primera iteración del algoritmo $\mathcal{ACDC}$ sobre todos los datasets de clasificación de *KEEL Standard Dataset Repository* (2004), obteniendo características descriptivas de cada uno de ellos. En la sección 6.2 se empiezan a ver las dimensiones de las colecciones de catálogos robustos que contienen los datasets de clasificación. Un dataset de clasificación que ocupa en disco menos de 23KB –`automobile`–

puede contener más de 18 millones de evidencias robustas debido a que sus 25 atributos son redundantes y generan una gran cantidad de combinaciones.

No es aconsejable aplicar el algoritmo completo a cualquier dataset de clasificación antes de conocer sus características, pueden aparecer problemas de desbordamiento de recursos tras horas de análisis, abortando la ejecución del algoritmo $\mathcal{ACDC}$ antes de obtener todos los resultados que puede descubrir.

En esta sección se muestran los descubrimientos más relevantes obtenidos al aplicar el algoritmo $\mathcal{ACDC}$ a datasets de clasificación con un número moderado de atributos redundantes. Una vez se sabe que el dataset de clasificación puede ser gestionado en memoria, lo que se ha comprobado con todos los que forman la colección de *KEEL Standard Dataset Repository* 2004 en la sección 6.1, el algoritmo busca todas las combinaciones de atributos redundantes que proporcionen un catálogo robusto reducido. Los datasets de clasificación con mayores dimensiones se analizarán de forma supervisada en la sección 6.4.

La tabla 6.2 muestra los resultados obtenidos al aplicar el algoritmo $\mathcal{ACDC}$ de forma no supervisada a todos los atributos con 25 o menos atributos redundantes. El umbral se ha decidido simplemente por las capacidades del equipo con que se ha llevado a cabo esta investigación, podría aumentarse si se dispone de mejores recursos.

La primera columna muestra el nombre del dataset de clasificación. En la segunda se indica el número de catálogos robustos que contiene y en la siguiente se indica el número de evidencias robustas que forman esos catálogos. En las dos siguientes columnas se muestra el número de evidencias robustas que tiene el catálogo robusto inicial y el que tiene el catálogo robusto con menos filas del $\mathcal{CCR}_{\mathcal{D}}$. En las dos siguientes columnas se hace la misma comparación respecto al número de atributos, es decir, la sexta columna contiene el número de atributos del experimento de clasificación y la séptima, encabezada con la marca (1), contiene el mínimo número de atributos necesario para generar un catálogo robusto en el experimento. En la última columna se muestra el tiempo empleado en la ejecución completa del algoritmo.

Tabla 6.2: Resultados de la ejecución del algoritmo $\mathcal{ACDC}$ sin supervisión

Dataset	$\lvert\mathcal{CCR}_{\mathcal{D}}\rvert$	**Evidencias robustas**	$\lvert\mathcal{D}\rvert$	$\min \lvert\mathcal{D}^{-\mathcal{I}}\rvert$	$\lvert\mathcal{A}\rvert$	**(1)**	**Tiempo (sg.)**
abalone	94	392,352	4,174	4,172	8	3	0.61
adult	96	4,335,264	45,170	45,153	14	8	8.42
appendicitis	118	12,496	106	104	7	2	0.01
australian	6,646	4,576,806	690	674	14	3	5.27
automobile	9,091,192	1,394,258,692	159	75	25	2	1,839.99
bands	452,244	164,387,074	365	323	19	4	181.55
breast	4	991	257	243	9	7	0.00
bupa	28	9,535	341	337	6	3	0.01
chess	288	882,560	3,196	2,826	36	29	2.80
cleveland	5,306	1,573,725	297	280	13	3	1.85
crx	18,924	12,327,144	653	621	15	3	15.30
ecoli	521	17,452	336	334	7	3	0.03
fars	315	27,514,164	90,097	87,262	29	23	67.86
flare	2	444	287	213	11	10	0.005
german	573,982	573,645,094	1,000	975	20	20	751.62
glass	440	93,419	213	204	9	2	0.15
hayes-roth	2	130	84	55	4	3	0.00
heart	5,694	1,534,040	270	245	13	3	1.45
hepatitis	496,789	39,437,484	80	46	19	2	36.95
housevotes	332	43,340	160	78	16	8	0.12
iris	5	718	147	142	4	3	0.00
letter	283	5,108,962	18,668	16,841	16	11	13.08
lymph[2]	41,457	5,933,244	148	113	18	6	8.89
magic	984	18,602,291	18,905	18,893	10	2	26.59
monk-2	8	1,296	432	36	6	3	0.00
mushroom	1,046,712	602,292,392	5,644	22	22	3	1,085.09
newthyroid	18	3,845	215	203	5	2	0.01
page-blocks	565	3,032,340	5,393	5,230	10	3	4.15
penbased	55,404	608,900,100	10,992	10,660	16	5	952.20
phoneme	16	84,962	5,349	5,238	5	2	0.22
pima	158	121,306	768	764	8	3	0.18
ring	1,048,450	7,758,529,904	7,400	7,396	20	2	15,515.26
saheart	435	200,899	462	454	9	2	0.31
segment	130,976	272,626,680	2,086	2,074	19	2	600.20
shuttle	160	5,008,378	57,999	6,455	9	4	12.03
tae	4	406	106	101	5	3	0.00
thyroid	982,016	6,865,495,424	7,129	5,454	21	3	14,441.62
tic-tac-toe	10	9,580	958	958	9	8	0.04

[2] `lymphography`

Resultados de la ejecución del algoritmo $\mathcal{ACDC}$ sin supervisión (cont.)

Dataset	$\lvert\mathcal{CCR}_{\mathcal{D}}\rvert$	Evidencias robustas	$\lvert\mathcal{D}\rvert$	mín $\lvert\mathcal{D}^{-\mathcal{I}}\rvert$	$\lvert\mathcal{A}\rvert$	(1)	Tiempo (sg.)
titanic	8	19	14	1	3	1	0.00
twonorm	1,048,554	7,759,299,600	7,400	7,400	20	2	14,004.66
vehicle	257,153	217,500,407	846	820	18	3	273.01
vowel	8,092	8,011,078	990	989	13	2	13.30
wine	8,147	1,450,107	178	171	13	2	2.23
win-red[3]	1,707	2,312,932	1,359	1,347	11	3	3.90
win-white[4]	1,311	5,169,560	3,961	3,916	11	4	9.94
wisconsin	142	51,120	449	278	9	4	0.09
yeast	32	46,468	1,453	1,450	8	4	0.14
zoo	8,912	361,576	59	20	16	5	0.57

A continuación se discute la información más relevante descubierta por el algoritmo $\mathcal{ACDC}$ en los datasets de clasificación analizados en esta sección.

6.3.1. Número de catálogos robustos del dataset de clasificación

Cada uno de los catálogos robustos descubierto por el algoritmo $\mathcal{ACDC}$ contiene la misma información para clasificación que el mayor catálogo del dataset de clasificación. Por ejemplo, si $\mathcal{D}^{-3}$ y $\mathcal{D}^{-2-7}$ son catálogos robustos, utilizando sólo los atributos del conjunto $\mathcal{A}^{-3}$ o $\mathcal{A}^{-2-7}$ se clasifican a los individuos de la población del mismo modo que si se usara el catálogo robusto $\mathcal{D}$, usando los $\mathcal{N}$ atributos del experimento de clasificación.

Aunque no era uno de los objetivos de esta investigación, los resultados obtenidos se engloban en las técnicas de selección de atributos (en inglés, Features Selection). Cada uno de los catálogos robustos descubierto representa a una selección de atributos, $\mathcal{A}^{-\mathcal{I}}$, que proporciona exactamente la misma información sobre clasificación que la obtenida con el dataset de clasificación original.

Cuantos más catálogos robustos tiene un dataset de clasificación, más posibilidades tiene el investigador a la hora de seleccionar los atributos que va a medir en un nuevo individuo para clasificarlo. Si no tiene la herramienta para medir el atributo redundante A_i, puede utilizar la caracterización incompleta x^{-i} para clasificar a un individuo a partir del catálogo

[3] `winequality-red`

[4] `winequality-white`

robusto $\mathcal{D}^{-i}$. Si $\mathcal{D}^{-\mathcal{I}}$ es un catálogo robusto, puede obtener sólo la caracterización $x^{-\mathcal{I}}$ y usarla para clasificar a un individuo sin medir en él los $\mathcal{N}$ atributos del experimento de clasificación.

`automobile` tiene 9,091,192 catálogos robustos. Para clasificar un automóvil utilizando este dataset de clasificación, que no tiene incertidumbre según indica la tabla 6.7, se puede usar cualquiera de las más de 9 millones de combinaciones de atributos descubiertas. El experimento de clasificación se ha diseñado recogiendo información sobre 25 atributos, todos redundantes según la tabla 6.7. Usando sólo dos de los atributos, según se indica en la séptima columna de la tabla 6.2, se obtiene la misma clasificación que usándolos todos.

Los datasets de clasificación `mushroom`, `ring` y `twonorm` tienen más de un millón de catálogos robustos. Cuanto mayor es este número, más juego tiene el investigador para clasificar a un nuevo individuo utilizando caracterizaciones incompletas.

6.3.2. Número mínimo de evidencias robustas

La tercera columna de la tabla 6.2 muestra el número de evidencias robustas descubiertas en el dataset de clasificación original. Son reglas de asociación con un 100 % de confianza, pero no son todas las que contiene el dataset, sólo son las que contienen los catálogos robustos descubiertos por el algoritmo $\mathcal{ACDC}$. Las evidencias robustas son clasificadores ingenuos, cuantas más se descubran más probable será que una de ellas sirva para clasificar correctamente a un nuevo individuo de la población.

Supóngase que los atributos A_i y A_j proporcionan miles de evidencias robustas, miles de caracterizaciones (x_1, x_2) asociadas únicamente a una clase. Supóngase también que el dataset de clasificación contiene las evidencias (x_1^1, x_2^1, c) y (x_1^1, x_2^1, c'). La caracterización (x_1^1, x_2^1) contiene incertidumbre, por lo que el catálogo $\mathcal{D}^{-1-2}$ no es robusto. En este caso, el algoritmo $\mathcal{ACDC}$ no descubre las miles de evidencias robustas que contiene $\mathcal{D}^{-1-2}$, para hacerlo tendría que gestionarlas individualmente para diferenciarlas de las que no son robustas, lo que provocaría problemas con la gestión de recursos necesarios para ejecutar el algoritmo.

Los números que muestra la tabla 6.2 dan una idea de la dificultad que tienen los algoritmos de ARM para encontrar las mejores reglas de asociación de un dataset de clasificación. Esto explica la necesidad de fijar

un soporte mínimo en las investigaciones sobre CARM expuestas en la sección 2.2.

ring Este dataset de clasificación contiene, como mínimo, 7,758,529,904 evidencias robustas. Se trata de un dataset sintético con 20 atributos reales procedentes de distribuciones normales multivariantes.

twonorm Este dataset de clasificación contiene, como mínimo, 7,759,299,600 evidencias robustas. Es muy similar al dataset `ring`, con rangos más amplios en sus 20 atributos reales.

En los datasets de clasificación `bands`, `mushroom`, `penbased`, `segment`, `thyroid` y `vehicle` se descubren más de 100 millones de evidencias agrupadas en catálogos robustos. En general, el algoritmo $\mathcal{ACDC}$ descubre un gran número de evidencias robustas en cada uno de los datasets analizados.

6.3.3. Evidencias robustas de los sub-catálogos

La quinta columna de la tabla 6.2 muestra el número de evidencias que hay en el catálogo robusto con menos filas del conjunto $\mathcal{CCR}_{\mathcal{D}}$. En algunos casos no se reduce el número de evidencias robustas del catálogo original, como ocurre con `tic-tac-toe` y `twonorm`. Todos los catálogos robustos descubiertos contienen el mismo número de registros que el catálogo original, utilizando menos atributos.

En el extremo opuesto, hay datasets de clasificación en los que si se suprimen atributos redundantes, se reduce significativamente el número de caracterizaciones robustas conocidas en el experimento de clasificación. Si se quiere trabajar con ordenadores que tengan pocos recursos, es preferible utilizar matrices con un número menor de filas, sabiendo que la información que contienen sobre clasificación es la misma que la que contiene el dataset de clasificación original.

monk-2 Este dataset de clasificación contiene 432 evidencias y 6 atributos, 3 de ellos redundantes. No contiene duplicados ni incertidumbre, el dataset original es un catálogo robusto. Al aplicar el algoritmo $\mathcal{ACDC}$ se

descubre que contiene, entre otros 8, un catálogo robusto formado por 36 evidencias y 3 atributos, una vez eliminados los atributos 1, 3 y 6.

mushroom Ya se descubrió en la sección 6.1 que con sólo eliminar el atributo A_9 se necesitan únicamente 1,662 evidencias robustas de las 5,644 que contiene el catálogo robusto original. Al aplicar el algoritmo de forma no supervisada descubrimos que su $\mathcal{CCR}_{\mathcal{D}}$ contiene un catálogo robusto con tan solo 22 evidencias.

shuttle Se obtiene un catálogo robusto con sólo el 11 % de las evidencias robustas que tiene el dataset de clasificación original. Teniendo en cuenta que el dataset original ya es un catálogo robusto, la reducción de datos a analizar es muy significativa.

6.3.4. Reducción de atributos

Todos los catálogos robustos descubiertos por el algoritmo $\mathcal{ACDC}$ tienen al menos un atributo menos que el catálogo que se obtiene del dataset de clasificación original. En algunos casos, se descubre que son muy pocos los atributos necesarios para hacer la misma clasificación que se haría con el dataset original.

En el experimento de clasificación `hepatitis`, utilizando sólo 2 atributos se obtiene la misma clasificación que si se usan los 19 atributos recogidos en el dataset de clasificación original. En `mushroom` sólo se necesitan 3 atributos para clasificar con un 100 % de confianza cualquiera de las caracterizaciones de su dataset original, que gestiona 22 atributos. En `ring` se puede trabajar con caracterizaciones que sólo tengan 2 de los 20 atributos del experimento. En `vowel` y `wine`, de los 13 atributos que tienen ambos experimentos, se obtiene la misma clasificación si sólo se utilizan 2.

6.4. Datasets de grandes dimensiones

En la sección 6.1 se muestran las características de los 75 datasets de clasificación publicados en *KEEL Standard Dataset Repository* 2004. En la sección 6.3 se muestran los resultados obtenidos al ejecutar el algoritmo $\mathcal{ACDC}$ sobre una selección de 48 datasets de KEEL. No se analizan ni los datasets que no tienen atributos redundantes, ni los que tienen un número

grande de atributos redundantes. En el primer caso no se descubre nada con el algoritmo $\mathcal{ACDC}$, en el segundo caso puede haber una explosión de resultados que aborte la ejecución del algoritmo por falta de recursos.

Para ejecutar el algoritmo $\mathcal{ACDC}$, los datasets de clasificación de grandes dimensiones no son sólo aquellos que tienen muchas evidencias robustas y muchos atributos. El número de atributos redundantes que contiene es un parámetro más determinante en este aspecto.

El conjunto $\mathcal{CCR}_{\mathcal{D}}$ crece exponencialmente en función del número de atributos que pueden ser eliminados del catálogo robusto original. Según muestra la tabla 6.2, datasets como `ring`, `thyroid` o `twonorm`, con 20 o 21 atributos redundantes, contienen miles de millones de evidencias robustas. Si se lanza el proceso sin supervisar sobre datasets con mayores dimensiones, es muy probable que al cabo de unas horas o días se detenga el proceso por sobrecarga de memoria RAM.

Derrac et al. (2010) usan datasets de UCI, de los que especifican que `chess`, `movement_libras`, `satimage`, `spambase`, `splice` y `texture` son de grandes dimensiones, básicamente por tener 36, 90, 36, 57 y 60 atributos respectivamente. En la tabla 6.2 se comprueba que el experimento de clasificación `chess` contiene un dataset que puede ser analizado por completo. A pesar de tener 36 atributos, sólo 9 son redundantes. El algoritmo $\mathcal{ACDC}$ descubre, sin supervisión y en menos de 3 segundos, los 288 catálogos robustos que contiene.

En esta sección se presentan los resultados obtenidos al analizar 4 datasets de clasificación de KEEL con grandes dimensiones. El más grande de ellos respecto al tamaño del dataset en disco es `kddcup`, que ocupa 74MB. Para probar con datasets de clasificación de mayor tamaño en disco, se incorporó al estudio el dataset `kddcup99`, de UCI, que ocupa 742MB y tiene un millón de evidencias.

Para evitar problemas de desbordamiento de memoria al analizar datasets de clasificación de grandes dimensiones, se ha de limitar el número de iteraciones del algoritmo, K. Se utiliza $\mathcal{CCR}_{\mathcal{D},\mathcal{K}}$ para denotar el conjunto de catálogos robustos obtenido en la iteración K del algoritmo.

Los resultados obtenidos en este análisis se muestran en la tabla 6.3, donde la primera columna muestra el nombre del dataset. En la siguiente columna se muestra el número de evidencias robustas que tiene el catálogo robusto inicial, y en la siguiente, el número de atributos del experimento de clasificación y, entre paréntesis, el número de atributos redundantes

y el de los que son constantes. Estos valores están marcados en verde en el dataset `movement_libras`, para destacar que todos sus atributos son redundantes.

