

Introducción a la Minería de Datos



José Hernández Orallo
M^a José Ramírez Quintana
Cèsar Ferri Ramírez

INTRODUCCIÓN A LA MINERÍA DE DATOS

INTRODUCCIÓN A LA MINERÍA DE DATOS

**José Hernández Orallo
M.^a José Ramírez Quintana
Cèsar Ferri Ramírez**

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia



Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima
Montevideo • San Juan • San José • Santiago • São Paulo • White Plains

Datos de catalogación bibliográfica

HERNÁNDEZ ORALLO, J.; RAMÍREZ QUINTANA, M. J.;
FERRI RAMÍREZ, C.

INTRODUCCIÓN A LA MINERÍA DE DATOS

PEARSON EDUCACIÓN, S.A.. Madrid, 2004

ISBN: 84-205-4091-9

MATERIA: Informática 681.3

Formato: 195 × 250 mm

Páginas: 680

Última reimpresión, 2005

Todos los derechos reservados. Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de la propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (*arts. 270 y sgts. Código Penal*).

DERECHOS RESERVADOS

© 2004 por PEARSON EDUCACIÓN, S.A.
Ribera del Loira, 28
28042 Madrid (España)

PEARSON PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S.A.

INTRODUCCIÓN A LA MINERÍA DE DATOS

HERNÁNDEZ ORALLO, J.; RAMÍREZ QUINTANA, M. J.; FERRI RAMÍREZ, C

ISBN: 84-205-4091-9

Depósito legal: SE-5115-2006 U.E.

ISBN eBook: 978-84-8322-558-5

Equipo editorial

Editor: David Fayerman Aragón

Técnico editorial: Ana Isabel García Borro

Equipo de producción:

Director: José Antonio Clares

Técnico: José Antonio Hernán

Diseño de cubierta: Equipo de diseño de PEARSON EDUCACIÓN, S.A.

Impreso por: Publidisa

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

Autores Colaboradores

Tomàs Aluja Banet

Universitat Politècnica de Catalunya

Xavier Carreras Pérez

Universitat Politècnica de Catalunya

Emilio S. Corchado Rodríguez

Universidad de Burgos

Mª José del Jesus Díaz

Universidad de Jaén

Pedro Delicado Useros

Universitat Politècnica de Catalunya

Vicent Estruch Gregori

Universitat Politècnica de València

Colin Fyfe

University of Paisley, Reino Unido

José Antonio Gámez Martín

Universidad de Castilla-La Mancha

Ismael García Varea

Universidad de Castilla-La Mancha

Pedro González García

Universidad de Jaén

Francisco Herrera Triguero

Universidad de Granada

Pedro Isasi Viñuela

Universidad Carlos III de Madrid

Lluís Márquez Villodre

Universitat Politècnica de Catalunya

José Miguel Puerta Callejón

Universidad de Castilla-La Mancha

Enrique Romero Merino

Universitat Politècnica de Catalunya

ÍNDICE DE

CONTENIDO

Índice de contenido	VII
Prefacio.....	XIII
Motivación y objetivos del libro.....	XIII
Destinatarios	XV
Organización e itinerarios.....	XVI
Terminología.....	XVIII
Agradecimientos	XVIII

PARTE I: INTRODUCCIÓN

Capítulo 1 ¿Qué es la minería de datos?	3
1.1 Nuevas necesidades.....	3
1.2 El concepto de minería de datos. Ejemplos	5
1.3 Tipos de datos.....	9
1.4 Tipos de modelos	12
1.5 La minería de datos y el proceso de descubrimiento de conocimiento en bases de datos.....	13
1.6 Relación con otras disciplinas.....	14
1.7 Aplicaciones.....	16
1.8 Sistemas y herramientas de minería de datos	18
Capítulo 2 El proceso de extracción de conocimiento	19
2.1 Las fases del proceso de extracción de conocimiento.....	19
2.2 Fase de integración y recopilación.....	21
2.3 Fase de selección, limpieza y transformación	22
2.4 Fase de minería de datos	24
2.5 Fase de evaluación e interpretación.....	35
2.6 Fase de difusión, uso y monitorización.....	39

PARTE II: PREPARACIÓN DE DATOS

Capítulo 3 Recopilación. Almacenes de datos	43
3.1 Introducción.....	44
3.2 Necesidad de los almacenes de datos	46
3.3 Arquitectura de los almacenes de datos	49
3.4 Carga y mantenimiento del almacén de datos.....	59
3.5 Almacenes de datos y minería de datos	62
Capítulo 4 Limpieza y transformación.....	65
4.1 Introducción.....	66
4.2 Integración y limpieza de datos.....	67
4.3 Transformación de atributos. Creación de características.....	78
4.4 Discretización y numerización.....	89
4.5 Normalización de rango: escalado y centrado	93
4.6 Otras transformaciones	94
Capítulo 5 Exploración y selección	97
5.1 Introducción. El contexto de la vista minable	97
5.2 Exploración mediante visualización.....	103
5.3 Sumarización, descripción, generalización y pivotamiento.....	107
5.4 Selección de datos	112
5.5 Lenguajes, primitivas e interfaces de minería de datos	125

PARTE III: TÉCNICAS DE MINERÍA DE DATOS

Capítulo 6 El problema de la extracción de patrones	137
6.1 Introducción.....	137
6.2 Tareas y métodos	139
6.3 Minería de datos y aprendizaje inductivo.....	148
6.4 El lenguaje de los patrones. Expresividad	154
6.5 Breve comparación de métodos	161
Capítulo 7 Modelización estadística paramétrica	165
<i>Tomàs Aluja y Pedro Delicado</i>	
7.1 Concepto de modelización estadística	166
7.2 Modelo de regresión.....	167
7.3 Modelos de regresión sobre componentes incorrelacionados	183
7.4 Modelos de regresión con variables categóricas.....	185
7.5 Análisis de los residuos.....	187
7.6 Ejemplo: aplicación a los datos SERVO	190
7.7 Modelos lineales generalizados	194
7.8 Análisis discriminante.....	203
7.9 Sistemas, aplicabilidad y recomendaciones de uso.....	211

Capítulo 8 Modelización estadística no paramétrica.....	213
<i>Pedro Delicado y Tomàs Aluja</i>	
8.1 Introducción.....	213
8.2 Regresión no paramétrica	215
8.3 Discriminación no paramétrica	229
8.4 Conclusiones, aplicabilidad y sistemas	236
Capítulo 9 Reglas de asociación y dependencia	237
9.1 Introducción.....	237
9.2 Reglas de asociación	239
9.3 Reglas de dependencias	243
9.4 Reglas de asociación multinivel	247
9.5 Reglas de asociación secuenciales.....	249
9.6 Aprendizaje de reglas de asociación con sistemas de minería de datos.....	252
Capítulo 10 Métodos bayesianos	257
<i>José A. Gámez Martín, Ismael García Varea y José M. Puerta Callejón</i>	
10.1 Introducción.....	257
10.2 Teorema de Bayes e hipótesis MAP	259
10.3 Naïve Bayes.....	260
10.4 Redes bayesianas.....	263
10.5 Aprendizaje de redes bayesianas	266
10.6 Clasificadores basados en redes bayesianas	271
10.7 Tratamiento de datos desconocidos	275
10.8 Sistemas	278
Capítulo 11 Árboles de decisión y sistemas de reglas.....	281
11.1 Introducción.....	281
11.2 Sistemas por partición: árboles de decisión para clasificación.....	283
11.3 Sistemas de aprendizaje de reglas por cobertura.....	287
11.4 Poda y reestructuración.....	290
11.5 Árboles de decisión para regresión, agrupamiento o estimación de probabilidades	293
11.6 Aprendizaje de árboles de decisión híbridos	295
11.7 Adaptación para grandes volúmenes de datos.....	295
11.8 Sistemas, aplicabilidad y recomendaciones de uso	297
Capítulo 12 Métodos relacionales y estructurales.....	301
12.1 Introducción.....	301
12.2 Programación lógica y bases de datos.....	304
12.3 Programación lógica inductiva.....	306
12.4 Programación lógica inductiva y minería de datos	312
12.5 Otros métodos relacionales y estructurales	317
12.6 Sistemas	325

Capítulo 13 Redes neuronales artificiales.....	327
<i>Emilio Corchado y Colin Fyfe</i>	
13.1 Introducción.....	327
13.2 El aprendizaje en las redes neuronales artificiales	330
13.3 Aprendizaje supervisado en RNA	330
13.4 Aprendizaje no supervisado en RNA	343
13.5 Sistemas, aplicabilidad y recomendaciones de uso.....	351
Capítulo 14 Máquinas de vectores soporte.....	353
<i>Xavier Carreras, Lluís Márquez y Enrique Romero</i>	
14.1 Introducción.....	353
14.2 Máquinas de vectores soporte para clasificación binaria	356
14.3 Justificación teórica	367
14.4 Aplicaciones de las máquinas de vectores soporte	367
14.5 Extensiones y temas avanzados	375
14.6 Paquetes <i>software</i> y recomendaciones de uso.....	378
Anexo. Optimización con restricciones lineales	381
Capítulo 15 Extracción de conocimiento con algoritmos evolutivos y reglas difusas.....	383
<i>María José del Jesus, Pedro González y Francisco Herrera</i>	
15.1 Introducción.....	383
15.2 Computación evolutiva.....	385
15.3 Algoritmos evolutivos para la extracción de conocimiento	389
15.4 Lógica difusa.....	403
15.5 Lógica difusa en minería de datos	405
15.6 Sistemas evolutivos difusos en minería de datos	409
15.7 Ejemplos	412
15.8 Sistemas <i>software</i>	417
15.9 Conclusiones.....	418
Capítulo 16 Métodos basados en casos y en vecindad	421
<i>Pedro Isasi</i>	
16.1 Introducción.....	421
16.2 Técnicas para agrupamiento.....	428
16.3 Técnicas para clasificación	440
16.4 Métodos de vecindad con técnicas evolutivas	448
16.5 Otros métodos y aplicabilidad	455
PARTE IV: EVALUACIÓN, DIFUSIÓN Y USO DE MODELOS	
Capítulo 17 Técnicas de evaluación	461
17.1 Introducción.....	461
17.2 Evaluación de clasificadores.....	462
17.3 Evaluación de modelos de regresión.....	476