En la cuarta columna de la tabla se indica el número de iteraciones del algoritmo realizadas. En la quinta columna se indica el número de catálogos robustos que contiene habiendo suprimido un máximo de K atributos, y en la siguiente el número de evidencias robustas que contienen esos catálogos. La séptima columna informa sobre el número de evidencias robustas que tiene el catálogo robusto con menos filas del $\mathcal{CCR}_{\mathcal{D},\mathcal{K}}$. En la última columna se muestra el tiempo empleado en la ejecución completa del algoritmo.

Tabla 6.3: Ejecución supervisada del algoritmo $\mathcal{ACDC}$

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{A}\rvert$	K	$\lvert\mathcal{CCR}_{\mathcal{D},\mathcal{K}}\rvert$	**Evidencias robustas**	$\min \lvert\mathcal{D}^{-\mathcal{I}}\rvert$	**Tiempo (sg.)**
census	139,303	41(38)(2)	1	37	5,152,838	138,495	30
			2	664	92,447,127	138,283	271
			3	7,695	1,071,052,509	137,838	2,966
			4	64,695	9,002,142,142	137,655	23,343
coil2000	8,261	85(76)(-)	1	77	626,871	8,100	3
			2	2,915	23,729,171	8,089	65
			3	72,327	588,709,706	8,085	1,594
			4	1,322,956	10,767,206,805	8,074	28,453
kddcup	145,583	41(37)(2)	1	38	5,454,396	113,564	37
			2	701	99,168,860	104,179	300
			3	8,362	1,165,483,398	96,953	3,297
			4	72,488	9,950,832,287	91,926	27,464
kddcup99	1,074,974	41(37)(1)	1	37	39,243,116	914,464	208
			2	660	690,812,738	765,854	1,629
			3	7,564	7,814,485,719	700,146	17,156
mov_lib[5]	360	90(90)(-)	1	91	30,030	330	1
			2	4,096	1,351,680	330	2
			3	121,576	40,120,080	330	74
			4	2,676,766	883,332,780	330	1,616
			5	46,626,034	15,386,591,220	330	29,912

En la tabla se comprueba el crecimiento exponencial del número de catálogos y evidencias robustos encontrado conforme se incrementa el va-

[5] `movement_libras`

lor de K. También se observa que, conforme se suprimen atributos, los catálogos robustos descubiertos necesitan menos evidencias para describir a las caracterizaciones reducidas. Los datasets de clasificación kddcup y kddcup99 reducen notablemente el número de evidencias a gestionar si se utilizan los catálogos robustos reducidos que contienen.

El nivel K alcanzado en la tabla para cada dataset de clasificación se debe a los resultados obtenidos en cada iteración. Al aplicar nivel $K = 1$ al dataset census, se obtienen resultados en tan solo 30 segundos y sin problemas de desbordamiento de memoria, por lo que es razonable pensar que no habrá problemas si se aplica el nivel $K = 2$. En poco menos de 5 minutos, y sin problemas de desbordamiento, se obtiene un análisis más profundo, por lo que se puede aplicar el nivel $K = 3$, donde los recursos no presentan ningún problema pero empieza a ser más costosa la ejecución del algoritmo en cuanto al tiempo empleado, esta vez casi 50 minutos. Observando la secuencia, no es previsible que aparezcan problemas de desbordamiento de memoria en la siguiente iteración, y el tiempo necesario puede ser de aproximadamente diez veces la ejecución anterior. Para ejecutar el algoritmo con nivel $K = 4$ se necesitan casi 6 horas y media. El siguiente nivel, que necesitará muchos recursos de memoria y podría presentar desbordamiento de memoria, tardaría varios días en dar resultados, es preferible no plantearlo y seguir la estrategia propuesta en la sección 6.4.1.

En el resto de datasets de clasificación revisados en esta sección, se decide el nivel de parada siguiendo el mismo criterio. No se puede aplicar un análisis no supervisado más profundo a un dataset de grandes dimensiones si no se dispone de un equipo capaz de gestionar grandes colecciones de datos en tiempos aceptables.

6.4.1. Profundizando en grandes datasets

Los resultados mostrados en la tabla 6.3 son sólo una parte de los que puede obtener el algoritmo $\mathcal{ACDC}$ sobre los datasets de clasificación analizados. Se han eliminado 3, 4 o 5 atributos a los dataset, descubriendo un enorme número de catálogos robustos y de evidencias, con crecimiento exponencial en función del número de iteraciones realizado, K.

Para obtener más información, no se puede hacer un simple incremento de K. Sólo se obtendría una ejecución fallida del algoritmo, horas o días

después de haberlo iniciado. Todos los catálogos robustos $\mathcal{D}^{-\mathcal{I}}$ de $\mathcal{CCR}_{\mathcal{D},\mathcal{K}}$ tienen las mismas propiedades que el catálogo robusto inicial $\mathcal{D}$, por lo que se puede ejecutar el algoritmo $\mathcal{ACDC}$ sobre cualquiera de ellos, obteniendo su propia colección de catálogos robustos $\mathcal{CCR}_{\mathcal{D}^{-\mathcal{I}},\mathcal{K}}$.

Se ha utilizado esta estrategia con el dataset de clasificación census. Este dataset tiene 2 atributos constantes que son eliminados antes de entrar en la fase recursiva del algoritmo, por lo que de los 41 atributos del experimento de clasificación sólo se utilizan 39, siendo todos redundantes menos tres. Al eliminar hasta $K = 4$ atributos redundantes, ya aparecen más de nueve mil millones de evidencias robustas, lo que asegura una sobrecarga de recursos si no se supervisa el análisis. Se ha de ejecutar el algoritmo sobre uno de los catálogos de $\mathcal{CCR}_{\mathcal{D},4}$.

En un experimento de clasificación en el que se conozca el coste –económico, social…– de medir cada atributo, se podría seleccionar un catálogo robusto de $\mathcal{CCR}_{\mathcal{D},4}$ en el que no estén los atributos más costosos. También se puede forzar la no aparición de aquellos atributos que sea imposible adquirir por no disponer de una herramienta concreta. La tabla 6.3 muestra que se han descubierto 64,695 catálogos robustos distintos en census, uno de ellos con 39 atributos, 37 con 38 atributos, 627 con 37 atributos….

En este caso no hay disponible información de costos, por lo que se utilizará el catálogo robusto de menores dimensiones. El catálogo robusto con menos evidencias robustas encontrado en census es

$$\mathcal{D}^{-\mathcal{I}} \,/\, \mathcal{I} = \{16, 23, 24, 32, 33, 41\}$$

Contiene 137,655 evidencias y 35 atributos. El dataset de clasificación original tiene 1 atributo esencial, por lo que este catálogo reducido tendrá, al menos, un atributo esencial. Es posible, pues, que $\mathcal{D}^{-\mathcal{I}}$ tenga 34 atributos redundantes, por lo que no es oportuna la ejecución del algoritmo sin supervisión. Se inicia un nuevo análisis supervisado, partiendo de este catálogo robusto y obteniendo su $\mathcal{CCR}_{\mathcal{D}^{-\mathcal{I}},\mathcal{K}}$ con niveles $K = 1 \ldots$

Los resultados obtenidos se muestran en la tabla 6.4, que tiene la misma estructura que la tabla 6.3. Se observa que este catálogo robusto tiene 6 atributos esenciales, cinco más que el dataset original. 5 de los atributos de $\mathcal{A} - \mathcal{I} = \mathcal{A} - \{16, 23, 24, 32, 33, 41\}$, que aún están en los catálogos reducidos, se podían eliminar de $\mathcal{D}$ porque la información sobre clasificación que proporcionan se podía obtener a partir de los atributos de $\mathcal{I}$. Una vez

Tabla 6.4: Colección de catálogos robustos del dataset census sin los atributos del conjunto $\mathcal{I} = \{16, 23, 24, 32, 33, 41\}$

$\lvert\mathcal{D}^{-\mathcal{I}}\rvert$	$\lvert\mathcal{A}^{-\mathcal{I}}\rvert$	K	$\lvert\mathcal{CCR}_{\mathcal{D}^{-\mathcal{I}}}\rvert$	**Evidencias robustas**	mín $\lvert\mathcal{D}^{-\mathcal{I}-\mathcal{J}}\rvert$	**Tiempo (sg.)**
137,655	35(29)(-)	1	30	4,129,073	137,284	24
		2	434	59,725,266	136,988	168
		3	4,031	554,645,993	136,967	1,425
		4	27,005	3,715,177,551	136,761	9,243

eliminados los atributos de $\mathcal{I}$, estos 5 atributos son necesarios para poder hacer estimaciones basadas en evidencias robustas.

El algoritmo $\mathcal{ACDC}$ descubre que $\mathcal{D}^{-16-23-24-32-33-41}$ contiene 27,005 catálogos robustos en su $\mathcal{CCR}_{\mathcal{D}^{-\mathcal{I}},4}$, con más de tres mil millones de evidencias robustas. Uno de los catálogos robustos descubierto tiene 136,761 evidencias y 31 atributos. Se trata de

$$\mathcal{D}^{-\mathcal{I}-\mathcal{J}} \,/\, \mathcal{I} = \{16, 23, 24, 32, 33, 41\} \,\wedge\, \mathcal{J} = \{12, 35, 36, 37\}$$

Se sabe que este catálogo robusto tiene como mínimo 6 atributos esenciales, los que hereda de $\mathcal{D}^{-\mathcal{I}}$. Puede contener hasta 31-6=25 atributos redundantes, por lo que se vuelve a aplicar el algoritmo $\mathcal{ACDC}$ de forma supervisada con nivel $K = 1\ldots$, descubriendo los resultados que muestra la tabla 6.5.

Tabla 6.5: Colección de catálogos robustos del dataset census sin los atributos del conjunto $\mathcal{I} = \{12, 16, 23, 24, 32, 33, 35, 36, 37, 41\}$

$\lvert\mathcal{D}^{-\mathcal{I}}\rvert$	$\lvert\mathcal{A}^{-\mathcal{I}}\rvert$	K	$\lvert\mathcal{CCR}_{\mathcal{D}^{-\mathcal{I}}}\rvert$	**Evidencias robustas**	mín $\lvert\mathcal{D}^{-\mathcal{I}-\mathcal{J}}\rvert$	**Tiempo (sg.)**
136,761	31(25)(-)	1	26	3,555,699	136,735	19
		2	324	44,308,317	136,693	134
		3	2,575	352,131,960	136,679	865
		4	14,655	2,004,017,232	136,666	4,813
		5	63,596	8,696,257,039	136,652	21.571

Ahora se sabe que $\mathcal{D}^{-\mathcal{I}} = \mathcal{D}^{-12-16-23-24-32-33-35-36-37-41}$ tiene 25 atributos redundantes. Si se suprimen de $\mathcal{D}^{-\mathcal{I}}$ los atributos del conjunto

$\mathcal{J} = \{7, 27, 28, 40\}$, se obtiene un catálogo robusto con 136,652 evidencias y 26 atributos. Este catálogo tiene como mínimo 6 atributos esenciales, por lo que tendrá como máximo 20 atributos redundantes. Es previsible que no haya problemas de desbordamiento de memoria si se aplica el algoritmo $\mathcal{ACDC}$ sin supervisión sobre el catálogo

$$\mathcal{D}^{-7-12-16-23-24-27-28-31-32-33-35-36-37-40-41}$$

La tabla 6.6 muestra los resultados obtenidos tras este último análisis. Cabe destacar que estos resultados son sólo una parte de los catálogos robustos que contiene el dataset de clasificación original, census. Son los catálogos de un experimento de clasificación reducido en el que intervienen sólo 26 de los 41 atributos originales del experimento.

Tabla 6.6: Colección de catálogos robustos del dataset census sin los atributos del conjunto $\mathcal{I} = \{7, 12, 16, 23, 24, 27, 28, 31, 32, 33, 35, 36, 37, 40, 41\}$

$\lvert\mathcal{D}^{-\mathcal{I}}\rvert$	$\lvert\mathcal{A}^{-\mathcal{I}}\rvert$	K	$\lvert\mathcal{CCR}_{\mathcal{D}-\mathcal{I}}\rvert$	**Evidencias robustas**	$\min \lvert\mathcal{D}^{-\mathcal{I}-\mathcal{J}}\rvert$	**Tiempo (sg.)**
136,6521	26(17)(-)	∞	91,264	12,467,899,776	136,524	37,563

Los resultados obtenidos no son necesariamente los óptimos para analizar el dataset de clasificación census.

1. Tras el primer análisis, en el que se han eliminado hasta 4 de sus atributos redundantes (tabla 6.3), se ha seleccionado el catálogo robusto con menos evidencias del conjunto $\mathcal{CCR}_{\mathcal{D},4}$. Utilizando otro criterio se podría haber seleccionado cualquiera de los 64,695 catálogos robustos descubiertos hasta el momento.

2. Se ha vuelto a aplicar el algoritmo$\mathcal{ACDC}$ sobre el catálogo robusto seleccionado en el paso anterior, eliminando hasta 5 atributos redundantes (tabla 6.5). De nuevo, se selecciona el catálogo robusto reducido con menos evidencias, que tiene 26 atributos.

3. Este último catálogo robusto tiene, a lo sumo, 20 atributos redundantes. Son muchos, pero los resultados de la sección 6.3 demuestran que el algoritmo $\mathcal{ACDC}$ puede ser aplicado sin supervisión sobre datasets de clasificación de estas dimensiones. Se descubre que

todavía tiene 17 los atributos redundantes, y que basta con usar 12 atributos para obtener la misma clasificación que obtendríamos con los 41 atributos del experimento de clasificación original.

Si en los pasos 1 y 2 hubiera seleccionado otro catálogo robusto los resultados serían distintos. El número total de catálogos robustos que contiene un dataset de clasificación de grandes dimensiones, si el experimento de clasificación está sobredimensionado, es tan grande que no pueden obtenerse resultados óptimos si no se utilizan técnicas más avanzadas. Para elegir en cada uno de los pasos intermedios un catálogo que proporcione mejores resultados, se podría utilizar algún modelo de *Investigación Operativa*, como se propone en la sección 7.2.

Tabla 6.7: Características de los datasets de clasificación estándar de KEEL

#	dataset	Tamaño	$\mathcal{N}$(r)(c)	$rg(\mathcal{A})$	$\|\mathcal{C}\|$	$\|dataset\|$	$\|\mathcal{D}_0\|$	Inc.	(1)	(2)	t (sg)
1	abalone	224.58KB	8(8)(-)	6,047	28	4,174	4,174	-	A_1 (6,044 / 4,174)	A_5 (3,619 / 4,174)	0.0845
2	adult	5.23MB	14(7)(-)	27,243	2	45,222	45,170	5	A_{14} (27,202 / 45,160)	A_{12} (27,146 / 45,162)	0.9729
3	appendicitis	4.73KB	7(7)(-)	530	2	106	106	-	A_1 (456 / 106)	A_7 (431 / 106)	0.0024
4	australian	30.43KB	14(13)(-)	1,143	2	690	690	-	A_3 (942 / 689)	A_{14} (903 / 690)	0.0142
5	automobile	22.98KB	25(25)(1)	795	6	159	159	-	A_{25} (650 / 155)	A_{25} (650 / 155)	0.0060
6	balance	14.35KB	4(0)(-)	20	3	625	625	-			0.0009
7	banana	86.72KB	2(0)(-)	4,045	2	5,300	5,291	1			0.0079
8	bands	32.82KB	19(19)(-)	719	2	365	365	-	A_1 (692 / 365)	A_{11} (614 / 365)	0.0095
9	breast	18.38KB	9(2)(-)	41	2	277	257	6	A_4 (34 / 247)	A_4 (34 / 247)	0.0017
10	bupa	13.19KB	6(6)(-)	328	2	345	341	-	A_1 (302 / 341)	A_5 (234 / 341)	0.0029
11	car	51.00KB	6(0)(-)	21	4	1,728	1,728	-			0.0032
12	census	65.87MB	41(36)(2)	69,224	2	142,521	139,303	-	A_{32} (69,219 / 138,495)	A_6 (68,193 / 139,303)	16.8982
13	chess	241.32KB	36(9)(-)	73	2	3,196	3,196	-	A_{11} (71 / 3,091)	A_2 (71 / 3,160)	0.0815
14	cleveland	18.26KB	13(13)(-)	397	5	297	297	-	A_1 (356 / 297)	A_5 (245 / 297)	0.0055
15	coil2000	1.63MB	85(76)(-)	650	2	9,822	8,261	119	A_{54} (644 / 8,100)	A_5 (640 / 8,142)	1.3054
16	connect-4	5.76MB	42(42)(-)	126	3	67,557	67,557	-	A_1 (123 / 67,557)	A_1 (123 / 67,557)	5.7513
17	contraceptive	30.82KB	9(0)(-)	71	3	1,473	1,358	62			0.0034
18	crx	34.77KB	15(15)(-)	1,116	2	653	653	-	A_3 (917 / 652)	A_2 (776 / 653)	0.0121
19	dermatology	26.73KB	34(34)(-)	189	6	358	358	-	A_{34} (129 / 356)	A_{34} (129 / 356)	0.0144
20	ecoli	12.34KB	7(7)(-)	357	8	336	336	-	A_2 (294 / 335)	A_6 (277 / 336)	0.0032
21	fars	56.40MB	29(9)(-)	752	8	100,968	90,097	2,696	A_{29} (734 / 87,321)	A_{21} (679 / 87,401)	3.7714
22	flare	25.50KB	11(1)(-)	41	6	1,066	287	56	A_7 (39 / 213)	A_7 (39 / 213)	0.0034
23	german	99.63KB	20(20)(-)	1,075	2	1,000	1,000	-	A_5 (154 / 998)	A_5 (154 / 998)	0.0319
24	glass	16.80KB	9(9)(-)	905	6	214	213	-	A_5 (772 / 212)	A_1 (756 / 213)	0.0032
25	haberman	6.53KB	3(0)(-)	92	2	306	283	6			0.0007
26	hayes-roth	1.98KB	4(1)(-)	15	3	160	84	9	A_1 (12 / 55)	A_1 (12 / 55)	0.0003
27	heart	10.58KB	13(13)(-)	380	2	270	270	-	A_1 (339 / 270)	A_5 (236 / 270)	0.0049
28	hepatitis	4.71KB	19(19)(-)	272	2	80	80	-	A_1 (232 / 80)	A_{15} (213 / 80)	0.0021
29	housevotes	10.49KB	16(10)(-)	32	2	232	160	-	A_{10} (30 / 144)	A_4 (30 / 160)	0.0028
30	ionosphere	76.48KB	33(33)(-)	7,279	2	351	350	-	A_{27} (7,031 / 348)	A_{17} (7,020 / 350)	0.0289
31	iris	5.49KB	4(4)(-)	123	3	150	147	-	A_1 (88 / 142)	A_3 (80 / 143)	0.0010
32	kddcup	71.89MB	41(37)(2)	19,445	23	494,020	145,583	2	A_{23} (18,955 / 113,564)	A_6 (8,720 / 138,690)	13.9529

Características de los datasets de clasificación estándar de KEEL (cont.)