17.4 Comparación de técnicas de aprendizaje.....	477
17.5 Evaluación basada en complejidad de la hipótesis. El principio MDL.....	477
17.6 Evaluación de modelos de agrupamiento	480
17.7 Evaluación de reglas de asociación.....	481
17.8 Otros criterios de evaluación	482
Capítulo 18 Combinación de modelos	485
18.1 Introducción.....	485
18.2 Métodos de construcción de multiclasificadores	487
18.3 Métodos de fusión.....	492
18.4 Métodos híbridos	494
Capítulo 19 Interpretación, difusión y uso de modelos	503
19.1 Introducción.....	503
19.2 Extracción de reglas comprensibles	504
19.3 Visualización posterior	506
19.4 Intercambio y difusión de modelos: estándares de representación	510
19.5 Integración con la toma de decisiones.....	512
19.6 Actualización y revisión de modelos.....	520

PARTE V: MINERÍA DE DATOS COMPLEJOS

Capítulo 20 Minería de datos espaciales, temporales, secuenciales y multimedia.....	525
20.1 Introducción.....	525
20.2 Minería de datos espaciales	526
20.3 Minería de datos temporales	531
20.4 Extracción de patrones secuenciales.....	536
20.5 Minería de datos multimedia	539
Capítulo 21 Minería de web y textos	545
21.1 Introducción.....	545
21.2 Minería web	548
21.3 Minería del contenido de la web.....	551
21.4 Minería de la estructura de la web.....	560
21.5 Minería de uso web.....	563
21.6 Sistemas de minería de web y textos	568

PARTE VI: IMPLANTACIÓN E IMPACTO DE LA MINERÍA DE DATOS

Capítulo 22 Implantación de un programa de minería de datos	573
22.1 Introducción.....	573
22.2 ¿Cuándo empezar? Necesidades y objetivos de negocio.....	575
22.3 Formulación del programa: fases e implantación.....	580
22.4 Integración con las herramientas y proyectos de la organización	586

22.5 Recursos necesarios	590
Capítulo 23 Repercusiones y retos de la minería de datos	597
23.1 Impacto social de la minería de datos	597
23.2 Cuestiones éticas y legales	599
23.3 Escalabilidad. Minería de datos distribuida	601
23.4 Tendencias futuras	605

APÉNDICES

Apéndice [A] Sistemas y herramientas de minería de datos	609
<i>Vicent Estruch Gregori</i>	
Librerías	609
Suites	611
Herramientas específicas	621
Apéndice [B] Datos de ejemplo	625
Tabla resumen	627
Referencias bibliográficas	629
Índice analítico	651

PREFACIO

La minería de datos es un término relativamente moderno que integra numerosas técnicas de análisis de datos y extracción de modelos. Aunque se basa en varias disciplinas, algunas de ellas más tradicionales, se distingue de ellas en la orientación más hacia el fin que hacia el medio, hecho que permite nutrirse de todas ellas sin prejuicios. Y el fin lo merece: ser capaces de extraer patrones, de describir tendencias y regularidades, de predecir comportamientos y, en general, de sacar partido a la información computerizada que nos rodea hoy en día, generalmente heterogénea y en grandes cantidades, permite a los individuos y a las organizaciones comprender y modelar de una manera más eficiente y precisa el contexto en el que deben actuar y tomar decisiones.

Pese a la popularidad del término, la minería de datos es sólo una etapa, si bien la más importante, de lo que se ha venido llamando el proceso de extracción de conocimiento a partir de datos. Este proceso consta de varias fases e incorpora muy diferentes técnicas de los campos del aprendizaje automático, la estadística, las bases de datos, los sistemas de toma de decisión, la inteligencia artificial y otras áreas de la informática y de la gestión de información.

El presente libro describe este proceso de extracción de conocimiento a partir de datos. Se explica, de una manera metodológica y pragmática, el proceso en su conjunto, sus motivaciones y beneficios, estableciendo conexiones con las disciplinas relacionadas y los sistemas con los que debe integrarse. El libro contrasta y despliega, mediante numerosos ejemplos realizados en paquetes de minería de datos, las técnicas que se requieren en cada fase del proceso: técnicas de preparación y almacenes de datos, técnicas propias de extracción de modelos (clasificación, agrupamiento, regresión, asociación, etc.) y técnicas de evaluación y difusión del conocimiento extraído.

Motivación y objetivos del libro

La información reduce nuestra incertidumbre sobre algún aspecto de la realidad y, por tanto, nos permite tomar mejores decisiones. Desde la antigüedad, los sistemas de información se han recopilado y organizado para asistir en la toma de decisiones. En las últimas décadas, el almacenamiento, organización y recuperación de la información se ha automatizado gracias a los sistemas de bases de datos, pero la ubicuidad de la información en formato electrónico ha empezado a ser patente a finales del siglo XX con la irrupción de Internet.

Prácticamente, no existe hoy en día una faceta de la realidad de la cual no se disponga información de manera electrónica, ya sea estructurada, en forma de bases de datos, o no estructurada, en forma textual o hipertextual. Desgraciadamente, gran parte de esta información se genera con un fin concreto y posteriormente no se analiza ni integra con el resto de información o conocimiento del dominio de actuación. Un ejemplo claro podemos encontrarlo en muchas empresas y organizaciones, donde existe una base de datos *transaccional* (el sistema de información de la organización) que sirve para el funcionamiento de las aplicaciones del día a día, pero que raramente se utiliza con fines analíticos. Esto se debe, fundamentalmente, a que no se sabe cómo hacerlo, es decir, no se dispone de las personas y de las herramientas indicadas para ello.

Afortunadamente, la situación ha cambiado de manera significativa respecto a unos años atrás, donde el análisis de datos se realizaba exclusivamente en las grandes corporaciones, gobiernos y entidades bancarias, por departamentos especializados con nombres diversos: planificación y prospectiva, estadística, logística, investigación operativa, etc. Tanto la tecnología informática actual, la madurez de las técnicas de aprendizaje automático y las nuevas herramientas de minería de datos de sencillo manejo, permiten a una pequeña o mediana organización (o incluso un particular) tratar los grandes volúmenes de datos almacenados en las bases de datos (propias de la organización, externas o en la web).

Esto ha multiplicado el número de posibles usuarios, especialmente en el ámbito empresarial y de toma de decisiones, y cada día son más numerosos los particulares y las organizaciones que utilizan alguna herramienta para analizar los datos. Las aplicaciones de la minería de datos, como veremos, no se restringen a la evaluación de clientes, el diseño de campañas de marketing o la predicción de la facturación año tras año, sino que abarca un espectro mucho más amplio de aplicaciones en la empresa y empieza a ser una herramienta cada vez más popular y necesaria en otros ámbitos: análisis de datos científicos, biomedicina, ingeniería, control, administraciones, domótica, etc.

Uno de los aspectos que ha facilitado esta popularización, como hemos dicho, es la aparición de numerosas herramientas y paquetes de “minería de datos”, como, por ejemplo, Clementine de SPSS, Intelligent Miner de IBM, Mine Set de Silicon Graphics, Enterprise Miner de SAS, DM Suite (Darwin) de Oracle, WEKA (de libre distribución), Knowledge Seeker, etc., que posibilitan el uso de técnicas de minería de datos a no especialistas. Sin embargo, para poder sacar mayor provecho a dichas herramientas no es suficiente con el manual de usuario de las mismas. Se hace necesario complementar estas herramientas y manuales específicos con una visión global y generalista de un libro de texto, complementada con ejemplos de uso en varios dominios y con técnicas diferentes. La estimación de qué técnicas son necesarias, cuáles son más apropiadas, qué limitaciones tienen y con qué metodología utilizarlas, es algo que los productos tienden a difuminar pero que un libro de texto puede destacar y tratar en detalle.

El objetivo fundamental de este libro es, por tanto, que el lector sea capaz de utilizar apropiadamente las diversas técnicas existentes en cada una de las fases de la extracción de conocimiento a partir de datos: la *recopilación de datos*, mediante almacenes de datos o de manera directa, la *preparación de datos*, mediante visualización, agregación, limpieza o transformación, la *minería de datos*, mediante técnicas descriptivas o predictivas, la *evaluación y mejora de modelos*, mediante validación cruzada, combinación o análisis de

costes y, finalmente, *la difusión y uso del conocimiento extraído*, mediante estándares de intercambio de conocimiento, XML, modelos convertidos a lenguajes de programación u otras herramientas. En otras palabras, se proporciona una metodología que permite detectar, independientemente de las herramientas *software* concretas que se estén utilizando, qué técnicas son más fiables, más eficientes, más comprensibles y, en definitiva, más pertinentes para un problema en cuestión.

Sin embargo, es importante destacar que no se trata de un libro “publicista” de la minería de datos, cuyo objetivo sea meramente describir sus ventajas o sus posibilidades, como algunos existentes en el ámbito empresarial, sino que se trata de un libro donde se describen los problemas a los que se enfrentan las organizaciones y particulares, la metodología para resolverlos y las técnicas necesarias. Para ello, el libro discurre apoyándose en numerosos ejemplos prácticos y utilizando herramientas de minería de datos como SPSS Clementine, WEKA y otras, ilustrando cada técnica con las diferentes implementaciones que de ella proporciona cada sistema.

Destinatarios

Cualquier profesional o particular puede tener interés o necesidad de analizar sus datos, desde un comercial que desea realizar un agrupamiento de sus clientes, hasta un *broker* que pretende utilizar la minería de datos para analizar el mercado de valores, pasando por un psiquiatra que intenta clasificar sus pacientes en violentos o no violentos a partir de sus comportamientos anteriores. Los problemas y las dudas a la hora de cubrir estas necesidades aparecen si se desconoce por dónde empezar, qué herramientas utilizar, qué técnicas estadísticas o de aprendizaje automático son más apropiadas y qué tipo de conocimiento puedo llegar a obtener y con qué fiabilidad. Este libro intenta resolver estas dudas pero, además, presenta una serie de posibilidades y, por supuesto, de limitaciones, que ni el comercial, el *broker* o el psiquiatra podrían haberse planteado sin conocer las bases de la tecnología de la extracción de conocimiento a partir de datos.

La necesidad o el interés, ante un problema concreto, es una manera habitual de reciclarse y de aprender una nueva tecnología. No obstante, la anticipación o la preparación es otra razón muy importante para adquirir un determinado conocimiento. No es de extrañar, por ello, que cada día existan más materias y contenidos sobre “minería de datos” en estudios de nivel superior, sean grados universitarios o másteres, así como en cursos organizados por empresas acerca de esta tecnología (generalmente centrada en sus propias herramientas). Aparecen por tanto un sinfín de asignaturas obligatorias u optativas en estudios informáticos, ingenieriles, empresariales, de ciencias de la salud y ciencias sociales, y muchos otros, como complemento o como alternativa a las materias más clásicas de estadística que, hasta ahora, estaban más bien centradas en la validación de hipótesis bajo diferentes distribuciones, la teoría de la probabilidad, el análisis multivariante, los estudios correlacionales o el análisis de la varianza.

Por todo lo anterior, el libro va dirigido en primer lugar a profesionales involucrados en el análisis de los sistemas de información o en la toma de decisiones de su organización o de sus clientes. En segundo lugar, el libro se dirige a estudiantes universitarios en titulaciones de ingeniería, informática, empresariales o biomédicas que cursen asignaturas de “análisis de datos”, “extracción de conocimiento”, “aprendizaje automático” o, cómo no,

“minería de datos”, o bien que quieran complementar su formación estadística con la perspectiva del proceso de extracción de conocimiento mediante la generación de modelos a partir de bases de datos.

Esta diversidad de lectores obliga, primero, a asumir muy poco acerca del bagaje previo del lector y, segundo, a permitir diferentes itinerarios según las características y la profundidad que se desee. Los conocimientos exigidos se reducirán al mínimo, con lo que el lector no deberá tener una base estadística ni tampoco un conocimiento extenso de bases de datos relacionales. Para ello, el libro enfoca la extracción de conocimiento como un proceso, donde es necesario desarrollar una serie de fases, para las cuales existen unas técnicas específicas (tanto para la preparación, la minería, la evaluación o el uso de modelos). En este sentido, se centrará en conocer la finalidad de cada técnica, sus posibilidades y limitaciones, comparándolas con otras, más que introducir un análisis profundo, matemático y riguroso de cada una de ellas, que se deja para otros libros más avanzados o especializados. De éstos, se hace medida referencia a lo largo del texto y se pueden encontrar al final bajo el epígrafe de “Referencias bibliográficas”. No obstante, es importante destacar que este libro no es un recetario y que va más allá de la simplificación *facilona* de que a cada problema de análisis de datos se le asocia una técnica para su resolución. Analizar los datos y obtener resultados útiles y reveladores es un proceso que requiere conocimiento sobre un abanico de herramientas, experiencia o metodología de cuándo usar cada una de ellas, así como herramientas informáticas adecuadas.

En definitiva, la necesidad e impulso crecientes del área de la minería de datos, la variedad de destinatarios, y la práctica inexistencia de otros libros de minería de datos en castellano, nos ha sugerido la realización de un texto genérico e integrador, que está dirigido a un amplio espectro de estudiantes y profesionales.

Organización e itinerarios

Pese a tratarse de un libro general sobre la minería de datos, se ha organizado de tal manera que pueda adaptarse en gran medida a las necesidades de lectores diversos o incluso del mismo lector en situaciones diferentes. La organización se ha diseñado con el objetivo de facilitar la asimilación de conceptos, de permitir diferentes aproximaciones y profundizaciones en su lectura, y de ser compatible con un uso posterior del libro como texto de referencia.

El libro se estructura en seis partes bien diferenciadas, que incluyen 23 capítulos que son, en gran medida, autocontenidos, y que se pueden seguir con varios itinerarios:

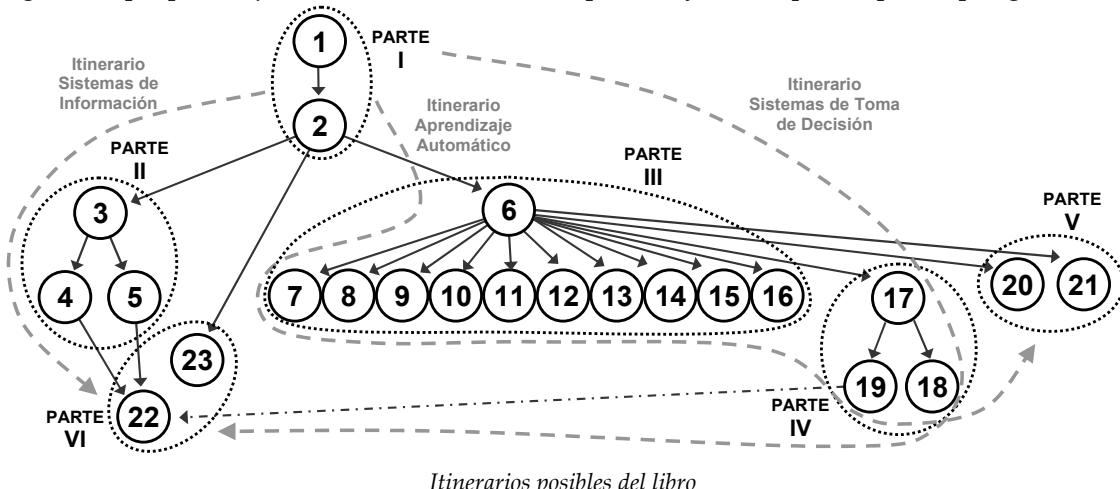
- Parte I. Introducción. En esta parte se introduce la minería de datos, los problemas que viene a resolver y se pone en el contexto de la extracción de conocimiento, describiendo sus fases.
- Parte II. Preparación de datos. Introducido el proceso de extracción de conocimiento, en esta parte se detallan las primeras fases: recopilación (a través o no de un almacén de datos), limpieza y transformación. El enfoque es práctico y apoyado por numerosos ejemplos cortos.
- Parte III. Técnicas de minería de datos. Esta parte es la más extensa y ligeramente más técnica del libro, que trata de describir el funcionamiento y la conveniencia de las técnicas de la fase central del proceso: la fase de minería de datos. Cada capítulo

va acompañado de ejemplos ilustrados en algún sistema o paquete de minería de datos.

- Parte IV. Evaluación, difusión y uso de modelos. En esta parte se aborda la última parte del proceso, la evaluación, difusión, uso y revisión de los modelos extraídos.
- Parte V. Minería de datos complejos. Los capítulos que la integran tratan los problemas complejos de la minería de datos (espaciales, temporales, secuenciales y multimedia) y el análisis de la información no estructurada (textual, HTML, XML). También se aborda someramente el problema de la recuperación de la información (*information retrieval*).
- Parte VI. Implantación e impacto de la minería de datos. En esta parte final se introducen una serie de pautas y recomendaciones para implantar un programa de minería de datos en una organización (recursos humanos y materiales necesarios), pasos, costes, etc., así como unas breves nociones sobre el uso y mal uso de la minería de datos, su impacto y su futuro.

Esta estructura permite, en primer lugar, que se puedan utilizar itinerarios rápidos, orientados a los sistemas de información, para una primera lectura (por ejemplo seleccionando de las partes I, II y VI), o itinerarios para un uso más empresarial, centrado en los sistemas de toma de decisión (por ejemplo seleccionando de las partes I, IV y VI) o itinerarios más centrados en el aprendizaje automático (por ejemplo seleccionando de las partes I, III, IV y V), aparte del itinerario completo. Esta organización y la inclusión de un índice analítico posibilitan también el uso como libro de referencia.

En la figura siguiente se muestran las dependencias existentes entre los diferentes capítulos y se resaltan los tres itinerarios más diferenciados. No obstante, cada lector puede utilizar las dependencias mostradas en el gráfico para construirse su propio itinerario, según sus propios objetivos, sus conocimientos previos y el tiempo de que disponga.



Obviamente, el itinerario total, es decir, la lectura del libro completo en el orden en el que está escrito, es claramente la ruta más comprehensiva y en la que aparecen menos referencias hacia adelante o hacia atrás.

Terminología

Ser pionero tiene sus ventajas e inconvenientes. El hecho de que en el momento de escribir este libro no existieran otros libros de minería de datos en castellano no nos permitió seguir una terminología o un consenso previos que nos fueran válidos a la hora de “fijar” los términos en nuestro idioma.

Se ha intentado utilizar los términos comúnmente aceptados en castellano, evitando, en lo posible, anglicismos innecesarios. En algunos casos existe multiplicidad de términos o algunos son relativamente nuevos en nuestro idioma, como por ejemplo las traducciones “minería de datos” o “prospección de datos” para “*data mining*”, o “máquinas de vectores soporte” o “máquinas de soporte vectorial” para “*support vector machines*”. Debido a esto, en general, al menos la primera aparición del término en castellano irá acompañada del término original en inglés, con el objetivo de evitar confusiones.

En el caso de multiplicidad de términos en el ámbito hispanohablante se utilizará el término más conciliador o preferible (según la R.A.E.) entre los existentes. Así, hablaremos de “almacenes de datos” y no de “bodegas de datos” (“*data-warehouses*”), de “computadoras” más que de “ordenadores”, de “informática” más que de “computación”, etc.

En relación a otros aspectos, se ha intentado minimizar el uso de acrónimos y abreviaturas, así como acompañar cualquier término técnico, matemático o específico con una explicación lo más general posible de su significado.

Agradecimientos

El libro cuenta con la ayuda de prestigiosos colaboradores en áreas específicas de la minería de datos, que han elaborado varios capítulos de este libro y han participado también en la revisión final del texto. Estos colaboradores han enriquecido en gran medida el alcance y visión del libro. Querríamos, por tanto, agradecer en primer lugar la participación de los autores colaboradores de los capítulos 7, 8, 10, 13, 14, 15, 16 y el Apéndice A.

En segundo lugar, hemos de destacar las sugerencias y correcciones realizadas por varios revisores, en especial, la concienzuda revisión reallizada por Ricardo Blanco en gran parte de libro. En tercer lugar, agradecemos a los alumnos de numerosos cursos y asignaturas de minería de datos y extracción de conocimiento, tanto de grado como de postgrado, que han aportado, con sus preguntas, impresiones e incluso experiencias profesionales, una realimentación inestimable para un material que, en gran parte, ha sido la base para este libro. De hecho, la necesidad de este libro se hacía patente cada vez que habíamos de recomendar un libro de texto integrador. Esperamos haber cubierto dicha necesidad.