#	dataset	Tamaño	$\mathcal{N}$(r)(c)	$rg(\mathcal{A})$	$\lvert\mathcal{C}\rvert$	$\lvert dataset\rvert$	$\lvert\mathcal{D}_0\rvert$	Inc.	(1)	(2)	t (sg)
32b	kddcup99[6]	708.18MB	41(37)(1)	41,894	23	4,898,431	1,074,974	17	A_{23} (41,382 / 914,464)	A_6 (20,401 / 950,794)	208.8805
33	kr-vs-k	547.33KB	6(0)(-)	40	18	28,056	28,056	-			0.0575
34	led7digit	32.57KB	7(0)(-)	14	10	500	85	37			0.0013
35	letter	716.18KB	16(13)(-)	256	26	20,000	18,668	-	A_2 (240 / 18,177)	A_1 (240 / 18,498)	0.5173
36	lymphography	15.43KB	18(18)(-)	59	4	148	148	-	A_{15} (56 / 147)	A_{14} (51 / 148)	0.0050
37	magic	1.43MB	10(10)(-)	147,107	2	19,020	18,905	-	A_1 (128,464 / 18,905)	A_6 (128,403 / 18,905)	0.5491
38	mammographic	10.81KB	5(0)(-)	92	2	830	519	45			0.0014
39	marketing	188.72KB	13(0)(-)	75	9	6,876	5,631	429			0.0173
40	monk-2	9.13KB	6(3)(-)	17	2	432	432	-	A_1 (14 / 144)	A_1 (14 / 144)	0.0017
41	movement_libras	257.76KB	90(90)(-)	18,180	15	360	330	-	A_1 (17,966 / 330)	A_{89} (17,945 / 330)	0.0771
42	mushroom	254.84KB	22(22)(1)	98	2	5,644	5,644	-	A_9 (89 / 1,662)	A_9 (89 / 1,662)	0.1699
43	newthyroid	5.07KB	5(5)(-)	334	3	215	215	-	A_1 (279 / 215)	A_2 (234 / 215)	0.0016
44	nursery	1.12MB	8(0)(-)	27	5	12,960	12,960	-			0.0365
45	optdigits	817.18KB	64(64)(2)	914	10	5,620	5,620	-	A_2 (905 / 5,620)	A_3 (897 / 5,620)	0.6831
46	page-blocks	241.03KB	10(10)(-)	9,094	5	5,472	5,393	13	A_1 (8,990 / 5,380)	A_9 (7,376 / 5,380)	0.0900
47	penbased	710.16KB	16(16)(-)	1,608	10	10,992	10,992	-	A_1 (1,507 / 10,992)	A_1 (1,507 / 10,992)	0.3276
48	phoneme	170.88KB	5(5)(-)	11,178	2	5,404	5,349	-	A_1 (9,109 / 5,326)	A_3 (8,659 / 5,341)	0.0588
49	pima	34.13KB	8(8)(-)	1,254	2	768	768	-	A_1 (1,237 / 768)	A_7 (737 / 768)	0.0099
50	poker	23.01MB	10(0)(-)	85	10	1,025,009	1,022,770	-			3.3446
51	post-operative	4.56KB	8(0)(-)	23	3	87	71	6			0.0010
52	ring	1.11MB	20(20)(-)	75,126	2	7,400	7,400	-	A_1 (71,386 / 7,400)	A_3 (71,319 / 7,400)	0.4051
53	saheart	21.04KB	9(9)(-)	1,767	2	462	462	-	A_1 (1,705 / 462)	A_4 (1,359 / 462)	0.0074
54	satimage	978.71KB	36(36)(-)	2,806	6	6,435	6,435	-	A_1 (2,755 / 6,435)	A_{12} (2,702 / 6,435)	0.4429
55	segment	410.58KB	19(17)(1)	14,929	7	2,310	2,086	-	A_2 (14,691 / 2,077)	A_{19} (12,992 / 2,086)	0.0793
56	shuttle	1.47MB	9(8)(-)	1,109	7	57,999	57,999	-	A_6 (810 / 27,480)	A_6 (810 / 27,480)	0.8551
57	sonar	75.87KB	60(60)(-)	8,208	2	208	208	-	A_1 (8,143 / 208)	A_{31} (8,018 / 208)	0.0246
58	spambase	1.33MB	57(50)(-)	15,091	2	4,597	4,203	3	A_{50} (14,450 / 4,189)	A_{55} (12,930 / 4,198)	0.5399
59	spectfheart	36.57KB	44(44)(-)	1,887	2	267	267	-	A_1 (1,848 / 267)	A_{42} (1,826 / 267)	0.0174
60	splice	574.01KB	60(60)(-)	287	3	3,190	3,005	1	A_1 (282 / 3,002)	A_{35} (281 / 3,003)	0.3150
61	tae	2.33KB	5(2)(-)	101	3	151	106	4	A_4 (99 / 101)	A_1 (99 / 102)	0.0007
62	texture	1.46MB	40(40)(-)	39,556	11	5,500	5,473	-	A_1 (38,695 / 5,473)	A_{33} (38,246 / 5,473)	0.5359

[6]Procedente de *UCI Machine Learning Repository* 2013.

Características de los datasets de clasificación estándar de KEEL (cont.)

#	dataset	Tamaño	$\mathcal{N}$(r)(c)	$rg(\mathcal{A})$	$\|\mathcal{C}\|$	$\|dataset\|$	$\|\mathcal{D}_0\|$	Inc.	(1)	(2)	t (sg)
63	thyroid	635.77KB	21(21)(-)	1,438	3	7,200	7,129	-	A_1 (1,340 / 6,980)	A_{21} (972 / 7,129)	0.2798
64	tic-tac-toe	35.10KB	9(9)(-)	27	2	958	958	-	A_1 (24 / 958)	A_1 (24 / 958)	0.0114
65	titanic	51.86KB	3(3)(-)	8	2	2,201	14	10	A_1 (4 / 2)	A_1 (4 / 2)	0.0017
66	twonorm	1.23MB	20(20)(-)	135,085	2	7,400	7,400	-	A_1 (128,337 / 7,400)	A_{10} (128,295 / 7,400)	0.4535
67	vehicle	71.11KB	18(18)(-)	1,430	4	846	846	-	A_1 (1,386 / 846)	A_{12} (1,006 / 846)	0.0216
68	vowel	72.54KB	13(13)(-)	8,106	11	990	990	-	A_1 (8,104 / 990)	A_5 (7,229 / 990)	0.0209
69	wdbc	102.75KB	30(30)(-)	8,314	2	569	569	-	A_1 (7,858 / 569)	A_{24} (7,770 / 569)	0.0337
70	wine	13.70KB	13(13)(-)	1,276	3	178	178	-	A_1 (1,150 / 178)	A_2 (1,143 / 178)	0.0049
71	winequality-red	90.35KB	11(11)(-)	1,453	6	1,599	1,359	-	A_{11} (1,388 / 1,355)	A_8 (1,017 / 1,359)	0.0293
72	winequality-white	281.17KB	11(11)(-)	2,308	7	4,898	3,961	-	A_1 (2,240 / 3,943)	A_8 (1,418 / 3,961)	0.0723
73	wisconsin	14.37KB	9(8)(-)	89	2	683	449	-	A_1 (79 / 385)	A_1 (79 / 385)	0.0058
74	yeast	73.54KB	8(7)(-)	412	10	1,484	1,453	-	A_4 (334 / 1,451)	A_1 (331 / 1,452)	0.0178
75	zoo	4.06KB	16(14)(-)	36	7	101	59	-	A_7 (34 / 49)	A_1 (34 / 58)	0.0014
TOTAL			**(63)(6)**				**37**	**19**			**54.5209**

7

Conclusiones

En este trabajo se cuestiona el uso de técnicas de minería de reglas de asociación sobre cualquier datasets de clasificación. En la investigación llevada a cabo al revisar el estado del arte sobre la cuestión, se ha descubierto que se utilizan dos tipos de datasets de clasificación: muestras representativas y catálogos. Las muestras representativas tienen información que permite hacer estimaciones estadísticas sobre la población en estudio. Un catálogo es un resumen de una de estas muestras, obtenido al suprimir sus registros duplicados.

Una muestra contiene información estructural y probabilística sobre el problema en estudio. Además de recoger las relaciones que existen entre los valores de diferentes atributos y una clase, contienen información sobre el número de veces que ocurren esas relaciones. Un catálogo contiene la misma información estructural que la muestra de la que procede, pero no tiene ningún tipo de información probabilística.

Las métricas usadas en minería de reglas de asociación se basan en las propiedades estadísticas de las muestras representativas. Su adaptación al problema de clasificación genera una disciplina, la minería de reglas de clasificación asociativa, que mejora los resultados obtenidos por la minería de reglas de clasificación clásica, según las investigaciones mostradas en el capítulo 2. Sin embargo, al aplicar técnicas de minería de reglas de clasificación asociativa sobre catálogos, se utilizan propiedades que no poseen este tipo de datasets de clasificación.

En este trabajo investigan las propiedades que tienen los catálogos, descubriendo una metodología que permite obtener, a partir de un catálogo, otros catálogos reducidos con la misma información estructural sobre la población en estudio.

Al ser un catálogo un resumen de una muestra, esta metodología se puede aplicar sobre ambos tipos de datasets de clasificación. Los resultados experimentales obtenidos muestran la utilidad de la metodología propuesta.

7.1. Resultados de la investigación

El objetivo buscado en todo problema de clasificación es determinar a qué clase pertenece un individuo del que se conocen hasta $\mathcal{N}$ atributos. Para llevarlo a cabo desde la perspectiva de la minería de datos, se obtiene el valor de $\mathcal{N}$ atributos sobre individuos ya clasificados, guardando en un dataset de clasificación toda la información recogida. Un dataset de clasificación es, pues, un conjunto de registros formados por $\mathcal{N}+1$ valores, el valor observado en cada uno de los atributos y la clase a la que pertenece el individuo registrado, pudiendo haber valores desconocidos en los atributos.

La minería de reglas de asociación ha demostrado gran eficiencia al trabajar con grandes bases de datos. Una regla de asociación es una relación entre dos o más valores de una colección de datos, relación que se adaptan sin esfuerzo al problema de clasificación: "*Si un individuo tiene ciertos atributos, entonces pertenece a cierta clase*". La minería de reglas de clasificación asociativa se basa en el descubrimiento de estas reglas.

Como en todo problema de minería de datos, el número de resultados obtenidos al analizar cualquier colección de datos puede ser muy grande, por lo que se renuncia a obtener toda la información escondida en los datos obtenidos. Al utilizar técnicas de minería de reglas de asociación, basadas en las propiedades estadísticas de las muestras que se analizan, se renuncia a obtener información sobre los valores menos frecuentes de los datasets de clasificación en estudio, utilizando el criterio de soporte mínimo. Según este criterio, la mejor información está en los datos que aparecen con mayor frecuencia en el dataset, por lo que no se analizan los datos con soporte menor al fijado como mínimo.

En una muestra representativa, la frecuencia de cualquier dato o regla es un buen estimador de su frecuencia poblacional. En esta tesis se demuestra que esta afirmación no es correcta cuando se analiza un catálogo. Al analizar datasets de clasificación, se ha de saber si el dataset contiene una muestra representativa o un catálogo. En el primero caso, los algoritmos de minería de reglas de clasificación asociativa proporcionan los mejores resultados desde el punto de vista estadístico. En el segundo no.

Otro criterio usado en minería de reglas de asociación para determinar cuáles son las mejores reglas de un dataset es el de confianza. La confianza de una regla de asociación representa el porcentaje de veces que se verifica en el dataset analizado. Es una medida que depende del soporte de la regla, entonces, si no se debe utilizar el soporte para analizar catálogos, no parece tener sentido utilizar la confianza para determinar qué reglas son mejores. Sin embargo, cuando una regla se verifica en todos los registros de un catálogo, no importa cuál es su soporte.

En todos los datasets de clasificación analizados durante esta investigación se han descubierto millones de reglas de asociación con un 100 % de confianza. En el problema de clasificación, si la regla "*Si un individuo tiene ciertos atributos, entonces pertenece a cierta clase*" tiene un 100 % de confianza, se concluye que no hay incertidumbre a la hora de clasificar a los individuos que tienen los atributos en cuestión.

Estudiando las características y propiedades de los catálogos, en este trabajo se propone dividirlos en tres conjuntos de datos homogéneos: registros con datos desconocidos, registros con incertidumbre y registros sin incertidumbre.

El análisis del conjunto de registros sin incertidumbre, proporciona todas las reglas de asociación sin incertidumbre que tiene el dataset. Para no tener problemas de desbordamiento de memoria cuando son muchas las reglas que contiene el dataset, se aprovecha la estructura de los datos en un problema de clasificación. No se guardan una a una todas las reglas de asociación descubiertas, se guardan agrupadas en catálogos reducidos.

Tras revisar los experimentos realizados en esta investigación, se puede concluir que la metodología propuesta ofrece gran cantidad de información sobre la estructura de la población que se quiere clasificar. Esta información permite al investigador extraer, de los datasets de clasificación que analiza, conocimiento de mejor calidad, lo que le permite hacer un análisis de clasificación más completo.

El principal objetivo de este trabajo de investigación, ser capaces de extraer información sobre las mejores reglas de asociación que contiene un dataset de clasificación sin utilizar criterios de frecuencia estadística ni eliminar los datos poco frecuentes, se ha cumplido.

La formalización teórica de la metodología propuesta permite ampliar el alcance de esta investigación a datasets de clasificación de mayores dimensiones, utilizando para ello librerías especializadas en el tratamiento de matrices de grandes dimensiones y tecnología capaz de manipular grandes datasets. Esto nos permite afirmar que se ha cumplido el segundo objetivo del presente trabajo.

Los muchos experimentos realizados se han llevado a cabo con una colección completa y heterogénea de datasets que, no siendo extremadamente grandes, no se podían analizar por completo desde la perspectiva clásica de ARM, con lo que se cumple el último objetivo de esta investigación.

7.1.1. Publicaciones

Los resultados obtenidos en esta investigación se han publicado en las siguientes ponencias y artículos.

- Botella, F., Enrique Lazcorreta, Antonio Fernández-Caballero y Pascual González. "*Mejora de la usabilidad y la adaptabilidad mediante técnicas de minería de uso web*". En: Actas del VI Congreso Internacional Interacción Persona-Ordenador (Interacción'05). 19. AIPO'05 (CEDI'05). Thomson, págs. 299-306. 2005.

- Botella, F., Enrique Lazcorreta, Antonio Fernández-Caballero y Pascual González. "*Personalization through Inferring UserNavigation Maps from Web Log Files*". En: Proc. of the 11th International Conference on Human-Computer Interaction. 20. Las Vegas, Nevada, EEUU. 2005.

- Botella, F., Enrique Lazcorreta, Antonio Fernández-Caballero, Pascual González, José A. Gallud y Alejandro Bia. "*Selecting the Best Tailored Algorithm for Personalizing a Web Site*". En: Proceedings of the 12th International Conference on Human-Computer Interaction (HCII'07). Beijing (China). 2007.

- Lazcorreta Puigmartí, Enrique, F. Botella y Antonio Fernández-Caballero '"*Towards personalized recommendation by two-step modified Apriori data mining algorithm*'". En: Expert Systems with Applications, 35.3. 69, págs. 1422-1429. 2008.
- Lazcorreta Puigmartí, Enrique, F. Botella y Antonio Fernández-Caballero "*Reglas de Oportunidad: mejorando las recomendaciones web*". En: Actas del X Congreso Internacional Interacción Persona-Ordenador (Interacción'09). 70 71. 2009.
- Lazcorreta Puigmartí, Enrique, F. Botella y Antonio Fernández-Caballero. "*Recomendaciones en sistemas web mediante el estudio de ítems raros en transacciones*". En: Actas del XI Congreso Internacional Interacción Persona-Ordenador (Interacción'10). Vol. 2. 72 73, págs. 385-388. 2010.
- Lazcorreta Puigmartí, Enrique, F. Botella y Antonio Fernández-Caballero. "*Efficient Analysis of Transactions to Improve Web Recommendations*". En: Proceedings of the 13th International Conference on Interacción Persona-Ordenador (Interacción'12). En: ACM International Conference Proceeding Series. 48. 74 75. New York, NY, USA: ACM. 2012.

7.2. Trabajo futuro

El campo abierto por esta investigación es muy amplio. Se podría:

- Utilizar modelos de investigación operativa para determinar qué catálogos robustos minimizan funciones de coste en cierto experimento de clasificación.
- En datasets de clasificación con incertidumbre se podría buscar, también con investigación operativa, el conjunto de atributos a medir que maximicen la fiabilidad de las clasificaciones obtenidas.
- Añadir la capacidad de trabajar con catálogos robustos extraídos de muestras representativas. Se trata de utilizar la métrica soporte para poder estimar con qué frecuencia ocurre en la población en estudio el fenómeno que estamos describiendo.

- Utilizar computación en paralelo para mejorar la eficiencia del algoritmo $\mathcal{ACDC}$. La generación de de la $\mathcal{CCR}_{\mathcal{D}}$ de cada atributo redundante de un dataset de clasificación puede ejecutarse en un hilo totalmente independiente del descubrimiento de la $\mathcal{CCR}_{\mathcal{D}}$ del resto de atributos redundantes. La paralelización del algoritmo es inmediata, sólo hay que tener una máquina con suficientes procesadores y memoria RAM exclusiva para obtener los beneficios de esta transformación.

- Crear un sistema experto que emule el proceso humano de toma de decisiones. Combinando las primeras ideas de trabajo futuro, en que se introduce la investigación operativa para estudiar múltiples problemas de optimización, con la gestión de una base de datos que recoja nuevas muestras e incorpore en los datasets de clasificación el nuevo conocimiento adquirido, aumentará la información que posee el sistema sobre el problema de clasificación en estudio, los catálogos $\mathcal{D}$, $\mathcal{D}_{\mathcal{E}}$, $\mathcal{D}_{?}$ y sus derivados. Con esta información actualizada se puede clasificar mediante el sistema experto a cualquier nuevo individuo desde el momento en que se obtengan las primeras reglas de clasificación asociativa con un 100 % de confianza.

Bibliografía

Agrawal, Rakesh, Tomasz Imielinski y Arun N. Swami (1993a). «Database Mining: A Performance Perspective». En: *IEEE Transactions on Knowledge and Data Engineering* 5. , págs. 914-925 (vid. págs. 16, 56, 59, 61, 63, 98, 99, 102).

— (1993b). «Mining Association Rules between Sets of Items in Large Databases». En: *Proc. of the 1993 ACM SIGMOD International Conference on Management of data*. Ed. por P. Bunemann y S. Jajodia. . Washington, D.C., United States, págs. 207-216 (vid. págs. 14, 18, 25, 29, 31, 35, 51, 63).

Agrawal, Rakesh, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen y A. Inkeri Verkamo (1996). «Advances in knowledge discovery and data mining». En: ed. por Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth y Ramasamy Uthurusamy. . Menlo Park, CA, USA: American Association for Artificial Intelligence. Cap. Fast discovery of association rules, págs. 307-328 (vid. pág. 39).

Agrawal, Rakesh y John C. Shafer (1996). «Parallel Mining of Association Rules». En: *IEEE Trans. on Knowl. and Data Eng.* 8.6. , págs. 962-969 (vid. pág. 43).

Agrawal, Rakesh y Ramakrishnan Srikant (1994). «Fast Algorithms for Mining Association Rules». En: *Proc. of the 20th Very Large Data Bases Conference*. Ed. por Jorge B. Bocca, Matthias Jarke y Carlo Zaniolo. .

VLDB. Morgan Kaufmann Publishers Inc., págs. 487-499 (vid. págs. 36, 37, 39, 63, 76).

Agrawal, Rakesh y Ramakrishnan Srikant (1995). «Mining Sequential Patterns». En: *Proceedings of the Eleventh International Conference on Data Engineering (ICDE '95)*. Ed. por Philip S. Yu y Arbee S. P. Chen. Taipei, Taiwan: IEEE Computer Society Press, págs. 3-14 (vid. pág. 14).

Ahmed, Shakil, Frans Coenen y Paul Leng (2006). «Tree-based partitioning of date for association rule mining». En: *Knowledge and Information Systems* 10 (3). 10.1007/s10115-006-0010-1, págs. 315-331 (vid. pág. 52).

Ahn, Kwang-Il y Jae-Yearn Kim (2004). «Efficient Mining of Frequent Itemsets and a Measure of Interest for Association Rule Mining». En: *Journal of Information & Knowledge Management* 03.03., págs. 245-257 (vid. pág. 66).

Bayardo, Roberto J. (1998). «Efficiently mining long patterns from databases». En: *Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data*., págs. 85-93 (vid. pág. 47).

Bodon, Ferenc (2003). «A fast APRIORI implementation». En: *Proceedings of the 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI'03)*. Vol. 90. (vid. pág. 72).

— (2004). «Surprising results of trie-based FIM algorithms». En: *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'04)*. Ed. por Bart Goethals, Mohammed Javeed Zaki y Roberto Bayardo. Vol. 126. CEUR Workshop Proceedings. Brighton, UK (vid. pág. 72).

— (2005). «A trie-based APRIORI implementation for mining frequent item sequences». En: *Proceedings of the 1st International Workshop on Open Source Sata Mining (OSDM)*. Chicago, Illinois: ACM, págs. 56-65 (vid. pág. 72).

— (2006). *A Survey on Frequent Itemset Mining*. (vid. pág. 72).

Borgelt, Christian (2004). «Efficient implementations of Apriori and Eclat». En: *Proc. of the Workshop on Frequent Itemset Mining Implementations*. . FIMI'04 (vid. págs. 62, 99, 110).

Borges, José y Mark Levene (1999). «Data Mining of User Navigation Patterns». En: *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling - WEBKDD* 1836. , págs. 92-11 (vid. pág. 15).

Botella, F., Enrique Lazcorreta, Antonio Fernández-Caballero y Pascual González (2005a). «Mejora de la usabilidad y la adaptabilidad mediante técnicas de minería de uso web». En: *Actas del VI Congreso Internacional Interacción Persona-Ordenador (Interacción'05)*. . AIPO'05 (CEDI'05). Thomson, págs. 299-306 (vid. págs. 12, 37).

— (2005b). «Personalization through Inferring User Navigation Maps from Web Log Files». En: *Proc. of the 11th International Conference on Human-Computer Interaction*. . Las Vegas, Nevada, EEUU (vid. pág. 12).

Brin, Sergey, Rajeev Motwani, Jeffrey D. Ullman y Shalom Tsur (1997). «Dynamic Itemset Counting and Implication Rules for Market Basket Data». En: *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. Ed. por Joan Peckham. . ACM Press, págs. 255-264 (vid. págs. 28, 44).

Burdick, Douglas, Manuel Calimlim, Jason Flannick, Johannes Gehrke y Tomi Yiu (2005). «MAFIA: A Maximal Frequent Itemset Algorithm». En: *IEEE Trans. on Knowl. and Data Eng.* 17.11, págs. 1490-1504 (vid. pág. 50).

C++ Standard Library (libc++ 5.0) (2011). URL: `http://libcxx.llvm.org` (vid. pág. 147).

C++ Standard Template Library (STL) (2011). URL: `https://libcxx.llvm.org` (vid. págs. 108, 201).

Calders, Toon y Bart Goethals (2006). «Non-derivable itemset mining». En: *Data Mining and Knowledge Discovery* 14.1. , págs. 171-206 (vid. pág. 52).

Carmona, C.J., S. Ramírez-Gallego, F. Torres, E. Bernal, M.J. del Jesús y Salvador García (2012). «Web usage mining to improve the design of an e-commerce website: OrOliveSur.com». En: *Expert Systems with Applications* 39. , págs. 11243-11249 (vid. pág. 12).

Chen, Xin y Xiaodong Zhang (2003). «A Popularity-Based Prediction Model for Web Prefetching». En: *IEEE Computer*. , págs. 63-70 (vid. pág. 16).

Cheung, William y Osmar R. Zaiane (2003). *Incremental Mining of Frequent Patterns Without Candidate Generation Or Support*. (vid. pág. 51).

Coenen, Frans (2003). *The LUCS-KDD discretised/normalised ARM and CARM Data Library*. URL: `http://www.csc.liv.ac.uk/~frans/KDD/Software/` (visitado) (vid. pág. 65).

Coenen, Frans y Paul Leng (2004). «An Evaluation of Approaches to Classification Rule Selection». En: *2013 IEEE 13th International Conference on Data Mining* 00.undefined, págs. 359-362 (vid. pág. 65).

— (2007). «The effect of threshold values on association rule based classification accuracy». En: *Journal of Data and Knowledge Engineering* 60.2, págs. 345-360 (vid. págs. 65, 115).

Coenen, Frans, Paul Leng y Shakil Ahmed (2004). «Data Structure for Association Rule Mining: T-Trees and P-Trees». En: *IEEE Trans. on Knowl. and Data Eng.* 16.6, págs. 774-778 (vid. pág. 50).

Derrac, Joaquín, Salvador García y Francisco Herrera (2010). «IFS-CoCo: Instance and feature selection based on cooperative coevolution with nearest neighbor rule». En: *Pattern Recognition* 43.6, págs. 2082-2105 (vid. págs. 68, 165).

Dong, Jie y Min Han (2007). «BitTableFI: An efficient mining frequent itemsets algorithm». En: *Knowledge-Based Systems* 20.4, págs. 329-335 (vid. págs. 29, 51, 52, 62, 99).

Dougherty, James, Ron Kohavi y Mehran Sahami (1995). «Supervised and unsupervised discretization of continuous features». En: *in A. Prieditis & S. Russell, eds, Machine Learning: Proceedings of the Twelfth International Conference*. Morgan Kaufmann, págs. 194-202 (vid. pág. 67).

Etzioni, Oren (1996). «The World-Wide Web: Quagmire or Gold Mine?» En: *Communications of the ACM* 39.11, págs. 65-68 (vid. pág. 8).

Fayyad, Usama M. y K. B. Irani (1993). «Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning». En: *13th International Joint Conference on Uncertainly in Artificial Intelligence(IJCAI93)*, págs. 1022-1029 (vid. pág. 67).

Fayyad, Usama M., Gregory Piatetsky-Shapiro y Padhraic Smyth (1996). «From Data Mining to Knowledge Discovery in Databases». En: *AI Magazine* 17.3, págs. 37-54 (vid. págs. 7, 8).

FIM Data Repository (2004). URL: `http://fimi.ua.ac.be/data` (vid. págs. 30, 73, 104, 199).

FIMI mushroom (2004). URL: `http://fimi.ua.ac.be/data/mushroom.dat`.

Frawley, William J., Gregory Piatetsky-Shapiro y Christopher J. Matheus (1992). «Knowledge Discovery in Databases: An Overview». En: *AI Magazine* 13.3, págs. 57-70 (vid. pág. 7).

Friedman, Jerome H. (1997). «Data mining and statistics: What's the connection?» En: *Proceedings of the 29th Symposium on the Interface*. (vid. pág. 8).

Goethals, Bart (2002). «Efficient Frequent Pattern Mining». Tesis doct. School voor Informatietechnologie (vid. pág. 49).

— (2003). *Survey on Frequent Pattern Mining*. Inf. téc. HIIT Basic Research Unit; Department of Computer Science; University of Helsinki, pág. 43 (vid. págs. 13, 49).

Gouda, Karam y Mohammed Zaki (2005). «GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets». En: *Data Mining and Knowledge Discovery* 11.3, págs. 223-242 (vid. pág. 50).

Han, Jiawei y Yongjian Fu (1995). «Discovery of Multiple-Level Association Rules from Large Databases». En: *Proc. of the Int. Conf. on Very Large Data Bases*, págs. 420-431 (vid. págs. 40, 53, 54).

Han, Jiawei, Jian Pei y Yiwen Yin (2000). «Mining frequent patterns without candidate generation». En: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. SIGMOD '00. Dallas, Texas, United States: ACM, págs. 1-12 (vid. págs. 47, 51, 96).

Hernández León, Raudel (2011). «Desarrollo de Clasificadores basados en Reglas de Asociación de Clase». Tesis de mtría. Instituto Nacional de Astrofísica, Óptica y Electrónica (vid. pág. 111).

Hernández León, Raudel, Jesús A. Carrasco Ochoa, José Fco. Martínez Trinidad y José Hernández Palancar (2012). «Classification based on specific rules and inexact coverage». En: *Expert Systems with Applications* 39.12. , págs. 11203-11211 (vid. pág. 65).

Hernández León, Raudel, Jesús A. Carrasco, José Hernández Palancar y J. Fco. Martínez Trinidad (2010). *Desarrollo de Clasificadores basados en Reglas de Asociación*. Inf. téc. . Coordinación de Ciencias Computacionales (INAOE) (vid. págs. 65, 111).

Hervás-Martínez, César, Cristóbal Romero Morales y Sebastián Ventura Soto (2004a). «Comparación de medidas de evaluación de reglas de asociación». En: *Actas del Tercer Congreso de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'04)*. (vid. pág. 15).

— (2004b). «Selección de medidas de evaluación de reglas obtenidas mediante programación genética basada en gramática». En: . Tendencias de la Minería de Datos en España, Red Española de Minería de Datos. Cap. 23 (vid. pág. 15).

Hipp, Jochen, Ulrich Güntzer y Gholamreza Nakhaeizadeh (2000). «Algorithms for Association Rule Mining - A General Survey and Comparison». En: *SIGKDD Explorations* 2.1. , págs. 58-64 (vid. pág. 48).

Holsheimer, Marcel, Martin Kersten, Heikki Mannila y Hannu Toivonen (1995). «A Perspective on Databases and Data Mining». En: *In 1st Intl. Conf. Knowledge Discovery and Data Mining*. , págs. 150-155 (vid. pág. 29).

Holt, John D. y Soon M. Chung (1999). «Efficient mining of association rules in text databases». En: *Proceedings of the eighth international conference on Information and knowledge management*. CIKM '99. . Kansas City, Missouri, United States: ACM, págs. 234-242 (vid. pág. 47).

— (2002). «Mining association rules using inverted hashing and pruning». En: *Information Processing Letters* 83 (4). , págs. 211-220 (vid. pág. 49).

Hong, Tzung-Pei, Chan-Sheng Kuo y Shyue-Liang Wang (2004). «A fuzzy AprioriTid mining algorithm with reduced computational time». En: *Applied Soft Computing* 5.1. , págs. 1-10 (vid. pág. 50).

Houtsma, Maurice A. W. y Arun N. Swami (1995). «Set-Oriented Mining for Association Rules in Relational Databases». En: *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*. . Washington, DC, USA: IEEE Computer Society, págs. 25-33 (vid. págs. 29, 32).

Hu, Ya-Han y Yen-Liang Chen (2006). «Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism». En: *Decision Support Systems* 42.1. Referenciado en ESWA-Apriori2, págs. 1-24 (vid. pág. 55).

Jamali, M., F. Taghi Yareh y Masoud Rahgozar (2005). «Fast Algorithm for Generating Association Rules with Encoding Databases Layout». En: *WEC (2)*, págs. 167-170 (vid. pág. 50).

Jin, Ruoming, Scott McCallen, Yuri Breitbart, Dave Fuhry y Dong Wang (2009). «Estimating the Number of Frequent Itemsets in a Large Database». En: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. EDBT '09. . Saint Petersburg, Russia: ACM, págs. 505-516 (vid. pág. 110).

Kahramanli, Humar y Novruz Allahverdi (2009). «A new Method for Composing Classification Rules: AR+OPTBP». En: *Proceedings of The 5th International Advanced Technologies Symposium (IATS'09)*. , págs. 1-6 (vid. págs. 14, 23).

KEEL abalone (1995). URL: http://sci2s.ugr.es/keel/dataset.php?cod=52 (vid. págs. 112, 149, 150).

KEEL mushroom (1987). URL: http://sci2s.ugr.es/keel/dataset.php?cod=178.

KEEL Standard Dataset Repository (2004). J. Alcalá-Fdez et al. Universidad de Granada. URL: http://sci2s.ugr.es/keel/category.php?cat=clas (visitado) (vid. págs. 5, 30, 105, 111, 115, 147, 149, 152, 157-159, 164, 199).

Khanmohammadi, Sina y Chun-An Chou (2016). «A Gaussian mixture model based discretization algorithm for associative classification of medical data». En: *Expert Systems with Applications* 58., págs. 119-129 (vid. pág. 67).

Kiran, R. Uday y P. Krishna Reddy (2009). *An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules*. Inf. téc. 2009 IEEE Symposium on Computational Intelligence y Data Mining (IEEE CIDM 2009) (vid. págs. 55, 96, 97).

Kohavi, Ron y Ross Quinlan (1999). *Decision Tree Discovery*. (vid. págs. 13, 14).

Kouris, Ioannis N., Christos Makris y Athanasios K. Tsakalidis (2003). «An Improved Algorithm for Mining Association Rules Using Multiple Support Values». En: *FLAIRS Conference*, págs. 309-313 (vid. pág. 55).