PARTE I

INTRODUCCIÓN

Esta parte introduce el concepto de minería de datos, sus aplicaciones, las disciplinas relacionadas, el proceso de extracción de conocimiento, del que la minería de datos es sólo una fase, y el resto de fases de este proceso.

CAPÍTULOS

1. **¿Qué es la minería de datos?**
2. **El proceso de extracción de conocimiento**

Capítulo 1

¿QUÉ ES LA MINERÍA DE DATOS?

El primer pensamiento de muchos al oír por primera vez el término “minería de datos” fue la reflexión “nada nuevo bajo el sol”. En efecto, la “minería de datos” no aparece por el desarrollo de tecnologías esencialmente diferentes a las anteriores, sino que se crea, en realidad, por la aparición de nuevas necesidades y, especialmente, por el reconocimiento de un nuevo potencial: el valor, hasta ahora generalmente infravalorado, de la gran cantidad de datos almacenados informáticamente en los sistemas de información de instituciones, empresas, gobiernos y particulares. Los datos pasan de ser un “producto” (el resultado histórico de los sistemas de información) a ser una “materia prima” que hay que explotar para obtener el verdadero “producto elaborado”, el conocimiento; un conocimiento que ha de ser especialmente valioso para la ayuda en la toma de decisiones sobre el ámbito en el que se han recopilado o extraído los datos. Es bien cierto que la estadística es la primera ciencia que considera los datos como su materia prima, pero las nuevas necesidades y, en particular, las nuevas características de los datos (en volumen y tipología) hacen que las disciplinas que integran lo que se conoce como “minería de datos” sean numerosas y heterogéneas.

En este capítulo analizaremos estas nuevas necesidades y posibilidades, precisaremos la definición de “minería de datos” dentro del contexto de la “extracción de conocimiento”, las disciplinas que la forman y destacaremos los tipos de problemas que se desea tratar y los modelos o patrones que se espera obtener.

1.1 Nuevas necesidades

El aumento del volumen y variedad de información que se encuentra informatizada en bases de datos digitales y otras fuentes ha crecido espectacularmente en las últimas décadas. Gran parte de esta información es histórica, es decir, representa transacciones o

situaciones que se han producido. Aparte de su función de “memoria de la organización”, la información histórica es útil para explicar el pasado, entender el presente y predecir la información futura. La mayoría de las decisiones de empresas, organizaciones e instituciones se basan también en información sobre experiencias pasadas extraídas de fuentes muy diversas. Además, ya que los datos pueden proceder de fuentes diversas y pertenecer a diferentes dominios, parece clara la inminente necesidad de analizar los mismos para la obtención de información útil para la organización.

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

Esta forma de actuar es lenta, cara y altamente subjetiva. De hecho, el análisis manual es impracticable en dominios donde el volumen de los datos crece exponencialmente: la enorme abundancia de datos desborda la capacidad humana de comprenderlos sin la ayuda de herramientas potentes. Consecuentemente, muchas decisiones importantes se realizan, no sobre la base de la gran cantidad de datos disponibles, sino siguiendo la propia intuición del usuario al no disponer de las herramientas necesarias. Éste es el principal cometido de la minería de datos: resolver problemas analizando los datos presentes en las bases de datos.

Por ejemplo, supongamos que una cadena de supermercados quiere ampliar su zona de actuación abriendo nuevos locales. Para ello, la empresa analiza la información disponible en sus bases de datos de clientes para determinar el perfil de los mismos y hace uso de diferentes indicadores demográficos que le permiten determinar los lugares más idóneos para los nuevos emplazamientos. La clave para resolver este problema es analizar los datos para identificar el patrón que define las características de los clientes más fieles y que se usa posteriormente para identificar el número de futuros buenos clientes de cada zona.

Hasta no hace mucho, el análisis de los datos de una base de datos se realizaba mediante consultas efectuadas con lenguajes generalistas de consulta, como el SQL, y se producía sobre la base de datos operacional, es decir, junto al procesamiento transaccional en línea (*On-Line Transaction Processing*, OLTP) de las aplicaciones de gestión. No obstante, esta manera de actuar sólo permitía generar información resumida de una manera previamente establecida (generación de informes), poco flexible y, sobre todo, poco escalable a grandes volúmenes de datos. La tecnología de bases de datos ha respondido a este reto con una nueva arquitectura surgida recientemente: el almacén de datos (*data warehouse*). Se trata de un repositorio de fuentes heterogéneas de datos, integrados y organizados bajo un esquema unificado para facilitar su análisis y dar soporte a la toma de decisiones. Esta tecnología incluye operaciones de procesamiento analítico en línea (*On-Line Analytical Processing*, OLAP), es decir, técnicas de análisis como pueden ser el resumen, la consolidación o la agregación, así como la posibilidad de ver la información desde distintas perspectivas.

Sin embargo, a pesar de que las herramientas OLAP soportan cierto análisis descriptivo y de sumarización que permite transformar los datos en otros datos agregados o cruzados de manera sofisticada, no generan reglas, patrones, pautas, es decir, conocimiento que pueda ser aplicado a otros datos. Sin embargo, en muchos contextos, como los negocios, la medicina o la ciencia, los datos por sí solos tienen un valor relativo. Lo que de verdad es interesante es el conocimiento que puede inferirse a partir de los datos y, más aún, la capacidad de poder usar este conocimiento. Por ejemplo, podemos saber estadísticamente que el 10 por ciento de los ancianos padecen Alzheimer. Esto puede ser útil, pero seguramente es mucho más útil tener un conjunto de reglas que a partir de los antecedentes, los hábitos y otras características del individuo nos digan si un paciente tendrá o no Alzheimer.

Existen otras herramientas analíticas que han sido empleadas para analizar los datos y que tienen su origen en la estadística, algo lógico teniendo en cuenta que la materia prima de esta disciplina son precisamente los datos. Aunque algunos paquetes estadísticos son capaces de inferir patrones a partir de los datos (utilizando modelización estadística paramétrica o no paramétrica), el problema es que resultan especialmente críticos para los no estadísticos, generalmente no funcionan bien para la talla de las bases de datos actuales (cientos de tablas, millones de registros, talla de varios gigabytes y una alta dimensionalidad) y algunos tipos de datos frecuentes en ellos (atributos nominales con muchos valores, datos textuales, multimedia, etc.), y no se integran bien con los sistemas de información. No obstante, sería injusto no reconocer que la estadística es, en cierto modo, la “madre” de la minería de datos. Ésta se vio, en un principio, como una hija rebelde, que tenía un carácter peyorativo para los estadísticos pero que poco a poco fue ganando un prestigio y una concepción como disciplina integradora más que disgregadora. Más adelante trataremos de esclarecer la relación existente entre la minería de datos y la estadística.

Todos estos problemas y limitaciones de las aproximaciones clásicas han hecho surgir la necesidad de una nueva generación de herramientas y técnicas para soportar la extracción de conocimiento útil desde la información disponible, y que se engloban bajo la denominación de minería de datos. La minería de datos se distingue de las aproximaciones anteriores porque no obtiene información extensional (datos) sino intensional (conocimiento) y, además, el conocimiento no es, generalmente, una parametrización de ningún modelo preestablecido o intuido por el usuario, sino que es un modelo novedoso y original, extraído completamente por la herramienta. El resultado de la minería de datos son conjuntos de reglas, ecuaciones, árboles de decisión, redes neuronales, grafos probabilísticos..., los cuales pueden usarse para, por ejemplo, responder a cuestiones como *¿existe un grupo de clientes que se comporta de manera diferenciada?*, *¿qué secuenciación de tratamientos puede ser más efectiva para este nuevo síndrome?*, *¿existen asociaciones entre los factores de riesgo para realizar un seguro de automóvil?*, *¿cómo califico automáticamente los mensajes de correo entre más o menos susceptibles de ser spam?*

1.2 El concepto de minería de datos. Ejemplos

En [Witten & Frank 2000] se define la minería de datos como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos. Es decir, la tarea fundamental de la minería de

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

Por lo tanto, dos son los retos de la minería de datos: por un lado, trabajar con grandes volúmenes de datos, procedentes mayoritariamente de sistemas de información, con los problemas que ello conlleva (ruido, datos ausentes, intratabilidad, volatilidad de los datos...), y por el otro usar técnicas adecuadas para analizar los mismos y extraer conocimiento novedoso y útil. En muchos casos la utilidad del conocimiento minado está íntimamente relacionada con la comprensibilidad del modelo inferido. No debemos olvidar que, generalmente, el usuario final no tiene por qué ser un experto en las técnicas de minería de datos, ni tampoco puede perder mucho tiempo interpretando los resultados. Por ello, en muchas aplicaciones es importante hacer que la información descubierta sea más comprensible por los humanos (por ejemplo, usando representaciones gráficas, convirtiendo los patrones a lenguaje natural o utilizando técnicas de visualización de los datos).

De una manera simplista pero ambiciosa, podríamos decir que el objetivo de la minería de datos es convertir datos en conocimiento. Este objetivo no es sólo ambicioso sino muy amplio. Por tanto, de momento, y para ayudar al lector a refinar su concepto de minería de datos, presentamos varios ejemplos muy sencillos.

1.2.1 Ejemplo 1: análisis de créditos bancarios

El primer ejemplo pertenece al ámbito de la banca. Un banco por Internet desea obtener reglas para predecir qué personas de las que solicitan un crédito no lo devuelven. La entidad bancaria cuenta con los datos correspondientes a los créditos concedidos con anterioridad a sus clientes (cuantía del crédito, duración en años...) y otros datos personales como el salario del cliente, si posee casa propia, etc. Algunos registros de clientes de esta base de datos se muestran en la Tabla 1.1.

Tabla 1.1. Datos para un análisis de riesgo en créditos bancarios.

A partir de éstos, las técnicas de minería de datos podrían sintetizar algunas reglas, como por ejemplo:

SI Cuentas-Morosas > 0 **ENTONCES** Devuelve-crédito = no
SI Cuentas-Morosas = 0 **Y** [(Salario > 2.500) **O** (D-crédito > 10)] **ENTONCES**
Devuelve-crédito = sí

El banco podría entonces utilizar estas reglas para determinar las acciones a realizar en el trámite de los créditos: si se concede o no el crédito solicitado, si es necesario pedir avales especiales, etc.