— (2005). «Using information retrieval techniques for supporting data mining». En: *Data & Knowledge Engineering* 52.3., págs. 353-383 (vid. pág. 55).

Lee, Wenke, Salvatore J. Stolfo y Kui W. Mok (1998). «Mining Audit Data to Build Intrusion Detection Models». En: *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. AAAI Press, págs. 66-72 (vid. pág. 53).

Li, Haifeng y Hong Chen (2009). «Mining non-derivable frequent itemsets over data stream». En: *Data Knowl. Eng.* 68.5., págs. 481-498 (vid. págs. 29, 110).

Li, Haifeng y Ning Zhang (2010). «Mining maximal frequent itemsets on graphics processors». En: *Procss of the Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'10)*., págs. 1461-1464 (vid. pág. 110).

Li, Wenmin, Jiawei Han y Jian Pei (2001). «CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules». En: *Procee-*

dings of the 2001 IEEE International Conference on Data Mining (ICDM '01). ICDM '01. . Washington, DC, USA: IEEE Computer Society, págs. 369-376 (vid. pág. 65).

Lin, Dao-I y Zvi M. Kedem (2002). «Pincer-search: An efficient algorithm for discovering the maximum frequent set». En: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* 14.3. , págs. 553-566 (vid. págs. 48, 51).

Liu, Bing y Wynne Hsu (1996). «Post-Analysis of Learned Rules». En: *Proceedings of the thirteenth national conference on Artificial intelligence (AAAI'96)*. Vol. 1. , págs. 828-834 (vid. págs. 13, 15).

Liu, Bing, Wynne Hsu y Yiming Ma (1998). «Integrating Classification and Association Rule Mining». En: *Knowledge Discovery and Data Mining*. , págs. 80-86 (vid. págs. 13, 14, 23, 63, 99).

— (1999). «Mining association rules with multiple minimum supports». En: *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. . San Diego, California, United States: ACM Press, págs. 337-341 (vid. págs. 53, 54, 96, 97).

Liu, Guimei (2005). «Supporting Efficient and Scalable Frequent Pattern Mining». . Tesis doct. The Hong Kong University of Sciencie y Technology (vid. págs. 50, 52).

Liu, Guimei, Hongjun Lu y Jeffrey Xu Yu (2007). «CFP-tree: A compact disk-based structure for storing and querying frequent itemsets». En: *Information Systems* 32.2. , págs. 295-319 (vid. pág. 52).

Liu, Li, Eric Li, Yimin Zhang y Zhizhong Tang (2007). «Optimization of frequent itemset mining on multiple-core processor». En: *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*. . Vienna, Austria: VLDB Endowment, págs. 1275-1285 (vid. pág. 51).

LUCS-KDD discretised/normalised ARM and CARM Data Library (2003). Coenen, Frans. URL: `http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html` (visitado) (vid. págs. 66, 111, 200).

LUCS-KDD mushroom (2004). URL: `http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/mushroom.D90.N8124.C2.num.gz`.

Luo, Ke y Xue-Mao Zhang (2007). «An Efficient Frequent Itemset Mining Algorithm». En: *Machine Learning and Cybernetics, 2007 International Conference on*. Vol. 2. , págs. 756-761 (vid. págs. 62, 99, 110).

Malik, Kuldeep Singh y Neeraj Raheja (2013). «Improving performance of Frequent Itemset algorithm». En: *International Journal of Research in Engineering & Applied Sciences* 3.3. , págs. 168-177 (vid. pág. 110).

Mannila, Heikki, Hannu Toivonen y A. Inkeri Verkamo (1994). «Efficient algorithms for discovering association rules». En: *AAAI Workshop on Knowledge Discovery in Databases (KDD'94)*. Ed. por Usama M. Fayyad y Ramasamy Uthurusamy. . Seattle, Washington: AAAI Press, págs. 181-192 (vid. pág. 39).

Mansuri, Ronen y Pavel Berengoltz (2002). «Generalizing Association Rules Beyond Market Baskets». Impartido en la 9 lectura de `http://www.cs.tau.ac.il/~fiat/DataMine02/datamine.html`. (vid. pág. 49).

Meretakis, Dimitrios (2002). «A unified view on association and classification mining and its applications». . Tesis doct. (vid. pág. 65).

Mueller, Andreas (1995). *Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison*. Inf. téc. . University of Maryland at College Park (vid. pág. 42).

Ng, Raymond T. y Jiawei Han (1994). «Efficient and Effective Clustering Methods for Spatial Data Mining». En: *Proc. of the 20th International Conference on Very Large Data Bases (VLBD'05)*. . Los Altos, CA 94022, USA: Morgan Kaufmann Publishers, págs. 144-155 (vid. págs. 13, 16).

Nguyen, D., L.T.T. Nguyen, B. Vo y Tzung-Pei Hong (2015). «A novel method for constrained class association rule mining». En: *Information Sciences* 320. , págs. 107-125 (vid. pág. 66).

Nguyen, D., L.T.T. Nguyen, B. Vo y W. Pedrycz (2016). «Efficient mining of class association rules with the itemset constraint». En: *Knowledge-Based Systems* 103.C. , págs. 73-88 (vid. pág. 66).

Orallo, J.H., M.J.R. Quintana y C.F. Ramírez (2004). *Introducción a la Minería de Datos*. Editorial Alhambra S. A. (SP) (vid. pág. 123).

Özel, S. Ayşe y H. Altay Güvenir (2001). «An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning». En: *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'01)*. Ed. por A. Acan, I. Aybay y M. Salamah. . Gazimagusa, T.R.N.C., págs. 257-264 (vid. pág. 48).

Ozmutlu, H. Cenk, Amanda Spink y Seda Ozmutlu (2002). «Analysis of large data logs: an application of Poisson sampling on excite web que-

ries». En: *Information Processing & Management* 38.4. , págs. 473-490 (vid. pág. 14).

Palshikar, Girish K., Mandar S. Kale y Manoj M. Apte (2007). «Association rules mining using heavy itemsets». En: *Data & Knowledge Engineering* 61.1. , págs. 93-113 (vid. pág. 54).

Park, Jong Soo, Ming-Syan Chen y Philip S. Yu (1995). «An effective hash-based algorithm for mining association rules». En: *ACM SIGMOD Record* 24.2. , págs. 175-186 (vid. pág. 39).

— (1997). «Using a Hash-Based Method with Transaction Trimming for Mining Association Rules». En: *Knowledge and Data Engineering* 9.5. , págs. 813-825 (vid. pág. 39).

Pei, Jian, Jiawei Han y Laks V. S. Lakshmanan (2004). «Pushing Convertible Constraints in Frequent Itemset Mining». En: *Data Min. Knowl. Discov.* 8.3, págs. 227-252 (vid. pág. 50).

Perkowitz, Mike y Oren Etzioni (1998). «Adaptive Web Sites: Automatically Synthesizing Web Pages». En: *Proc. of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI'98)*. . Madison, Wisconsin, United States: American Association for Artificial Intelligence, págs. 727-732 (vid. pág. 14).

— (1999a). «Adaptive Web Sites: Conceptual Cluster Mining». En: *Proceeding Proceedings of the 16th International Joint Conference on Artifical intelligence (IJCAI'99)*. Vol. 1. , págs. 264-269 (vid. págs. 13, 14).

— (1999b). «Towards Adaptive Web Sites: Conceptual Framework and Case Study». En: *Computer Networks: The International Journal of Computer and Telecommunications Networking* 31.11–16. , págs. 1245-1258 (vid. pág. 14).

— (2000). «Towards Adaptive Web Sites: Conceptual Framework and Case Study». En: *Computer Networks* 31.11–16. , págs. 1245-1258 (vid. págs. 13, 14).

Piatetsky-Shapiro, Gregory (1989). «Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop». En: *AI MAGACINE*. (vid. pág. 7).

Piatetsky-Shapiro, Gregory y William J. Frawley (1991). *Knowledge Discovery in Databases*. Ed. por MIT Press. . Cambridge, MA, USA: MIT Press (vid. págs. 7, 57).

Pinho Lucas, J., S. Segrera y M.N. Moreno (2012). «Making use of associative classifiers in order to alleviate typical drawbacks in recommender

systems». English. En: *Expert Systems with Applications* 39.1. , págs. 1273-1283 (vid. pág. 66).

Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (vid. págs. 61-63).

Rácz, Balázs, Ferenc Bodon y Lars Schmidt-Thieme (2005). «On benchmarking frequent itemset mining algorithms: from measurement to analysis». En: *OSDM '05: Proceedings of the 1st international workshop on open source data mining*. . Chicago, Illinois: ACM, págs. 36-45 (vid. pág. 72).

Rai, Devashree, Kesari Verma y A. S. Thoke (2012). «Classification Algorithm based on MS Apriori for Rare Classes». En: *International Journal of Computer Applications* 48.22. , págs. 52-56 (vid. pág. 63).

Ritu, Jitender Arora (2014). «Intensification of Execution of Frequent Item-Set Algorithms». En: *International Journal of Recent Development in Engineering and Technology (IJRDET)* 2 (6). (vid. pág. 110).

Rokach, Lior y Oded Maimon (2014). *Data Mining with Decision Trees: Theory and Applications*. 2nd. Vol. 81. Series in Machine Perception and Artificial Intelligence. . World Scientific Publishing Co. Pte. Ltd. (vid. págs. 8, 13).

Rozenberg, Boris y Ehud Gudes (2006). «Association rules mining in vertically partitioned databases». En: *Data & Knowledge Engineering* 59.2. , págs. 378-396 (vid. pág. 51).

Saglam, Burcu, F. Sibel Salman, Serpil Sayin y Metin Türkay (2006). «A mixed-integer programming approach to the clustering problem with an application in customer segmentation». En: *European Journal of Operational Research* 173.3. , págs. 866-879 (vid. pág. 13).

Sahoo, Jayakrushna, Ashok Kumar Das y A. Goswami (2014). «An Algorithm for Mining High Utility Closed Itemsets and Generators». En: *Computing Research Repository (CoRR)*. (vid. pág. 110).

Savasere, Ashok, Edward Omiecinski y Shamkant B. Navathe (1995). «An Efficient Algorithm for Mining Association Rules in Large Databases». En: *The VLDB Journal*. Ed. por Umeshwar Dayal, Peter M. D. Gray y Shojiro Nishio. . Morgan Kaufmann, págs. 432-444 (vid. pág. 41).

Schafer, J. Ben, Joseph A. Konstan y John Riedl (2001). «E-Commerce Recommendation Applications». En: *Data Mining and Knowledge Discovery* 5.1/2. , págs. 115-153 (vid. pág. 14).

Srikant, Ramakrishnan y Rakesh Agrawal (1996a). «Mining Quantitative Association Rules in Large Relational Tables». En: *SIGMOD Rec.* 25.2. , págs. 1-12 (vid. pág. 67).

— (1996b). «Mining Sequential Patterns: Generalizations and Performance Improvements». En: *Proceedings of the 5th International Conference on Extending Database Technology*. , págs. 3-17 (vid. pág. 14).

— (1997). «Mining generalized association rules». En: *Future Generation Computer Systems* 13.2–3. , págs. 161-180 (vid. pág. 44).

Srikant, Ramakrishnan, Quoc Vu y Rakesh Agrawal (1997). «Mining Association Rules with Item Constraints». En: *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining*. (vid. pág. 46).

Su, Zhong, Qiang Yang, Ye Lu y Hong-Jiang Zhang (2000). «WhatNext: A Prediction System for Web Requests using N-gram Sequence Models». En: *Proc. of the First International Conference on Web Information Systems Engineering (WISE'00)*. . Washington, DC, USA: IEEE Computer Society, págs. 200-207 (vid. págs. 15, 16).

Sun, Xiaoming, Zheng Chen, Liu Wenyin y Wei-Ying Ma (2002). «Intention Modeling for Web Navigation». En: *Proc. of International Conference on Intelligent Information Technology*. . ICIIT2002. Beijing, China, págs. 107-112 (vid. págs. 15, 16).

Suzuki, Einoshin (2004). «Discovering interesting exception rules with rule pair». En: *Proceedings of the ECML/PKDD Workshop on Advances in Inductive Rule Learning*. Ed. por J. Fuernkranz. , págs. 163-178 (vid. pág. 110).

Thabtah, Fadi, Peter Cowling y Suhel Hamoud (2006). «Improving rule sorting, predictive accuracy and training time in associative classification». En: *Expert Systems with Applications* 31.2. , págs. 414-426 (vid. págs. 14, 23, 65, 110).

Toivonen, Hannu (1996). «Sampling Large Databases for Association Rules». En: *In Proc. 1996 Int. Conf. Very Large Data Bases*. Ed. por T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan y Nandlal L. Sarda. . Morgan Kaufman, págs. 134-145 (vid. pág. 42).

Tsay, Yuh-Jiuan y Ya-Wen Chang-Chien (2004). «An efficient cluster and decomposition algorithm for mining association rules». En: *Information Sciences* 160.1-4, págs. 161-171 (vid. pág. 50).

Tsay, Yuh-Jiuan y Jiunn-Yann Chiang (2005). «CBAR: an efficient method for mining association rules». En: *Knowledge-Based Systems* 18.2-3, págs. 99-105 (vid. pág. 50).

Tsay, Yuh-Jiuan, Tain-Jung Hsu y Jing-Rung Yu (2009). «FIUT: A new method for mining frequent itemsets». En: *Information Sciences* 179.11. , págs. 1724-1737 (vid. págs. 13, 52).

Tseng, Ming-Cheng y Wen-Yang Lin (2007). «Efficient mining of generalized association rules with non-uniform minimum support». En: *Data & Knowledge Engineering* 62.1, págs. 41-64 (vid. pág. 55).

UCI abalone (1995). URL: `http://archive.ics.uci.edu/ml/datasets/Abalone` (vid. págs. 112, 150).

UCI Machine Learning Repository (2013). M. Lichman, University of California, Irvine, School of Information and Computer Sciences. URL: `http://archive.ics.uci.edu/ml` (vid. págs. 30, 67, 103, 105, 109, 115, 151, 156, 173, 199).

UCI mushroom (1987). URL: `http://archive.ics.uci.edu/ml/datasets/Mushroom`.

Wang, Ke, Senqiang Zhou y Yu He (2000). «Growing Decision Trees on Support-Less Association Rules». En: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '00. . Boston, Massachusetts, USA: ACM, págs. 265-269 (vid. pág. 65).

Wang, Yanbo J., Qin Xin y Frans Coenen (2008). «Mining Efficiently Significant Classification Association Rules». English. En: *Data Mining: Foundations and Practice*. Ed. por TsauYoung Lin, Ying Xie, Anita Wasilewska y Churn-Jung Liau. Vol. 118. Studies in Computational Intelligence. . Springer Berlin Heidelberg, págs. 443-467 (vid. págs. 65, 111).

Wang, Yang y Andrew KC Wong (2003). «From association to classification: Inference using weight of evidence». En: *Knowledge and Data Engineering, IEEE Transactions on* 15.3. , págs. 764-767 (vid. pág. 110).

Weiss, Gary M. (2004). «Mining with Rarity: A Unifying Framework». En: *SIGKDD Explor. Newsl.* 6.1. , págs. 7-19 (vid. pág. 55).

Yin, Xiaoxin y Jiawei Han (2003). «CPAR: Classification based on Predictive Association Rules». En: *Procs. of the SIAM International Conference on Data Mining (SDM'03)*. , págs. 331-335 (vid. pág. 65).

Yun, Hyunyoon, Danshim Ha, Buhyun Hwang y Keun Ho Ryu (2003). «Mining association rules on significant rare data using relative support». En: *J. Syst. Softw.* 67.3. , págs. 181-191 (vid. pág. 55).

Zaki, Mohammed Javeed, Srinivasan Parthasarathy, Mitsunori Ogihara y Wei Li (1997). «New Algorithms for Fast Discovery of Association Rules». En: *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*. Ed. por David Heckerman, Heikki Mannila, Daryl Pregibon, Ramasamy Uthurusamy y Menlo Park. . Rochester, NY, USA: AAAI Press, págs. 283-286 (vid. pág. 47).

Zhang, Ji, Tok Wang Ling, Robert M. Bruckner, A. Min Tjoa y Han Liu (2004). «On Efficient and Effective Association Rule Mining from XML Data». En: *Database and Expert Systems Applications: 15th International Conference, DEXA 2004, Zaragoza, Spain, August 30-September 3, 2004. Proceedings*. Ed. por Fernando Galindo, Makoto Takizawa y Roland Traunmüller. . Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 497-507 (vid. pág. 29).

Zhang, Sheng, Ji Zhang, Han Liu y Wei Wang (2005). «XAR-miner: Efficient Association Rules Mining for XML Data». En: *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*. WWW '05. http://dl.acm.org/citation.cfm?id=1062745.1062785. Chiba, Japan: ACM, págs. 894-895 (vid. pág. 29).

Zhang, Shichao, Jilian Zhang y Chengqi Zhang (2007). «EDUA: An efficient algorithm for dynamic database mining». En: *Information Sciences* 177.13. , págs. 2756-2767 (vid. pág. 51).

Zhao, Qiankun y Sourav S. Bhowmick (2003). *Association Rule Mining: A Survey*. Inf. téc. TR116. . School of Computer Engineering. University of Nanyang Technological (vid. pág. 49).

A

Sobre la bibliografía

Todas las citas bibliográficas que figuran en este informe, tanto de trabajos propios como de otros investigadores, contienen enlaces externos que han sido revisados a principios de 2017.