1.2.2 Ejemplo 2: análisis de la cesta de la compra

Éste es uno de los ejemplos más típicos de minería de datos. Un supermercado quiere obtener información sobre el comportamiento de compra de sus clientes. Piensa que de esta forma puede mejorar el servicio que les ofrece: reubicación de los productos que se suelen comprar juntos, localizar el emplazamiento idóneo para nuevos productos, etc. Para ello dispone de la información de los productos que se adquieren en cada una de las compras o cestas. Un fragmento de esta base de datos se muestra en la Tabla 1.2.

Idcesta	Huevos	Aceite	Pañales	Vino	Leche	Mantequilla	Salmón	Lechugas	...
1	sí	no	no	sí	no	sí	sí	sí	...
2	no	sí	no	no	sí	no	no	sí	...
3	no	no	sí	no	sí	no	no	no	...
4	no	sí	sí	no	sí	no	no	no	...
5	sí	sí	no	no	no	sí	no	sí	...
6	sí	no	no	sí	sí	sí	sí	no	...
7	no	no	no	no	no	no	no	no	...
8	sí	sí	sí	sí	sí	sí	sí	no	...
...

Tabla 1.2. Datos de las cestas de la compra.

Analizando estos datos el supermercado podría encontrar, por ejemplo, que el 100 por cien de las veces que se compran pañales también se compra leche, que el 50 por ciento de las veces que se compran huevos también se compra aceite o que el 33 por ciento de las veces que se compra vino y salmón entonces se compran lechugas. También se puede analizar cuáles de estas asociaciones son frecuentes, porque una asociación muy estrecha entre dos productos puede ser poco frecuente y, por tanto, poco útil.

1.2.3 Ejemplo 3: determinar las ventas de un producto

Una gran cadena de tiendas de electrodomésticos desea optimizar el funcionamiento de su almacén manteniendo un *stock* de cada producto suficiente para poder servir rápidamente el material adquirido por sus clientes. Para ello, la empresa dispone de las ventas efectuadas cada mes del último año de cada producto, tal y como se refleja en la Tabla 1.3.

Producto	mes-12	...	mes-4	mes-3	mes-2	mes-1
televisor plano 30' Phlipis	20	...	52	14	139	74
vídeo-dvd-recorder Miesens	11	...	43	32	26	59
discman mp3 LJ	50	...	61	14	5	28
frigorífico no frost Jazzussi	3	...	21	27	1	49
microondas con grill Sanson	14	...	27	2	25	12
...

Tabla 1.3. Ventas mensuales durante el último año.

Esta información permite a la empresa generar un modelo para predecir cuáles van a ser las ventas de cada producto en el siguiente mes en función de las ventas realizadas en los meses anteriores, y efectuar así los pedidos necesarios a sus proveedores para disponer del stock necesario para hacer frente a esas ventas.

1.2.4 Ejemplo 4: determinar grupos diferenciados de empleados

El departamento de recursos humanos de una gran empresa desea categorizar a sus empleados en distintos grupos con el objetivo de entender mejor su comportamiento y tratarlos de manera adecuada. Para ello dispone en sus bases de datos de información sobre los mismos (sueldo, estado civil, si tiene coche, número de hijos, si su casa es propia o de alquiler, si está sindicado, número de bajas al año, antigüedad y sexo). La Tabla 1.4 muestra algunos de los registros de su base de datos.

Id	Sueldo	Casado	Coche	Hijos	Alq/prop	Sindicado	Bajas/año	Antigüedad	Sexo
1	1.000	Sí	No	0	Alquiler	No	7	15	H
2	2.000	No	Sí	1	Alquiler	Sí	3	3	M
3	1.500	Sí	Sí	2	Prop	Sí	5	10	H
4	3.000	Sí	Sí	1	Alquiler	No	15	7	M
5	1.000	Sí	Sí	0	Prop	Sí	1	6	H
6	4.000	No	Sí	0	Alquiler	Sí	3	16	M
7	2.500	No	No	0	Alquiler	Sí	0	8	H
8	2.000	No	Sí	0	Prop	Sí	2	6	M
9	2.000	Sí	Sí	3	Prop	No	7	5	H
10	3.000	Sí	Sí	2	Prop	No	1	20	H
11	5.000	No	No	0	Alquiler	No	2	12	M
12	800	Sí	Sí	2	Prop	No	3	1	H
13	2.000	No	No	0	Alquiler	No	27	5	M
14	1.000	No	Sí	0	Alquiler	Sí	0	7	H
15	8.00	No	Sí	0	Alquiler	No	3	2	H
...

Tabla 1.4. Datos de los empleados.

Un sistema de minería de datos podría obtener tres grupos con la siguiente descripción:

Grupo 1:
 Sueldo: 1.535,2€
 Casado: No -> 0,777
 Sí -> 0,223
 Coche: No -> 0,82
 Sí -> 0,18
 Hijos: 0,05
 Alq/Prop: Alquiler -> 0,99
 Propia -> 0,01
 Sindic.: No -> 0,8
 Sí -> 0,2
 Bajas/Año: 8,3
 Antigüedad: 8,7
 Sexo: H -> 0,61
 M -> 0,39

Grupo 2:
 Sueldo: 1.428,7€
 Casado: No -> 0,98
 Sí -> 0,02
 Coche: No -> 0,01
 Sí -> 0,99
 Hijos: 0,3
 Alq/Prop: Alquiler -> 0,75
 Propia -> 0,25
 Sindic.: Sí -> 1,0
 Bajas/Año: 2,3
 Antigüedad: 8
 Sexo: H -> 0,25
 M -> 0,75

Grupo 3:
 Sueldo: 1.233,8€
 Casado: Sí -> 1,0
 Coche: No -> 0,05
 Sí -> 0,95
 Hijos: 2,3
 Alq/Prop: Alquiler -> 0,17
 Propia -> 0,83
 Sindic.: No -> 0,67
 Sí -> 0,33
 Bajas/Año: 5,1
 Antigüedad: 8,1
 Sexo: H -> 0,83
 M -> 0,17

Estos grupos podrían ser interpretados por el departamento de recursos humanos de la siguiente manera:

- Grupo 1: sin hijos y con vivienda de alquiler. Poco sindicados. Muchas bajas.
- Grupo 2: sin hijos y con coche. Muy sindicados. Pocas bajas. Normalmente son mujeres y viven en casas de alquiler.
- Grupo 3: con hijos, casados y con coche. Mayoritariamente hombres propietarios de su vivienda. Poco sindicados.

1.3 Tipos de datos

Llegados a este punto surge una pregunta obligada, ¿a qué tipo de datos puede aplicarse la minería de datos? En principio, ésta puede aplicarse a cualquier tipo de información, siendo las técnicas de minería diferentes para cada una de ellas. En esta sección damos una breve introducción a algunos de estos tipos. En concreto, vamos a diferenciar entre datos estructurados provenientes de bases de datos relacionales, otros tipos de datos estructurados en bases de datos (espaciales, temporales, textuales y multimedia) y datos no estructurados provenientes de la web o de otros tipos de repositorios de documentos.

1.3.1 Bases de datos relacionales

Una base de datos relacional es una colección de relaciones (tablas). Cada tabla consta de un conjunto de atributos (columnas o campos) y puede contener un gran número de tuplas (registros o filas). Cada tupla representa un objeto, el cual se describe a través de los valores de sus atributos y se caracteriza por poseer una clave única o primaria que lo identifica. Por ejemplo, la Figura 1.1 ilustra una base de datos con dos relaciones: *empleado* y *departamento*. La relación *empleado* tiene seis atributos: el identificador o clave primaria (*IdE*), el nombre del empleado (*Enombre*), su sueldo (*Sueldo*), su edad (*Edad*), su sexo (*Sexo*) y el departamento en el que trabaja (*IdD*), y la relación *departamento* tiene tres atributos: su identificador o clave primaria (*IdD*), el nombre (*Dnombre*) y su director (*Director*). Una relación puede además tener claves ajenas, es decir, atributos que hagan referencia a otra relación, como por ejemplo el sexto atributo de la relación *empleado*, *IdD*, que hace referencia (por valor) al *IdD* de *departamento*.

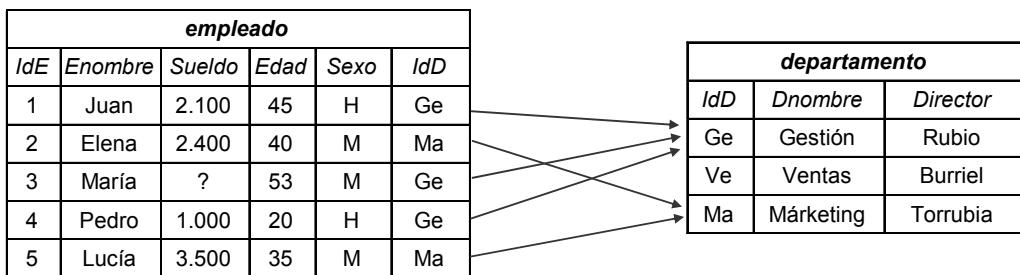


Figura 1.1. Base de datos relacional.

Una de las principales características de las bases de datos relacionales es la existencia de un esquema asociado, es decir, los datos deben seguir una estructura y son, por tanto, estructurados. Así, el esquema de la base de datos del ejemplo indica que las tuplas de la relación *empleado* tienen un valor para cada uno de sus seis atributos y las de la relación

departamento constan de tres valores, además de indicar los tipos de datos (numérico, cadena de caracteres, etc.).

La integridad de los datos se expresa a través de las restricciones de integridad. Éstas pueden ser de dominio (restringen el valor que puede tomar un atributo respecto a su dominio y si puede tomar valores nulos o no), de identidad (por ejemplo la clave primaria tiene que ser única) y referencial (los valores de las claves ajenas se deben corresponder con uno y sólo un valor de la tabla referenciada).

Como se ha comentado en la Sección 1.1, la obtención de información desde una base de datos relacional se ha resuelto tradicionalmente a través de lenguajes de consulta especialmente diseñados para ello, como SQL. La Figura 1.2 muestra una consulta típica SQL sobre una relación *empleado* que lista la media de edad de todos los empleados de una empresa cuyo sueldo es mayor de 2.000 euros, agrupado por departamento.