Se han utilizado diferente imágenes[1] para indicar los recursos adicionales sobre la bibliografía utilizada que he ido recopilando a través del proceso de investigación. En la versión impresa aparecen a modo informativo, describiendo visualmente la información adicional que contiene cada elemento bibliográfico.

 indica que hay una copia del artículo, resumen o presentación en el `DVD` que complementa esta tesis, en formato `pdf`, en alguna de las carpetas de `bib`. Sólo es útil si se está revisando este documento desde el `DVD` adjunto o desde una copia de este documento colocada junto a una copia de la carpeta `bib` del `DVD`. El nombre de los archivos está compuesto por el apellido de los autores, separado con un guión de un texto representando el título del artículo y finalizando, tras otro guión, con el año de publicación.

 enlaza a la URL desde la que se puede descargar el artículo.

 enlaza a la URL con información sobre la referencia.

[1] Obtenidas de http://sourceforge.net/projects/openiconlibrary/, una completa librería de iconos e imágenes de código abierto.

Scopus enlaza a la URL de `http://www.scopus.com` con información sobre la referencia. Requiere acceso identificado, en mi caso proporcionado por mi universidad.

enlaza a la URL con el documento publicado como *Working Paper* para el Instituto Universitario Centro de Investigación Operativa de la Universidad Miguel Hernández de Elche.

enlaza a la URL en que he publicado este informe o la parte de él que se cita.

También están en el `DVD` adjunto las bases bibliográficas que he utilizado en este informe, todas ellas en la carpeta `bib` y con formato BibLaTeX. En ellas encontrará el lector más información que la expuesta en este informe. He utilizado los gestores bibliográficos `JabRef` [2] y `BibDesk` [3] pero por tratarse de ficheros de texto plano pueden abrirse con múltiples aplicaciones.

Todos los artículos incluidos en formato `.pdf` han sido descargados sin usar servicios de pago por lo que asumo que su utilización a través de la edición digital de este informe no infringe ninguna licencia de uso. Personalmente me ha resultado muy productivo tener preparados estos enlaces para elaborar un informe de mayor calidad y espero que al lector le sea útil para tener a mano mucha más información sobre los temas tratados en esta tesis.

[2] JabRef reference manager `http://jabref.sourceforge.net/`

[3] `http://bibdesk.sourceforge.net/`

B

Acrónimos

Los acrónimos utilizados en esta memoria se corresponden con la expresión en su versión inglesa debido a que es la que más aparece en la bibliografía utilizada.

AR Association Rule, regla de asociación.

ARM Association Rules Mining, minería de reglas de asociación.

DB Data Base, base de datos.

DBMS Data Base Management System, gestor de bases de datos.

CAR Classification Association Rule, regla de clasificación asociativa.

CARM Classification Association Rules Mining, minería de reglas de clasificación asociativa.

CR Classification Rule, regla de clasificación.

CRM Classification Rules Mining, minería de reglas de clasificación.

DM Data Mining, minería de datos.

FIM Frequent Itemsets Mining, minería de itemsets frecuentes.

KDD Knowledge Database Discovery, descubrimiento de conocimiento en bases de datos.

ML Machine Learning, aprendizaje automatizado.

RS Recommender System, sistema de recomendación.

WRS Web Recommender System, sistema de recomendación web.

WM Web Mining, minería web.

WUM Web Usage Mining, minería de uso web.

WWW World Wide Web.

C

Datos

Los datos crudos son los verdaderos protagonistas de esta investigación. Todas las propuestas que contiene esta memoria están basadas exclusivamente en la información que contienen los datos. La ciencia va siempre más allá y utiliza la experiencia y la teoría conjuntamente, para no tener que partir siempre desde cero. Pero para obtener experiencia ha de analizar datos crudos, como si no se supiera aún nada de lo que nos tienen que decir.

A lo largo de mi investigación he utilizado gran cantidad de datasets, la mayoría de acceso público y algunos propios o cedidos que no puedo publicar. El primer repositorio de datos que utilicé fue *FIM Data Repository* (2004), del que obtuve los datasets de transacciones:

- `BMS-POS.dat` y `BMS-View1.dat`, utilizados en la secciones 3.1.1 y 3.1.4,
- `foodmart` en la sección 3.1.2,
- `T10I4D100K`, `T40I10D100K.dat` y `kosarak` en la sección 3.1.4,
- `mushroom` (FIMI) utilizado en la sección 3.2.

En el capítulo 6 me he centrado en datasets de clasificación. He usado los 75 datasets publicados en *KEEL Standard Dataset Repository* (2004) (ver tabla 6.7), he probado la capacidad del algoritmo para trabajar con datasets de clasificación de grandes dimensiones con el dataset `kddcup99` de *UCI Machine Learning Repository* (2013) y he revisado los publicados en

LUCS-KDD discretised/normalised ARM and CARM Data Library (2003) para compararlos con los datasets originales, pero no los he utilizado porque opino que la transformación que se les ha aplicado es demasiado agresiva y provoca que se pierda gran parte de la información contenida en los datos crudos.

Los datasets utilizados están en el `DVD` adjunto, en la carpeta `./datos`, con formato de texto plano y representación horizontal (ver sección 2.1.2). Los datasets de clasificación de las carpetas `./datos/KEEL` y `./datos/UCI` pueden ser analizados directamente con el código proporcionado en el apéndice D.

El dataset `mushroom` (FIMI) está en la carpeta `./datos/FIMI`, es una transformación del dataset de clasificación `mushroom` (UCI) en la que se codificó cada valor de cada atributo utilizando un número entero, para adaptarlo a la lectura de transacciones de la minería de reglas de asociación.

El dataset de clasificación `mushroom` (KEEL) es el resultado de eliminar de `mushroom` (UCI) los registros con valores desconocidos.

En el dataset de clasificación `abalone` (KEEL) `http://sci2s.ugr.es/keel/dataset.php?cod=52` hay 3 registros menos que en el original publicado en UCI `http://archive.ics.uci.edu/ml/datasets/Abalone`.

Sólo para guardar los datasets que forman la colección de catálogos robustos de `mushroom`, su $\mathcal{CCR}_{\mathcal{D}}$, utilicé más de 20GB de disco duro, razón por la que no se incluyen en el `DVD` adjunto los resultados generados para llevar a cabo esta investigación.

D

Código

El código mostrado en este apéndice se ha simplificado para que sea más fácil de comprender. A lo largo de la investigación mostrada en esta memoria he escrito decenas de aplicaciones, estructuras y clases para poder analizar datasets de múltiples orígenes. Existen excelentes herramientas que podría haber utilizado en mi investigación pero mi pasión por la programación y las propuestas que han surgido en este tiempo no están implementadas en esas herramientas, o no he querido usarlas sin conocer a fondo qué hace la aplicación realmente con los datos proporcionados.

Al escribir esta memoria he recreado el código que me llevó en su día a comprobar si las propuestas eran factibles. En su momento tuve en cuenta las dificultades de trabajar con datasets en forma de archivos de texto plano, creando clases para trabajar con diferentes formatos de archivo, pero siempre hay un archivo con otro formato que provocaría errores en mis aplicaciones. El uso de excepciones es necesario en cualquier aplicación, sin embargo si publicara el código que he utilizado durante los últimos años el lector se perdería entre tanta excepción contemplada. En este apéndice muestro un código diseñado para la lectura de archivos de texto plano con representación horizontal (ver sección 2.1.2), usando excepciones típicas de lectura/escritura de archivos y de gestión de memorias.

He utilizado `C++` estándar siempre que ha sido posible. La librería *C++ Standard Template Library (STL)* 2011 hace fácil la programación con estructuras dinámicas, aunque no está especializada en la gestión de grandes

cantidades de datos, han demostrado su eficiencia en todos los experimentos realizados para esta tesis. Los modelos teóricos y algoritmos presentados en los capítulos 4 y 5 permitirán a cualquier investigador utilizar librerías o herramientas más especializadas para aplicar el análisis a otros formatos de dataset o a datasets de mayores dimensiones.

D.1. Librería de funciones comunes

El archivo scd.h contiene las funciones descritas en la sección 5.2.1. El archivo con el código está en el DVD adjunto [1]. Todas las aplicaciones generadas para obtener los experimentos del capítulo 6 usan estas funciones.

Listado D.1: Funciones para gestionar datasets de clasificación estándar

```
//  scd.h
//  lectura-scd
//
//  Created by Enrique Lazcorreta Puigmartí on 20/4/17.
//  Copyright © 2017 Enrique Lazcorreta Puigmartí. Some rights reserved.
//
//  This file is part of ACDC.
//
//  ACDC is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//
//  ACDC is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
//  GNU General Public License for more details.
//
//  You should have received a copy of the GNU General Public License
//  along with ACDC. If not, see <http://www.gnu.org/licenses/>.

#ifndef scd_h
#define scd_h


#include <iostream>
using std::cout;
using std::endl;
#include <iomanip>

#include <string>
using std::string;
```

[1] ./codigo/lib/scd.h

```
#include <vector>
using std::vector;

#include <map>
using std::map;

#include <set>
using std::set;

#include <algorithm>

#include <fstream>
using std::ifstream;
using std::ofstream;

typedef map<string, set<string>> TCatalogo;
typedef set<int> TSetEnteros;


inline void LeeLinea(ifstream &archivo, string &linea)
{
    std::getline(archivo, linea);

    if (!linea.empty())
    {
        if (linea.back() == '\r')
            linea.pop_back();

        size_t primer_caracter = linea.find_first_not_of(", ");

        if (primer_caracter)
            linea.assign(linea.substr(primer_caracter));
    }
}



size_t LeeDataset(ifstream &dataset,
                  TCatalogo &D_cero,
                  vector<map<string, size_t>> &Atributos,
                  int *num_atributos,
                  bool &clase_al_final,
                  bool lee_valores_y_clases = true)
{
    size_t num_evidencias_dataset = 0;

    //Comprobación de la posición de la clase
    string linea;
    LeeLinea(dataset, linea);

    if (!linea.compare("#clase_al_final"))
```

```
        LeeLinea(dataset, linea);
    else if (!linea.compare("#clase_al_principio"))
    {
        clase_al_final = false;
        LeeLinea(dataset, linea);
    }

    //Lectura de metadatos, si los tiene (se ignoran)
    char caracter_inicial = linea[0];
    while (caracter_inicial == '@' || caracter_inicial == '#' ||
        linea.empty())
    {
        LeeLinea(dataset, linea);
        caracter_inicial = linea[0];
    }

    //Obtención del número de atributos a partir de la primera evidencia
    *num_atributos = -1;
    for (size_t posicion_1 = linea.find_first_not_of(" ,"); posicion_1 <
        string::npos; )
    {
        size_t posicion_2 = linea.find_first_of(" ,", posicion_1);
        posicion_1 = linea.find_first_not_of(" ,", posicion_2);
        (*num_atributos)++;
    }

    Atributos.resize(*num_atributos + 1);

    //Leemos todas las evidencias completas
    //TODO: Podrían leerse también las incompletas si lo pide el
        usuario. No implementado. Necesitaría otro parámetro, al menos
        para decir cuántas evidencias incompletas hay.
    while (dataset.good())
    {
        if (linea.empty() || linea.find('?') != string::npos)
        {
            LeeLinea(dataset, linea);
            continue;
        }

        string caracterizacion(linea),
               clase;

        if (clase_al_final)
        {
            clase =
                caracterizacion.substr(caracterizacion.find_last_of(",
                ") + 1);

            caracterizacion = caracterizacion.substr(0,
                caracterizacion.find_last_of(", "));
            caracterizacion = caracterizacion.substr(0,
```

```
                caracterizacion.find_last_not_of(", ") + 1);
        }
        else
        {
            clase = caracterizacion.substr(0,
                caracterizacion.find_first_of(", "));

            caracterizacion =
                caracterizacion.substr(caracterizacion.find_first_of(",
                ") + 1);
            caracterizacion =
                caracterizacion.substr(caracterizacion.find_first_not_of(",
                "));
        }

        D_cero[caracterizacion].insert(clase);

        if (lee_valores_y_clases)
        {
            size_t posicion_dato = linea.find_first_not_of(", ");

            for (int atributo = 0; atributo < *num_atributos; atributo++)
            {
                size_t posicion_separador = linea.find_first_of(" ,",
                    posicion_dato);
                string dato(linea.substr(posicion_dato,
                    posicion_separador - posicion_dato));
                Atributos.at(atributo)[dato]++;
                posicion_dato = linea.find_first_not_of(" ,",
                    posicion_separador);
            }

            Atributos.at(*num_atributos)[clase]++;
        }

        num_evidencias_dataset++;

        LeeLinea(dataset, linea);
    }

    return num_evidencias_dataset;
}



size_t EliminaIncertidumbre(TCatalogo &D)
{
    size_t num_evidencias_con_y_sin_incertidumbre = D.size();

    for (TCatalogo::iterator evidencia = D.begin(); evidencia !=
        D.end(); )
        if (evidencia->second.size() > 1)
```

```
            evidencia = D.erase(evidencia);
        else
            ++evidencia;

    return (num_evidencias_con_y_sin_incertidumbre - D.size());
}



void GetValoresYClases(vector<map<string, size_t>> &Atributos_rango,
                       size_t *num_clases_distintas,
                       size_t *num_valores_distintos)
{
    *num_clases_distintas = Atributos_rango.at(Atributos_rango.size() -
        1).size();

    *num_valores_distintos = 0;
    for (auto atributo : Atributos_rango)
        *num_valores_distintos += atributo.size();
    *num_valores_distintos -= *num_clases_distintas;
}



inline const string ObtenSubCaracterizacion(const string
    &caracterizacion,
                                             const int
                                                 atributo_a_eliminar,
                                             const int num_atributos)
{
    if (atributo_a_eliminar == 0)
        return caracterizacion.substr(caracterizacion.find_first_of(",
            ") + 1);

    if (atributo_a_eliminar == num_atributos - 1)
        return caracterizacion.substr(0 ,caracterizacion.find_last_of(",
            "));

    size_t posicion_inicial = 0,
           posicion_final = 0;

    for (int i = 0; i < atributo_a_eliminar; i++)
    {
        posicion_final = caracterizacion.find_first_of(", ",
            posicion_inicial);
        posicion_inicial = caracterizacion.find_first_not_of(", ",
            posicion_final);
    }

    return caracterizacion.substr(0, posicion_final) + "," +
        caracterizacion.substr(caracterizacion.find_first_of(", ",
        posicion_inicial) + 1);
```

```
}



inline int ColumnaAEliminar(const int i,
                            const int num_atributos,
                            const TSetEnteros &I)
{
    int columna_a_eliminar = 0;

    for (int atributo = 0 ; atributo < num_atributos; atributo++)
    {
        if (I.find(atributo) != I.end())
            continue;
        if (atributo == i)
            break;
        columna_a_eliminar++;
    }

    return  columna_a_eliminar;
}



bool CreaCatalogoRobustoSinAtributo_i(const TCatalogo &D_sin_I,
                                      const TSetEnteros &I,
                                      const int i,
                                      const int num_atributos,
                                      TCatalogo &D_sin_I_sin_i)
{
    int dato_a_eliminar = ColumnaAEliminar(i, num_atributos, I);

    for (TCatalogo::const_iterator evidencia = D_sin_I.begin();
        evidencia != D_sin_I.end(); ++evidencia)
    {
        string caracterizacion(ObtenSubCaracterizacion(evidencia->first,
            dato_a_eliminar, num_atributos - (int)I.size()));
        D_sin_I_sin_i[caracterizacion].insert(*(evidencia->second.cbegin()));

        if (D_sin_I_sin_i[caracterizacion].size() > 1)
        {
            D_sin_I_sin_i.clear();
            return false;
        }
    }

    return true;
}



size_t CreaCCR(TCatalogo &D_sin_I,
```

```
                TSetEnteros &I,
                const int ultimo_atributo_eliminado,
                const int num_atributos,
                set<TSetEnteros> &CCR)
{
    static size_t num_llamadas = 0;
    num_llamadas++;

    for (int atributo_a_eliminar = ultimo_atributo_eliminado + 1;
         atributo_a_eliminar < num_atributos; atributo_a_eliminar++)
    {
        if (I.find(atributo_a_eliminar) != I.end())
            continue;

        I.insert(atributo_a_eliminar);

        //Comprobar si ya sabemos que I es robusto (si D^-I es
            superconjunto de un catálogo robusto)




        for (TCatalogo::const_iterator evidencia = D_sin_I.begin();
            evidencia != D_sin_I.end(); ++evidencia)
        {

        }
    }

    return num_llamadas;
}



void EliminaAtributoConstante(TCatalogo &D_sin_I,
                              TSetEnteros &I,
                              const int i,
                              const int num_atributos)
{
    TCatalogo D_reducido;

    for (auto atributo : D_sin_I)
    {
        string
            caracterizacion_sin_i(ObtenSubCaracterizacion(atributo.first,
            i, num_atributos));
        D_reducido[caracterizacion_sin_i] = atributo.second;
    }

    D_sin_I.swap(D_reducido);
    I.insert(i);
```

```
}



void EliminaAtributosConstantes(TCatalogo &D_sin_I,
                                TSetEnteros &I,
                                const vector<map<string, size_t>>
                                    &Atributos_rango,
                                TSetEnteros &Atributos_constantes,
                                const int num_atributos)
{
    int indice_atributo = 0;

    for (vector<map<string, size_t>>::const_iterator atributo =
        Atributos_rango.begin(); atributo != Atributos_rango.end();
        ++atributo)
    {
        if (atributo->size() == 1)
        {
            Atributos_constantes.insert(indice_atributo);
            EliminaAtributoConstante(D_sin_I, I, indice_atributo,
                num_atributos);
            I.insert(indice_atributo);
        }

        indice_atributo++;
    }
}

#endif /* scd_h */
```

D.2. Características de los datasets de clasificación

La aplicación generada con el código del archivo `src/ccr-nivel-1/main.cpp` realiza la primera iteración del algoritmo $\mathcal{ACDC}$. El archivo con el código está en el DVD adjunto [2].

Clasifica los atributos como constantes, esenciales o redundantes para el experimento de clasificación. Elimina los atributos constantes y comprueba las características de los atributos redundantes A_i mostrando al investigador el número de valores distintos y el número de evidencias de cada dataset reducido $\mathcal{D}^{-i}$. Con la información que se descubre en el dataset, el investigador puede diseñar mejor el análisis completo del dataset, decidir si quiere eliminar un atributo redundante en concreto, bien para

[2] ./codigo/src/ccr-nivel-1/main.cpp

minimizar el número de evidencias a tratar, bien para minimizar el número de valores distintos del dataset, o bien para poder trabajar sin un atributo redundante concreto cuando su coste es desaconsejable o su medición compleja o imposible.

Listado D.2: Primer análisis de datasets de clasificación estándar

```
//  main.cpp
//  ccr-nivel-1
//
//  Created by Enrique Lazcorreta Puigmartí on 20/4/17.
//  Copyright © 2017 Enrique Lazcorreta Puigmartí. All rights reserved.
//

#include "../../lib/comun.h"
#include "../../lib/scd.h"

#include <iostream>
using std::cout;
using std::endl;
#include <iomanip>

#include <string>
using std::string;

#include <vector>
using std::vector;

#include <map>
using std::map;

#include <set>
using std::set;

#include <fstream>
using std::ifstream;
using std::ofstream;

typedef map<string, set<string>> TCatalogo;
typedef set<int> TSetEnteros;



void Uso(const char * argv[])
{
    cout << "Uso:" << endl << endl;
    cout << '\t' << NombreAplicacion(argv[0]) << "
        dataset-de-clasificacion" << endl << endl;
    cout << "El archivo dataset-de-clasificacion es de texto plano
        separado "
    "por comas (formato CSV)." << endl << endl;
```

```
    cout << "Puede contener metadatos en las primeras líneas si su
        primer "
    "caracter es '@' o '#'." << endl << endl;
    cout << "La clase es el último dato de cada línea. Sólo si la
        primera "
    "línea del dataset contiene \"#clase_al_principio\" se "
    "tomará el primer dato como la clase del experimento." << endl <<
        endl;
}



void ProcesaArgumentos(const int argc, const char * argv[])
{
    if (argc != 2)
    {
        Uso(argv);
        exit(1);
    }
}


int main(int argc, const char * argv[])
{
    ProcesaArgumentos(argc, argv);

    DECLARA_E_INICIA_RELOJ_TOTAL

    //Apertura del dataset
    string nombre_dataset(argv[1]);
    ifstream dataset(nombre_dataset);
    if (!dataset.is_open())
    {
        cout << "No se ha podido abrir el dataset" << endl << endl
        << nombre_dataset << endl << endl
        << "Comprueba que existe el archivo.";

        return 2;
    }
    else
        cout << "Analizando " << nombre_dataset << endl;

    ///< La clase puede ser el primer o último elemento de cada evidencia
    bool clase_al_final = true;

    TCatalogo D;

    vector<map<string, size_t>> Atributos_rango;

    int num_atributos;

    //Lectura de evidencias completas
```

```
    size_t num_evidencias_dataset = LeeDataset(dataset, D,
        Atributos_rango, &num_atributos, clase_al_final);

    //Ya no se usará el dataset
    dataset.close();

    size_t num_clases_distintas,
    num_valores_distintos;

    GetValoresYClases(Atributos_rango, &num_clases_distintas,
        &num_valores_distintos);

    size_t num_evidencias_completas_dataset = D.size();

    //Conjuntos de atributos
    TSetEnteros Atributos_constantes,
    Atributos_redundantes,
    Atributos_esenciales,
    Atributos_eliminados;

    //Los atributos constantes se han de eliminar desde el principio
    EliminaAtributosConstantes(D, Atributos_eliminados, Atributos_rango,
        Atributos_constantes, num_atributos);

    if (Atributos_constantes.size())
    {
        cout << "Se han eliminado " << Atributos_constantes.size() << "
            atributos constantes:";
        for (auto atributo : Atributos_constantes)
            cout << " A" << atributo + 1;
        cout << endl;

    }

    ///< Eliminación de incertidumbre
    size_t num_caracterizaciones_completas_con_incertidumbre =
        EliminaIncertidumbre(D);

    size_t num_evidencias_completas_robustas = D.size();

    //TODO: Estas variables sólo son necesarias si se va ha ejecutar ACDC
        de forma supervisada
    size_t num_evidencias_al_minimizar_evidencias =
        num_evidencias_completas_robustas + 1,
    num_valores_al_minimizar_evidencias = num_valores_distintos,
    num_evidencias_al_minimizar_valores =
        num_evidencias_completas_robustas,
    num_valores_al_minimizar_valores = num_valores_distintos + 1;
    int atributo_que_minimiza_num_evidencias = 0,
    atributo_que_minimiza_num_valores = 0;

    for (int atributo = 0; atributo < num_atributos; atributo++)
```

```
    {
        if (Atributos_eliminados.find(atributo) !=
            Atributos_eliminados.end())
            continue;

        TCatalogo D_sin_atributo;

        if (CreaCRsinAi(D, Atributos_eliminados, atributo,
            num_atributos, D_sin_atributo))
        {
            cout << "A" << atributo + 1 << ": \t" <<
                Miles(Atributos_rango.at(atributo).size()) << " \t" <<
                Miles(D_sin_atributo.size()) << endl;

            Atributos_redundantes.insert(atributo);

            if (num_evidencias_al_minimizar_evidencias >
                D_sin_atributo.size())
            {
                num_evidencias_al_minimizar_evidencias =
                    D_sin_atributo.size();
                num_valores_al_minimizar_evidencias =
                    num_valores_distintos -
                    Atributos_rango.at(atributo).size();
                atributo_que_minimiza_num_evidencias = atributo;
            }

            if (num_valores_al_minimizar_valores > num_valores_distintos
                - Atributos_rango.at(atributo).size())
            {
                num_valores_al_minimizar_valores = num_valores_distintos
                    - Atributos_rango.at(atributo).size();
                num_evidencias_al_minimizar_valores =
                    D_sin_atributo.size();
                atributo_que_minimiza_num_valores = atributo;
            }
        }
        else
            Atributos_esenciales.insert(atributo);
    }


    //Tabla formateada para LaTeX
    cout << NombreArchivo(nombre_dataset) << " & "
    << GetTamanyoArchivo(nombre_dataset) << " & ";

    //\newcommand{\todos}[2][-]{{\color{teal}#2{\tiny (#2)}{\tiny (#1)}}}
    //\newcommand{\ninguno}[2][-]{{\color{red}#2{\tiny (0)}{\tiny (#1)}}}
    //\newcommand{\alguno}[3][-]{#2{\tiny (#3)}{\tiny (#1)}}
    string msg_parametro_constantes = Atributos_constantes.size() ? "["
        + Miles(Atributos_constantes.size()) + "]" : "";
```

```
    string msg_atributos;
    if (num_atributos == Atributos_redundantes.size() +
        Atributos_constantes.size())
        msg_atributos = "\\todos" + msg_parametro_constantes + "{" +
            Miles(num_atributos) + "}";
    else if (!(Atributos_redundantes.size() +
        Atributos_constantes.size()))
        msg_atributos = "\\ninguno" + msg_parametro_constantes + "{" +
            Miles(num_atributos) + "}";
    else
        msg_atributos = "\\alguno" + msg_parametro_constantes + "{" +
            Miles(num_atributos) + "}{" +
            Miles(Atributos_redundantes.size() +
            Atributos_constantes.size()) + "}";
    cout << msg_atributos << " & "
    << Miles(num_valores_distintos) << " & "
    << Miles(num_clases_distintas) << " & "
    << Miles(num_evidencias_dataset) << " & ";

    //\newcommand{\sinDuplicados}[1]{{\color{teal}#1}}
    string msg_num_evidencias_completas;
    if (num_evidencias_completas_dataset == num_evidencias_dataset)
        msg_num_evidencias_completas = "\\sinDuplicados{" +
            Miles(num_evidencias_completas_dataset) + "}";
    else
        msg_num_evidencias_completas =
            Miles(num_evidencias_completas_dataset);
    cout << msg_num_evidencias_completas << " & "
    << (num_caracterizaciones_completas_con_incertidumbre ?
        Miles(num_caracterizaciones_completas_con_incertidumbre) : "-")
        << " & ";

    if (Atributos_redundantes.size())
        cout << "A_{"<<Miles(atributo_que_minimiza_num_evidencias+1)<<"} ("
        << Miles(num_valores_al_minimizar_evidencias) << " / "
        << Miles(num_evidencias_al_minimizar_evidencias) << ") & "
        << "A_{"<<Miles(atributo_que_minimiza_num_valores+1)<<"} ("
        << Miles(num_valores_al_minimizar_valores) << " / "
        << Miles(num_evidencias_al_minimizar_valores) << ") & ";
    else
        cout << "            &            & ";

    cout << std::setprecision(4) << std::fixed <<
        TIEMPO_TRANSCURRIDO_TOTAL << " \\\\" << endl << "\\hline" <<
        endl;

    //Generar la Colección de Catálogos Robustos


    return 0;
}
```

D.3. Análisis de múltiples datasets

El listado D.3 contiene el código utilizado para elaborar la tabla 6.7. El archivo con el código está en el DVD adjunto [3].

La aplicación generada recibe como parámetro la ruta y nombre de un archivo de texto plano en el que están todos los datasets que han de ser analizados. Cada línea del archivo contiene la ruta completa de un dataset. Si no hay ningún problema la aplicación mostrará en la salida estándar una tabla formateada en LaTeX, como la que he usado en la tabla mencionada.

Al leer un dataset obtiene sus características básicas: número de evidencias, número de atributos, valores distintos utilizados por los atributos y número de clases. Con esta información, si comprueba que contiene atributos constantes los elimina pues ya se sabe que no aportarán información válida para el problema de clasificación. A continuación ejecuta la primera iteración del algoritmo $\mathcal{C}$ sobre el resto de atributos, tras la que descubre qué atributos son redundantes y describe las características del dataset reducido $\mathcal{D}^{-i}$ para todo A_i que sea redundante. En el listado obtenido se muestra el atributo redundante que más evidencias elimina de $\mathcal{D}$ y el atributo redundante que más valores aporta al dataset.

Listado D.3: Lector de datasets de clasificación estándar

```
//  main.cpp
//  lectura-scd
//
//  Created by Enrique Lazcorreta Puigmartí on 20/4/17.
//  Copyright © 2017 Enrique Lazcorreta Puigmartí. All rights reserved.
//

#include "../../lib/comun.h"
#include "../../lib/scd.h"

#include <iostream>
using std::cout;
using std::endl;
#include <iomanip>

#include <string>
using std::string;

#include <vector>
using std::vector;
```

[3] ./codigo/src/lector-scd/main.cpp

```
#include <map>
using std::map;

#include <fstream>
using std::ifstream;



int main(int argc, const char * argv[])
{
    if (argc < 2)
    {
        cout << "Uso:" << endl << endl;
        cout << NombreAplicacion(argv[0]) << " listado-datasets" << endl
            << endl;
        cout << "En el archivo listado-datasets está el listado de los "
        "datasets a analizar, cada uno en una línea y con la "
        "ruta completa del dataset." << endl;

        return 1;
    }

    ifstream listado_datasets(argv[1]);
    if (!listado_datasets.is_open())
    {
        cout << "No se ha podido abrir el listado" << endl << endl
        << argv[1] << endl << endl
        << "Comprueba que existe el archivo.";

        return 2;
    }
    else
        cout << argv[1] << " abierto" << endl;

    DECLARA_E_INICIA_RELOJ_TOTAL

    int num_datasets_analizados = 0,
        num_datasets_con_atributos_redundantes = 0,
        num_datasets_con_atributos_constantes = 0,
        num_datasets_sin_duplicados = 0,
        num_datasets_con_incertidumbre = 0;

    //Cabecera de la tabla
    cout << "Cabecera de la tabla" << endl
         << "\t#       = número del dataset analizado" << endl
         << "\tDataset = nombre del dataset" << endl
         << "\tTamaño = tamaño del archivo que contiene el dataset" <<
            endl
         << "\tN(r)(c) = número de atributos (redundantes) (constantes)"
            << endl
         << "\trg(A)   = número de valores distintos (suma de rangos de
            los atributos)" << endl
```

```
        << "\tQ        = número de clases" << endl
        << "\t|dat|    = número de evidencias del dataset" << endl
        << "\tM        = número de evidencias completas" << endl
        << "\tInc.     = número de caracterizaciones con incertidumbre"
            << endl
        << "\t(1)      = \"atributo eliminado (valores, evidencias)\"
            para minimizar evidencias" << endl
        << "\t(2)      = \"atributo eliminado (valores, evidencias)\"
            para minimizar valores" << endl
        << "\tt(sg)    = tiempo de ejecución, en segundos" << endl
        << endl;

    cout << "#   &   "
        << "Dataset " << " & "
        << "Tamaño " << " & "
        << "N(r)(c) " << " & "
        << "rg(A)   " << " & "
        << "Q       " << " & "
        << "|dat|   " << " & "
        << "M       " << " & "
        << "Inc.    " << " & "
        << "(1)     " << " & "
        << "(2)     " << " & "
        << "t(sg)   " << " \\\\" << endl << "\\hline"
        << endl;

    while (listado_datasets.good())
    {
        DECLARA_E_INICIA_RELOJ

        //Apertura del dataset
        string nombre_dataset;
        LeeLinea(listado_datasets, nombre_dataset);
        ifstream dataset(nombre_dataset);
        if (nombre_dataset.empty())
            continue;

        if (!dataset.is_open())
        {
            cout << "******* " << nombre_dataset << " no abierto
                *******" << endl;
            continue;
        }

        num_datasets_analizados++;

        ///< La clase puede ser el primer o último elemento de cada
            evidencia
        bool clase_al_final = true;

        TCatalogo D;
```

```
        vector<map<string, size_t>> Atributos_rango;

        int num_atributos;

        //Lectura de evidencias completas
        size_t num_evidencias_dataset = LeeDataset(dataset,
                                                   D,
                                                   Atributos_rango,
                                                   &num_atributos,
                                                   clase_al_final);

        //Ya no se usará el dataset
        dataset.close();

        int indice_primer_atributo = clase_al_final ? 0 : 1,
            indice_ultimo_atributo = clase_al_final ? num_atributos - 1
                : num_atributos - 2,
            indice_clase = clase_al_final ? num_atributos : 0;

        size_t num_clases_distintas =
            Atributos_rango.at(indice_clase).size();

        size_t num_valores_distintos = 0;
        for (auto atributo : Atributos_rango)
            num_valores_distintos += atributo.size();
        num_valores_distintos -= num_clases_distintas;

        size_t num_evidencias_completas_dataset = D.size();

        ///< Eliminación de incertidumbre
        size_t num_caracterizaciones_completas_con_incertidumbre =
            EliminaIncertidumbre(D);

        if (num_caracterizaciones_completas_con_incertidumbre)
            num_datasets_con_incertidumbre++;

        size_t num_evidencias_completas_robustas = D.size();

        if (num_evidencias_completas_robustas == num_evidencias_dataset)
            num_datasets_sin_duplicados++;

        //Conjuntos de atributos
        TSetEnteros Atributos_constantes,
                    Atributos_redundantes,
                    Atributos_esenciales;

        //Generar la Colección de Catálogos Robustos
        size_t num_evidencias_al_minimizar_evidencias =
            num_evidencias_completas_robustas + 1,
               num_valores_al_minimizar_evidencias =
                   num_valores_distintos,
               num_evidencias_al_minimizar_valores =
```

```
                num_evidencias_completas_robustas,
            num_valores_al_minimizar_valores = num_valores_distintos
                + 1;
        int atributo_que_minimiza_num_evidencias = 0,
            atributo_que_minimiza_num_valores = 0;

        for (int atributo = indice_primer_atributo; atributo <=
            indice_ultimo_atributo; atributo++)
        {
            if (Atributos_rango.at(atributo).size() == 1)
            {
                Atributos_constantes.insert(atributo);
                continue;
            }

            TCatalogo D_reducido;

            for (TCatalogo::const_iterator evidencia = D.begin();
                 evidencia != D.end(); ++evidencia)
            {
                string
                    caracterizacion(ObtenSubCaracterizacion(evidencia->first,
                                                            atributo,
                                                            num_atributos));
                D_reducido[caracterizacion].insert(*(evidencia->second.cbegin()));
                if (D_reducido[caracterizacion].size() > 1)
                {
                    Atributos_esenciales.insert(atributo);
                    break;
                }
            }

            //TODO:Comprobar si find devuelve end() o npos (como string)
            if (Atributos_esenciales.find(atributo) ==
                Atributos_esenciales.end())
            {
                if (num_evidencias_al_minimizar_evidencias >
                    D_reducido.size())
                {
                    num_evidencias_al_minimizar_evidencias =
                        D_reducido.size();
                    num_valores_al_minimizar_evidencias =
                        num_valores_distintos -
                        Atributos_rango.at(atributo).size();
                    atributo_que_minimiza_num_evidencias = atributo + 1;
                }

                if (num_valores_al_minimizar_valores >
                    num_valores_distintos -
                    Atributos_rango.at(atributo).size())
                {
                    num_valores_al_minimizar_valores =
```

```
                        num_valores_distintos -
                        Atributos_rango.at(atributo).size();
                    num_evidencias_al_minimizar_valores =
                        D_reducido.size();
                    atributo_que_minimiza_num_valores = atributo + 1;
                }
            }
        }

        if (Atributos_esenciales.size() + Atributos_constantes.size() <
            num_atributos)
            num_datasets_con_atributos_redundantes++;

        if (Atributos_constantes.size())
        {
            num_datasets_con_atributos_constantes++;
//            num_datasets_con_atributos_redundantes++;
        }

        //Tabla formateada para LaTeX
        cout << Miles(num_datasets_analizados)<< " & "
             << NombreArchivo(nombre_dataset) << " & "
             << GetTamanyoArchivo(nombre_dataset) << " & ";

        //\newcommand{\todos}[2][-]{{\color{teal}#2{\tiny (#2)}{\tiny
            (#1)}}}
        //\newcommand{\ninguno}[2][-]{{\color{red}#2{\tiny (0)}{\tiny
            (#1)}}}
        //\newcommand{\alguno}[3][-]{#2{\tiny (#3)}{\tiny (#1)}}
        string msg_parametro_constantes = Atributos_constantes.size() ?
                                    "[" + Miles(Atributos_constantes.size())
                                        + "]" : "";
        int num_atributos_redundantes = num_atributos -
            (int)Atributos_esenciales.size() -
            (int)Atributos_constantes.size();
        string msg_atributos;
        if (num_atributos == num_atributos_redundantes +
            Atributos_constantes.size())
            msg_atributos = "\\todos" + msg_parametro_constantes + "{" +
                Miles(num_atributos) + "}";
        else if (!num_atributos_redundantes)
            msg_atributos = "\\ninguno" + msg_parametro_constantes + "{"
                + Miles(num_atributos) + "}";
        else
            msg_atributos = "\\alguno" + msg_parametro_constantes + "{"
                + Miles(num_atributos) + "}{" +
                Miles(num_atributos_redundantes) + "}";
        cout << msg_atributos << " & "
             << Miles(num_valores_distintos) << " & "
             << Miles(num_clases_distintas) << " & "
             << Miles(num_evidencias_dataset) << " & ";
```

```
        //\newcommand{\sinDuplicados}[1]{{\color{teal}#1}}
        string msg_num_evidencias_completas;
        if (num_evidencias_completas_dataset == num_evidencias_dataset)
            msg_num_evidencias_completas = "\\sinDuplicados{" +
                Miles(num_evidencias_completas_dataset) + "}";
        else
            msg_num_evidencias_completas =
                Miles(num_evidencias_completas_dataset);
        cout << msg_num_evidencias_completas << " & "
             << (num_caracterizaciones_completas_con_incertidumbre ?
                 Miles(num_caracterizaciones_completas_con_incertidumbre)
                 : "-") << " & ";

        if (num_atributos_redundantes)
            cout << "$A_{" << Miles(atributo_que_minimiza_num_evidencias) << "}$ ("
                 << Miles(num_valores_al_minimizar_evidencias) << " / "
                 << Miles(num_evidencias_al_minimizar_evidencias) << ")
                  & "
                 << "$A_{" << Miles(atributo_que_minimiza_num_valores) << "}$ ("
                 << Miles(num_valores_al_minimizar_valores) << " / "
                 << Miles(num_evidencias_al_minimizar_valores) << ") & ";
        else
            cout << "                &                & ";

        cout << std::setprecision(4) << std::fixed <<
            TIEMPO_TRANSCURRIDO << " \\\\" << endl << "\\hline" << endl;
    }

    cout << endl
         << "TOTAL &  &  & ("
         << Miles(num_datasets_con_atributos_redundantes) << ")("
         << Miles(num_datasets_con_atributos_constantes) << ") &  &  & "
         << Miles(num_datasets_sin_duplicados) << " & "
         << Miles(num_datasets_con_incertidumbre) << " &  &  & "
         << std::setprecision(4) << std::fixed <<
             TIEMPO_TRANSCURRIDO_TOTAL << endl;

    return 0;
}
```