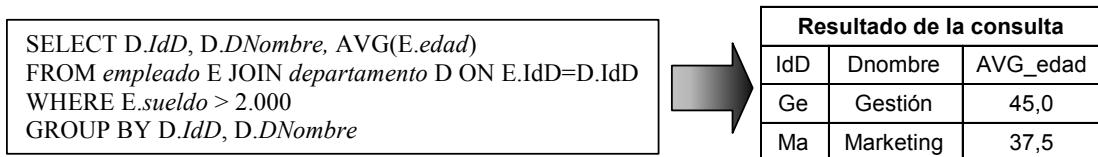


Figura 1.2. Consulta SQL.

Aunque las bases de datos relacionales (recogidas o no en un almacén de datos, normalizadas o estructuradas de una manera multidimensional) son la fuente de datos para la mayoría de aplicaciones de minería de datos, muchas técnicas de minería de datos no son capaces de trabajar con *toda* la base de datos, sino que sólo son capaces de tratar con una sola tabla a la vez. Lógicamente, mediante una consulta (por ejemplo en SQL, en una base de datos relacional tradicional, o con herramientas y operadores más potentes, en los almacenes de datos) podemos combinar en una sola tabla o *vista minable* aquella información de varias tablas que requiramos para cada tarea concreta de minería de datos. Por tanto, la presentación tabular, también llamada atributo-valor, es la más utilizada por las técnicas de minería de datos.

En esta presentación tabular, es importante conocer los tipos de los atributos y, aunque en bases de datos existen muchos tipos de datos (enteros, reales, fechas, cadenas de texto, etc.), desde el punto de vista de las técnicas de minería de datos más habituales nos interesa distinguir sólo entre dos tipos, numéricos y categóricos. Para tratar otras representaciones más complejas, como las cadenas de caracteres, los tipos textuales “memo”, los vectores, y otras muchas, harán falta técnicas específicas.

- Los atributos *numéricos* contienen valores enteros o reales. Por ejemplo, atributos como el *salario* o la *edad* son numéricos.
- Los atributos *categóricos* o *nominales* toman valores en un conjunto finito y preestablecido de categorías. Por ejemplo, atributos como el *sexo* (H, M), el nombre del departamento (Gestión, Marketing, Ventas) son categóricos.

Incluso sólo considerando estos dos tipos de datos, no todas las técnicas de minería de datos son capaces de trabajar con ambos tipos. Este hecho puede requerir la aplicación de un proceso previo de transformación/preparación de los datos del que hablaremos en el Capítulo 4.

1.3.2 Otros tipos de bases de datos

Aunque las bases de datos relacionales son, con gran diferencia, las más utilizadas hoy en día, existen aplicaciones que requieren otros tipos de organización de la información. Otros tipos de bases de datos que contienen datos complejos son:

- Las bases de datos espaciales contienen información relacionada con el espacio físico en un sentido amplio (una ciudad, una región montañosa, un atlas cerebral...). Estas bases de datos incluyen datos geográficos, imágenes médicas, redes de transporte o información de tráfico, etc., donde las relaciones espaciales son muy relevantes. La minería de datos sobre estas bases de datos permite encontrar patrones entre los datos, como por ejemplo las características de las casas en una zona montañosa, la planificación de nuevas líneas de metro en función de la distancia de las distintas áreas a las líneas existentes, etc.
- Las bases de datos temporales almacenan datos que incluyen muchos atributos relacionados con el tiempo o en el que éste es muy relevante. Estos atributos pueden referirse a distintos instantes o intervalos temporales. En este tipo de bases de datos las técnicas de minería de datos pueden utilizarse para encontrar las características de la evolución o las tendencias del cambio de distintas medidas o valores de la base de datos.
- Las bases de datos documentales contienen descripciones para los objetos (documentos de texto) que pueden ir desde las simples palabras clave a los resúmenes. Estas bases de datos pueden contener documentos no estructurados (como una biblioteca digital de novelas), semi-estructurados (si se puede extraer la información por partes, con índices, etc.) o estructurados (como una base de datos de fichas bibliográficas). Las técnicas de minería de datos pueden utilizarse para obtener asociaciones entre los contenidos, agrupar o clasificar objetos textuales. Para ello, los métodos de minería se integran con otras técnicas de recuperación de información y con la construcción o uso de jerarquías específicas para datos textuales, como los diccionarios y los tesauros.
- Las bases de datos multimedia almacenan imágenes, audio y vídeo. Soportan objetos de gran tamaño ya que, por ejemplo, los vídeos pueden necesitar varios gigabytes de capacidad para su almacenamiento. Para la minería de estas bases de datos también es necesario integrar los métodos de minería con técnicas de búsqueda y almacenamiento.

Las bases de datos objetuales y las objeto-relacionales son aproximaciones generales a la gestión de la información y, por tanto, pueden utilizarse para los mismos usos que las relacionales o para algunas de las bases de datos especiales que acabamos de ver.

1.3.3 La World Wide Web

La World Wide Web es el repositorio de información más grande y diverso de los existentes en la actualidad. Por ello, hay gran cantidad de datos en la web de los que se puede extraer conocimiento relevante y útil. Éste es precisamente el reto al que se enfrenta la “minería web”. Minar la web no es un problema sencillo, debido a que muchos de los datos son no estructurados o semi-estructurados, a que muchas páginas web contienen

datos multimedia (texto, imágenes, vídeo y/o audio), y a que estos datos pueden residir en diversos servidores o en archivos (como los que contienen los *logs*). Otros aspectos que dificultan la minería web son cómo determinar a qué páginas debemos acceder y cómo seleccionar la información que va a ser útil para extraer conocimiento. Toda esta diversidad hace que la minería web se organice en torno a tres categorías:

- minería del contenido, para encontrar patrones de los datos de las páginas web.
- minería de la estructura, entendiendo por estructura los hipervínculos y URLs.
- minería del uso que hace el usuario de las páginas web (navegación).

La mayoría de las técnicas de la Parte III de este libro se enfocan a bases de datos tradicionales. La Parte V se dedica a presentar técnicas específicas o adaptar las tradicionales para abordar los otros tipos de datos comentados: espaciales, temporales, documentales y textuales, multimedia y la web.

1.4 Tipos de modelos

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

En la práctica, los modelos pueden ser de dos tipos: *predictivos* y *descriptivos*. Los modelos predictivos pretenden estimar valores futuros o desconocidos de variables de interés, que denominamos *variables objetivo* o *dependientes*, usando otras variables o campos de la base de datos, a las que nos referiremos como *variables independientes* o *predictivas*. Por ejemplo, un modelo predictivo sería aquel que permite estimar la demanda de un nuevo producto en función del gasto en publicidad.

Los modelos descriptivos, en cambio, identifican patrones que explican o resumen los datos, es decir, sirven para explorar las propiedades de los datos examinados, no para predecir nuevos datos. Por ejemplo, una agencia de viaje desea identificar grupos de personas con unos mismos gustos, con el objeto de organizar diferentes ofertas para cada grupo y poder así remitirles esta información; para ello analiza los viajes que han realizado sus clientes e infiere un modelo descriptivo que caracteriza estos grupos.

Como veremos en el Capítulo 2 y, con más detalle, en el Capítulo 6, algunas tareas de minería de datos que producen modelos predictivos son la clasificación y la regresión, y las que dan lugar a modelos descriptivos son el agrupamiento, las reglas de asociación y el análisis correlacional. Por ejemplo, los problemas planteados en los ejemplos 1 y 3 de la Sección 1.2 se resolverían mediante modelos predictivos, mientras que los de los ejemplos 2 y 4 usarían modelos descriptivos.

Cada tarea puede ser realizada usando distintas técnicas. Por ejemplo, los modelos inferidos por los árboles de decisión y las redes neuronales (por citar dos técnicas de las más conocidas y utilizadas) pueden inferir modelos predictivos. Igualmente, para una misma técnica se han desarrollado diferentes algoritmos que difieren en la forma y criterios concretos con los que se construye el modelo. Todas estas cuestiones serán abordadas en

capítulos sucesivos. Así, el Capítulo 2 presenta una relación de las técnicas de generación de modelos más significativas. Asimismo, en la Parte III del libro se hace un estudio más profundo de algunas de estas técnicas y de los diferentes algoritmos desarrollados en ellas.

1.5 La minería de datos y el proceso de descubrimiento de conocimiento en bases de datos

Existen términos que se utilizan frecuentemente como sinónimos de la minería de datos. Uno de ellos se conoce como “análisis (inteligente) de datos” (véase por ejemplo [Berthold & Hand 2003]), que suele hacer un mayor hincapié en las técnicas de análisis estadístico. Otro término muy utilizado, y el más relacionado con la minería de datos, es la extracción o “descubrimiento de conocimiento en bases de datos” (*Knowledge Discovery in Databases, KDD*). De hecho, en muchas ocasiones ambos términos se han utilizado indistintamente, aunque existen claras diferencias entre los dos. Así, últimamente se ha usado el término KDD para referirse a un proceso que consta de una serie de fases, mientras que la minería de datos es sólo una de estas fases.

En [Fayyad et al. 1996a] se define el KDD como “el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos”. En esta definición se resumen cuáles deben ser las propiedades deseables del conocimiento extraído:

- válido: hace referencia a que los patrones deben seguir siendo precisos para datos nuevos (con un cierto grado de certidumbre), y no sólo para aquellos que han sido usados en su obtención.
- novedoso: que aporte algo desconocido tanto para el sistema y preferiblemente para el usuario.
- potencialmente útil: la información debe conducir a acciones que reporten algún tipo de beneficio para el usuario.
- comprensible: la extracción de patrones no comprensibles dificulta o imposibilita su interpretación, revisión, validación y uso en la toma de decisiones. De hecho, una información incomprendible no proporciona conocimiento (al menos desde el punto de vista de su utilidad).

Como se deduce de la anterior definición, el KDD es un proceso complejo que incluye no sólo la obtención de los modelos o patrones (el objetivo de la minería de datos), sino también la evaluación y posible interpretación de los mismos, tal y como se refleja en la Figura 1.3.

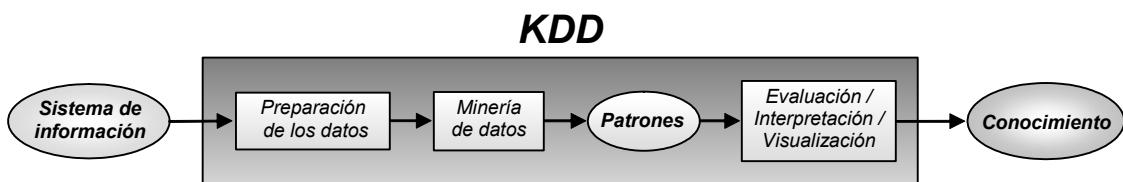


Figura 1.3. Proceso de KDD.

Así, los sistemas de KDD permiten la selección, limpieza, transformación y proyección de los datos; analizar los datos para extraer patrones y modelos adecuados; evaluar e

interpretar los patrones para convertirlos en conocimiento; consolidar el conocimiento resolviendo posibles conflictos con conocimiento previamente extraído; y hacer el conocimiento disponible para su uso. Esta definición del proceso clarifica la relación entre el KDD y la minería de datos: el KDD es el proceso global de descubrir conocimiento útil desde las bases de datos mientras que la minería de datos se refiere a la aplicación de los métodos de aprendizaje y estadísticos para la obtención de patrones y modelos. Al ser la fase de generación de modelos, comúnmente se asimila KDD con minería de datos. Además, las connotaciones de aventura y de dinero fácil del término “minería de datos” han hecho que éste se use como identificador del área, especialmente en el ámbito empresarial.

1.6 Relación con otras disciplinas

La minería de datos es un campo multidisciplinario que se ha desarrollado en paralelo o como prolongación de otras tecnologías. Por ello, la investigación y los avances en la minería de datos se nutren de los que se producen en estas áreas relacionadas.

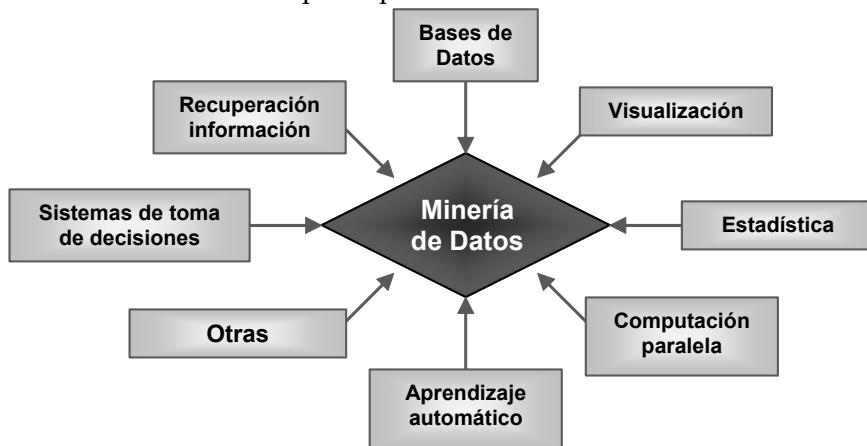


Figura 1.4. Disciplinas que contribuyen a la minería de datos.

Podemos destacar como disciplinas más influyentes las siguientes (Figura 1.4):

- **las bases de datos:** conceptos como los almacenes de datos y el procesamiento analítico en línea (OLAP) tienen una gran relación con la minería de datos, aunque en este último caso no se trata de obtener informes avanzados a base de agregar los datos de cierta manera compleja pero predefinida (como incluyen muchas herramientas de *business intelligence*, presentes en sistemas de gestión de bases de datos comerciales), sino de extraer conocimiento novedoso y comprensible. Las técnicas de indexación¹ y de acceso eficiente a los datos son muy relevantes para el diseño de algoritmos eficientes de minería de datos.
- **la recuperación de información (information retrieval, IR):** consiste en obtener información desde datos textuales, por lo que su desarrollo histórico se ha basado en el uso efectivo de bibliotecas (recientemente digitales) y en la búsqueda por Internet. Una tarea típica es encontrar documentos a partir de palabras claves, lo cual

¹ Es el término correcto para “indexing”, aunque frecuentemente se utiliza el término “indexación”.

puede verse como un proceso de clasificación de los documentos en función de estas palabras clave. Para ello se usan medidas de similitud entre los documentos y la consulta. Muchas de estas medidas se han empleado en aplicaciones más generales de minería de datos.

- **la estadística:** esta disciplina ha proporcionado muchos de los conceptos, algoritmos y técnicas que se utilizan en minería de datos, como por ejemplo, la media, la varianza, las distribuciones, el análisis univariante y multivariante, la regresión lineal y no lineal, la teoría del muestreo, la validación cruzada, la modelización paramétrica y no paramétrica, las técnicas bayesianas, y un largo etcétera. De hecho, algunos paquetes de análisis estadístico se comercializan como herramientas de minería de datos.
- **el aprendizaje automático:** ésta es el área de la inteligencia artificial que se ocupa de desarrollar algoritmos (y programas) capaces de aprender, y constituye, junto con la estadística, el corazón del análisis inteligente de los datos. Los principios seguidos en el aprendizaje automático y en la minería de datos son los mismos: la máquina aprende un modelo a partir de ejemplos y lo usa para resolver el problema.
- **los sistemas para la toma de decisión:** son herramientas y sistemas informatizados que asisten a los directivos en la resolución de problemas y en la toma de decisiones. El objetivo es proporcionar la información necesaria para realizar decisiones efectivas en el ámbito empresarial o en tareas de diagnóstico (por ejemplo en medicina). Herramientas como el análisis ROC (ver Capítulo 17.) o los mismos árboles de decisión provienen de esta área.
- **la visualización de datos:** el uso de técnicas de visualización permite al usuario descubrir, intuir o entender patrones que serían más difíciles de “ver” a partir de descripciones matemáticas o textuales de los resultados. Existen técnicas de visualización, como, por ejemplo, las gráficas (diagramas de barras, gráficas de dispersión, histogramas, etc.), las icónicas (basadas en figuras, colores, etc.), las basadas en píxeles (cada dato se representa como un único píxel), las jerárquicas (dividiendo el área de representación en regiones dependiendo de los datos) y muchas otras.
- **la computación paralela y distribuida:** actualmente, muchos sistemas de bases de datos comerciales incluyen tecnologías de procesamiento paralelo, distribuido o de computación en *grid*. En estos sistemas el coste computacional de las tareas más complejas de minería de datos se reparte entre diferentes procesadores o computadores. Su éxito se debe en parte a la explosión de los almacenes de datos (su adaptación distribuida) y de la minería de datos, en los que las prestaciones de los algoritmos de consulta son críticas. Una de las principales ventajas del procesamiento paralelo es precisamente la escalabilidad de los algoritmos, lo que lo hace idóneo para estas aplicaciones.
- **otras disciplinas:** dependiendo del tipo de datos a ser minados o del tipo de aplicación, la minería de datos usa también técnicas de otras disciplinas como el lenguaje natural, el análisis de imágenes, el procesamiento de señales, los gráficos por computadora, etc.

1.7 Aplicaciones

La integración de las técnicas de minería de datos en las actividades del día a día se está convirtiendo en algo habitual. Los negocios de la distribución y la publicidad dirigida han sido tradicionalmente las áreas en las que más se han empleado los métodos de minería, ya que han permitido reducir costes o aumentar la receptividad de ofertas. Pero éstas no son las únicas áreas a las que se pueden aplicar. De hecho, podemos encontrar ejemplos en todo tipo de aplicaciones: financieras, seguros, científicas (medicina, farmacia, astronomía, psicología, etc.), políticas económicas, sanitarias o demográficas, educación, policiales, procesos industriales y un largo etcétera.

Siendo un poco más concretos, a continuación incluimos una lista (que en modo alguno pretendemos que sea exhaustiva) de ejemplos en algunas de las áreas antes mencionadas para ilustrar en qué ámbitos se puede usar la minería de datos.

- Aplicaciones financieras y banca:
 - Obtención de patrones de uso fraudulento de tarjetas de crédito.
 - Determinación del gasto en tarjeta de crédito por grupos.
 - Cálculo de correlaciones entre indicadores financieros.
 - Identificación de reglas de mercado de valores a partir de históricos.
 - Análisis de riesgos en créditos.
- Análisis de mercado, distribución y, en general, comercio:
 - Análisis de la cesta de la compra (compras conjuntas, secuenciales, ventas cruzadas, señuelos, etc.).
 - Evaluación de campañas publicitarias.
 - Análisis de la fidelidad de los clientes. Reducción de fuga.
 - Segmentación de clientes.
 - Estimación de stocks, de costes, de ventas, etc.
- Seguros y salud privada:
 - Determinación de los clientes que podrían ser potencialmente caros.
 - Análisis de procedimientos médicos solicitados conjuntamente.
 - Predicción de qué clientes contratan nuevas pólizas.
 - Identificación de patrones de comportamiento para clientes con riesgo.
 - Identificación de comportamiento fraudulento.
 - Predicción de los clientes que podrían ampliar su póliza para incluir procedimientos extras (dentales, ópticos...).
- Educación:
 - Selección o captación de estudiantes.
 - Detección de abandonos y de fracaso.
 - Estimación del tiempo de estancia en la institución.
- Procesos industriales:
 - Extracción de modelos sobre comportamiento de compuestos.
 - Detección de piezas con trabas. Modelos de calidad.

- Predicción de fallos y accidentes.
- Estimación de composiciones óptimas en mezclas.
- Extracción de modelos de coste.
- Extracción de modelos de producción.
- Medicina:
 - Identificación de patologías. Diagnóstico de enfermedades.
 - Detección de pacientes con riesgo de sufrir una patología concreta.
 - Gestión hospitalaria y asistencial. Predicciones temporales de los centros asistenciales para el mejor uso de recursos, consultas, salas y habitaciones.
 - Recomendación priorizada de fármacos para una misma patología.
- Biología, bioingeniería y otras ciencias:
 - Análisis de secuencias de genes.
 - Análisis de secuencias de proteínas.
 - Predecir si un compuesto químico causa cáncer.
 - Clasificación de cuerpos celestes.
 - Predicción de recorrido y distribución de inundaciones.
 - Modelos de calidad de aguas, indicadores ecológicos.
- Telecomunicaciones:
 - Establecimiento de patrones de llamadas.
 - Modelos de carga en redes.
 - Detección de fraude.
- Otras áreas
 - Correo electrónico y agendas personales: clasificación y distribución automática de correo, detección de correo *spam*, gestión de avisos, análisis del empleo del tiempo.
 - Recursos Humanos: selección de empleados.
 - Web: análisis del comportamiento de los usuarios, detección de fraude en el comercio electrónico, análisis de los *logs* de un servidor web.
 - Turismo: determinar las características socioeconómicas de los turistas en un determinado destino o paquete turístico, identificar patrones de reservas, etc.
 - Tráfico: modelos de tráfico a partir de fuentes diversas: cámaras, GPS...
 - Hacienda: detección de evasión fiscal.
 - Policiales: identificación de posibles terroristas en un aeropuerto.
 - Deportes: estudio de la influencia de jugadores y de cambios. Planificación de eventos.
 - Política: diseño de campañas políticas, estudios de tendencias de grupos, etc.