D.4. Macros y funciones auxiliares

El archivo `comun.h` contiene macros y funciones auxiliares. El archivo con el código está en el DVD adjunto [4].

Listado D.4: Macros y funciones auxiliares

[4] ./codigo/lib/comun.h

```
//  comun.h
//  lectura-scd
//
//  Created by Enrique Lazcorreta Puigmartí on 20/4/17.
//  Copyright © 2017 Enrique Lazcorreta Puigmartí. All rights reserved.
//

#ifndef comun_h
#define comun_h

#include <ctime>
#define DECLARA_E_INICIA_RELOJ_TOTAL clock_t start_TOTAL = clock();
#define TIEMPO_TRANSCURRIDO_TOTAL ((clock() - start_TOTAL) /
    double(CLOCKS_PER_SEC))
#define MSG_TIEMPO_TRANSCURRIDO_TOTAL cout << std::setprecision(4) <<
    std::fixed << "[" << TIEMPO_TRANSCURRIDO_TOTAL << "sg]";

#define DECLARA_E_INICIA_RELOJ clock_t start = clock();
#define INICIA_RELOJ start = clock();
#define TIEMPO_TRANSCURRIDO ((clock() - start) / double(CLOCKS_PER_SEC))
#define MSG_TIEMPO_TRANSCURRIDO cout << std::setprecision(4) <<
    std::fixed << "(" << TIEMPO_TRANSCURRIDO << "sg)";

#include <string>
using std::string;

#include <fstream>
using std::ifstream;

#include <sstream>
using std::stringstream;

#include <iomanip>



string NombreAplicacion(const char *argv_0)
{
    string nombre_aplicacion(argv_0);
    nombre_aplicacion =
        nombre_aplicacion.substr(nombre_aplicacion.find_last_of("/\\") +
        1);
    return nombre_aplicacion;
}



string NombreArchivo(const string &nombre_y_ruta)
{
    string nombre =
        nombre_y_ruta.substr(nombre_y_ruta.find_last_of("/\\") + 1);
```

```
    size_t posicion = nombre.find_last_of(".");

    return nombre.substr(0, posicion);
}




string ExtensionArchivo(const string &nombre_y_ruta)
{
    string extension =
        nombre_y_ruta.substr(nombre_y_ruta.find_last_of("/\\") + 1);

    size_t posicion = extension.find_last_of(".");

    if (posicion != string::npos)
        return extension.substr(posicion + 1);

    return "";
}




string RutaArchivo(const string nombre_y_ruta)
{
    string ruta(nombre_y_ruta.substr(0,
        nombre_y_ruta.find_last_of("/\\") + 1));
    return ruta;
}



string TamanyoArchivoLegibleParaHumanos(size_t tamanyo_archivo)
{
    int unidad = 0;

    float tamanyo = (float)tamanyo_archivo;
    while (tamanyo > 1024 && unidad < 4)
    {
        tamanyo /= 1024;
        unidad++;
    }

    string resultado;
    char buffer[10];

    if (!unidad)
```

```
        sprintf(buffer, "%zu", (size_t)tamanyo);
    else
        sprintf(buffer, "%.2f", tamanyo);

    resultado.append(buffer);

    switch (unidad) {
        case 0: // Bytes
            resultado.append("B");
            break;

        case 1: // KB
            resultado.append("KB");
            break;

        case 2: // MB
            resultado.append("MB");
            break;

        case 3: // GB
            resultado.append("GB");
            break;

        case 4: // TB
            resultado.append("TB");
    }

    return resultado;
}


string GetTamanyoArchivo(const string &nombre_archivo)
{
    size_t tamanyo = 0;

    ifstream archivo(nombre_archivo);

    if (archivo.is_open())
    {
        size_t posicion_actual = archivo.tellg();
        archivo.seekg(0, archivo.end);
        tamanyo = (double)archivo.tellg();
        archivo.seekg(posicion_actual, archivo.beg);
    }

    archivo.close();

    return TamanyoArchivoLegibleParaHumanos(tamanyo);
}
```

```
#define SEPARADOR_DE_MILES ","
string Miles(size_t numero,
             const string separador_de_miles = SEPARADOR_DE_MILES)
{
    if (!numero)
        return "0";

    string numero_con_separaciones;
    char digitos_escritos = 0;

    while (numero)
    {
        numero_con_separaciones.insert(0, 1, (char)(numero - (numero /
            10) * 10 + 48));

        numero /= 10;

        digitos_escritos++;
        if (digitos_escritos == 3 && numero)
        {
            numero_con_separaciones.insert(0, separador_de_miles);
            digitos_escritos = 0;
        }
    }

    return numero_con_separaciones;
}


template <typename T>
string NumeroAString(T numero,
                     const int precision = 2)
{
    stringstream ss;
    ss << std::setprecision(precision) << std::fixed << numero;
    return ss.str();
}

#endif /* comun_h */
```

UNIVERSITAS
Miguel
Hernández

Índice alfabético

Algoritmo
- *ACDC* (Análisis de Caracterizaciones en Datasets de Clasificación), 4, 118, 129, 132, 136, 138, 143, 145–147, 149–152, 154–168, 176, 205
- ADT, 63
- AFOPT, 48
- AIS, 29, 30, 37
- Apriori-Hybrid, 35
- Apriori2, 67, 84, 85
- AprioriAll, 12
- AprioriHybrid, 34
- AprioriSome, 12
- AprioriTID, 34, 35
- Apriori, 34, 35, 37, 38, 45–49, 52, 61, 67, 68, 70, 73, 74, 80, 81, 85, 88, 89, 91, 94, 95, 97, 98, 100, 102, 131, 136
  - $\mathcal{C}$, Conjunto de Candidatos, 23, 29–32, 35–40, 43, 44, 69, 88, 89, 91, 94
  - $\mathcal{L}$, Conjunto de Itemsets Frecuentes, 23, 29, 30, 34–44, 49, 68, 69, 73–76, 88–91, 94, 98, 100, 101, 131, 136
- Basic, 42
- BitTableFI, 27, 49
- C4.5, 59, 62
- CAR-IC, 64
- CBAR, 48
- CBA, 61, 62
- CDAR, 48
- CFP-growth, 53
- CMAR, 12, 63
- CPAR, 63
- Candidate Distribution, 41, 42
- Clique, 45
- Count Distribution, 41, 42
- Cumulate, 42

DHP, 38, 45–47
DIC, 42, 53
Data Distribution, 41, 42
Direct, 44
DynamicSome, 12
Eclat, 45, 46
Efficient Dynamic Database Updating Algorithm (EDUA), 49
EstMerge, 43
FP-Growth, 45, 46, 48, 63, 94
FP-ORFind, 94
GSP, 12
GenMax, 48
IHP, 47
M-Apriori, 45
MAFIA, 48
MCAR, 63
ML_T1LA, 39
ML_T2L1, 39
ML_T2LA, 39
ML_TML1, 39
MSApriori, 52, 53
Max-Miner, 45
MaxClique, 45
MaxEclat, 45
Multipass DHP (M-DHP, 45
Multipass-Apriori, 45
MultipleJoins, 44
OCD, 37
ORFind, 90, 91, 93
PHP, 46
Partition, 39, 46, 47
Pincer-Search, 46
Reorder, 44
SEAR, 40
SETM, 30
SPEAR, 40
SPINC, 40
TFPC, 63
Aprendizaje Automatizado (ML), 5, 6, 11, 61, 96

C, 70, 72
C++, 70
Clasificación, 2, 3, 10–12, 18–21, 27, 51, 54, 55, 57, 59–61, 63, 65, 67, 84, 96, 100, 104–106, 108, 110, 114, 118, 126, 128, 131, 146, 147, 150, 152, 154, 159–162, 166, 173–176
  Catálogo, 4, 14, 96, 106, 110–112, 115, 117–120, 123–127, 130–143, 145, 151, 152, 159–162, 164, 166, 168, 169, 173–176
  Catálogo Robusto, 122, 124, 125, 127–134, 136, 138, 140–143, 146, 147, 149, 152, 154, 156, 157, 159–169, 176
  Colección de Catálogos Robustos, 130, 135, 136, 142, 156
  Experimento de clasificación $\mathcal{E}$, 55, 59, 65, 66, 96, 101, 113, 115, 116, 119, 123, 125–132, 135, 143, 147, 154, 156, 157, 159–163, 166, 168, 169, 176

Descubrimiento de Conocimiento en Bases de Datos (KDD), 1, 2, 5–9, 11, 13, 19, 26, 64, 132, 146
  Base de Datos (DB), 1, 5–8, 13, 16, 19, 26, 27, 29–31, 38, 40–

42, 45, 48–52, 54, 59, 65, 174, 176
Gestor de Bases de Datos (DBMS), 27, 30, 136

Minería de Datos (DM), 1, 2, 5, 6, 8, 10–12, 19, 26, 27, 53–55, 61, 64, 71, 121, 146, 147, 154, 174
Minería de Itemsets Frecuentes (FIM), 24, 33, 42, 46, 48, 50, 70, 73, 100, 108
Minería de Reglas de Asociación (ARM), 2, 3, 10, 12, 14, 17, 19, 20, 22–24, 26–29, 32–34, 37, 39–41, 45–52, 54, 55, 57–61, 63, 65, 67–70, 72, 74, 79, 84, 87, 88, 92, 96–98, 100–102, 104, 106, 109–118, 123, 125, 127, 131, 138, 142, 146–150, 155, 160, 173–175, 196
$\mathcal{D}$, dataset o base de datos de transacciones, 2, 3, 16–35, 37–43, 45–58, 60, 61, 67–69, 71–73, 77–85, 88, 91–98, 100–102, 104, 111, 146, 195, 196
Minería de Reglas de Clasificación (CRM), 54, 57, 60–63, 95, 117, 173
Minería de Reglas de Clasificación Asociativa (CARM), 3, 4, 12, 54, 60–64, 117, 123, 126, 127, 131, 147, 153, 161, 173–175
$\mathcal{D}_0$, dataset de clasificación, 3, 4, 55–57, 59–61, 63–66, 95–100, 102–120, 123–127, 130–141, 143, 145–157, 159–163, 165, 166, 168, 169, 171–176, 195, 196, 205
Minería Web (WM), 6
Minería de Uso Web (WUM), 7, 14, 18, 19, 21
Repositorio FIMI, 28, 70, 71, 108, 195
`mushroom`, 97, 98, 101, 104, 107, 108, 195, 196
Repositorio KEEL, 4, 28, 103, 105, 106, 109–111, 113, 115, 146–148, 151, 156, 162, 163, 195
`mushroom`, 104, 109, 196
Repositorio LUCS-KDD, 64, 109, 196
`mushroom`, 109
Repositorio UCI, 28, 63–65, 101, 103–105, 107–110, 163, 195, 196
`mushroom`, 58, 107–109, 113, 150, 196

Regla de Asociación (AR), 2, 16, 17, 19–26, 29, 33, 35, 37, 38, 40–42, 45, 48, 49, 51–53, 59, 61, 63, 67, 73, 75, 78–82, 86–92, 94–96, 98, 100, 101, 103, 104, 106, 110, 111, 115, 116, 123, 126, 127, 147, 148, 160, 174, 175
$conviction$, 26
$lift$, 25
Antecedente, 17, 20, 24, 26, 33,

42, 53, 61, 74, 86, 90, 105
Confianza, 17, 21, 24, 25, 35, 42, 53, 59, 61, 65, 73–75, 77–79, 81–84, 86–90, 92–94, 97, 113, 125, 126, 175
Consecuente, 17, 20, 24–26, 33, 42, 53, 59, 61, 86, 90, 96, 100, 105
Ítem Raro, 3, 22, 23, 51–54, 60, 68, 87, 88, 90, 92, 96–98, 102, 103, 111–114, 133, 147
Itemset, 17–19, 21–27, 29–35, 37–43, 45–50, 52, 53, 68, 69, 71, 73–77, 80, 81, 86–88, 94, 98–101
Soporte, 17, 21–25, 29–33, 37–40, 43, 45, 46, 48–54, 59–61, 65, 71–79, 81, 83–92, 94, 95, 97, 100–103, 105, 106, 108, 110–117, 119, 125, 126, 134, 138, 146, 147, 150, 161, 174, 175
TID, Identificador de Transacción, 28, 30, 47
Transacción, 10, 16–31, 33, 34, 37, 41, 42, 47, 50, 51, 58, 67, 72, 79–83, 85, 88, 89, 91, 94, 96–99, 101, 104, 195, 196
Regla de Clasificación (CR), 10, 55, 57, 59–61, 63, 96, 100, 104–106, 108, 109, 111–113, 126
Regla de Clasificación Asociativa (CAR), 59, 61–64, 97, 126, 127, 176

Sistema de Recomendación (RS), 14, 51, 53, 64, 68, 87, 91, 93, 95
Regla de Oportunidad, 68, 90–92, 94, 95
Sistema de Recomendación Web (WRS), 7–10, 12, 14, 21, 79, 80, 85, 87–90

World Wide Web (WWW), 6, 7