Todos estos ejemplos muestran la gran variedad de aplicaciones donde el uso de la minería de datos puede ayudar a entender mejor el entorno donde se desenvuelve la organización y, en definitiva, mejorar la toma de decisiones en dicho entorno.

1.8 Sistemas y herramientas de minería de datos

La diversidad de disciplinas que contribuyen a la minería de datos está dando lugar a una gran variedad de sistemas de minería de datos. Cada uno de ellos posee unas características apropiadas para realizar determinadas tareas o para analizar cierto tipo de datos. En esta sección presentamos varias clasificaciones de los sistemas y herramientas atendiendo a varios criterios (el modelo de datos que generan, el tipo de datos que minan, el tipo de técnica o el tipo de aplicación al que se pueden aplicar):

- Tipo de base de datos minada: teniendo en cuenta los diferentes modelos de datos podemos hablar de sistemas de minería de datos relacionales, multidimensionales, orientados a objetos, etc. Asimismo, atendiendo al tipo de datos manejados, hablamos de sistemas textuales, multimedia, espaciales o web.
- Tipo de conocimiento minado: también pueden tenerse en cuenta los niveles de abstracción del conocimiento minado: conocimiento generalizado (alto nivel de abstracción), a nivel primitivo (a nivel de filas de datos), o conocimiento a múltiples niveles (de abstracción). Por último, podemos igualmente distinguir entre los sistemas que buscan regularidades en los datos (patrones) frente a los que analizan las irregularidades (excepciones).
- Tipo de funcionalidad y de técnica: los sistemas de minería de datos se pueden clasificar basándose en su funcionalidad (clasificación, agrupamiento, etc.) o por los métodos de análisis de los datos empleados (técnicas estadísticas, redes neuronales, etc.).
- Tipo de aplicación: podemos distinguir dos clases de sistemas según la aplicación para la que se usan: los sistemas de propósito general y los sistemas específicos (como los usados en aplicaciones financieras, web, e-mail, análisis de *stocks*, etc.).

A finales de la década de los 90 aparecen las *suites*, que son capaces de trabajar con distintos formatos, de incorporar distintas técnicas, de obtener distintos tipos de conocimiento y de aplicarse a un gran abanico de áreas. A esto nos referimos cuando hablamos de sistemas integrados o paquetes de minería de datos.

En el Apéndice A se ha incluido una descripción de algunos de los sistemas y herramientas que consideramos más importantes y representativos.

Capítulo 2

EL PROCESO DE EXTRACCIÓN DE CONOCIMIENTO

En el capítulo anterior hemos visto que la minería de datos no es más que un paso esencial de un proceso más amplio cuyo objetivo es el descubrimiento de conocimiento en bases de datos (del inglés *Knowledge Discovery from Databases*, KDD). Este proceso consta de una secuencia iterativa de etapas o fases. En este capítulo se presentan las fases del proceso de extracción de conocimiento: Preparación de Datos, Minería de Datos, Evaluación, Difusión y Uso de Modelos. Se dan las nociones más básicas de cada una de ellas, se presenta una tipología de tareas de minería de datos (clasificación, estimación/regresión, agrupamiento, reglas de asociación...) y de técnicas para resolverlos (funciones lineales y no lineales, árboles de decisión, redes neuronales artificiales, aprendizaje basado en instancias o casos, métodos basados en núcleos, etc.). Se introducen las medidas básicas de evaluación (precisión, soporte, confianza, error cuadrático medio, distancias...) y el concepto de evaluación mediante los conjuntos de entrenamiento y de prueba.

2.1 Las fases del proceso de extracción de conocimiento

En la Figura 2.1 se muestra que el KDD es un proceso iterativo e interactivo. Es iterativo ya que la salida de alguna de las fases puede hacer volver a pasos anteriores y porque a menudo son necesarias varias iteraciones para extraer conocimiento de alta calidad. Es interactivo porque el usuario, o más generalmente un experto en el dominio del problema, debe ayudar en la preparación de los datos, validación del conocimiento extraído, etc.

El proceso de KDD se organiza entorno a cinco fases como se ilustra en la Figura 2.1. En la **fase de integración y recopilación de datos** se determinan las fuentes de información que pueden ser útiles y dónde conseguirlas. A continuación, se transforman todos los datos a un formato común, frecuentemente mediante un almacén de datos que consiga unificar

de manera operativa toda la información recogida, detectando y resolviendo las inconsistencias. Este almacén de datos facilita enormemente la “navegación” y visualización previa de sus datos, para discernir qué aspectos puede interesar que sean estudiados. Dado que los datos provienen de diferentes fuentes, pueden contener valores erróneos o faltantes. Estas situaciones se tratan en la **fase de selección, limpieza y transformación**, en la que se eliminan o corrigen los datos incorrectos y se decide la estrategia a seguir con los datos incompletos. Además, se proyectan los datos para considerar únicamente aquellas variables o atributos que van a ser relevantes, con el objetivo de hacer más fácil la tarea propia de minería y para que los resultados de la misma sean más útiles. La selección incluye tanto una criba o fusión horizontal (filas / registros) como vertical (columnas / atributos). Las dos primeras fases se suelen englobar bajo el nombre de “preparación de datos”. En la **fase de minería de datos**, se decide cuál es la tarea a realizar (clasificar, agrupar, etc.) y se elige el método que se va a utilizar. En la **fase de evaluación e interpretación** se evalúan los patrones y se analizan por los expertos, y si es necesario se vuelve a las fases anteriores para una nueva iteración. Esto incluye resolver posibles conflictos con el conocimiento que se disponía anteriormente. Finalmente, en la **fase de difusión** se hace uso del nuevo conocimiento y se hace partícipe de él a todos los posibles usuarios. Para cada una de estas fases se emplean distintas técnicas de las diferentes disciplinas relacionadas que vimos en la Sección 1.6.

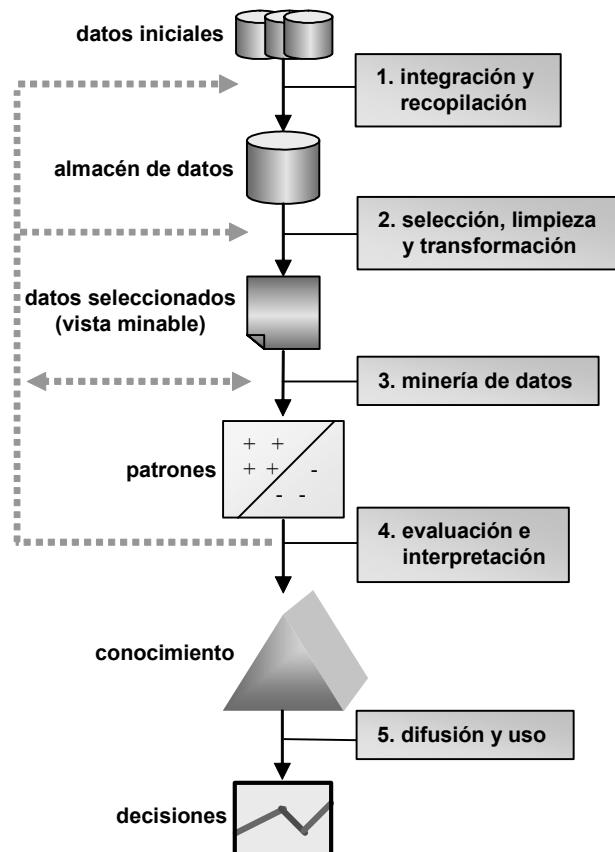


Figura 2.1. Fases del proceso de descubrimiento de conocimiento en bases de datos, KDD.

Además de las fases descritas, frecuentemente se incluye una fase previa de análisis de las necesidades de la organización y definición del problema [Two Crows Corporation 1999], en la que se establecen los objetivos de minería de datos. Por ejemplo, un objetivo de negocio de una entidad bancaria sería encontrar patrones en los datos que le ayuden a conservar los buenos clientes; para ello, podríamos tener varios objetivos de minería de datos: construir un modelo para predecir clientes rentables y un segundo modelo para identificar los clientes que probablemente dejarán de serlo.

A continuación pasamos a ver brevemente las fases de la Figura 2.1. A lo largo del libro iremos ampliando y profundizando sobre ellas.

2.2 Fase de integración y recopilación

Tal y como mencionamos en el capítulo anterior (Sección 1.1), las bases de datos y las aplicaciones basadas en el procesamiento tradicional de datos, que se conoce como **procesamiento transaccional en línea (OLTP, On-Line Transaction Processing)** son suficientes para cubrir las necesidades diarias de una organización (tales como la facturación, control de inventario, nóminas...). Sin embargo, resultan insuficientes para otras funciones más complejas como el análisis, la planificación y la predicción, es decir, para tomar decisiones estratégicas a largo plazo. En estos casos, y dependiendo de la aplicación, lo normal es que los datos necesarios para poder llevar a cabo un proceso de KDD pertenezcan a diferentes organizaciones, a distintos departamentos de una misma entidad. Incluso puede ocurrir que algunos datos necesarios para el análisis nunca hayan sido recolectados en el ámbito de la organización por no ser necesarios para sus aplicaciones. En muchos casos tendremos que adquirir además datos externos desde bases de datos públicas (como el censo, datos demográficos o climatológicos) o desde bases de datos privadas (como los datos de compañías de pagos, bancarias, eléctricas, etc., siempre que sea a un nivel agregado para no infringir la legalidad). Esto representa un reto, ya que cada fuente de datos usa diferentes formatos de registro, diferentes grados de agregación de los datos, diferentes claves primarias, diferentes tipos de error, etc. Lo primero, por lo tanto, es integrar todos estos datos. La idea de la integración de múltiples bases de datos ha dado lugar a la tecnología de **almacenes de datos (data warehousing)**. Este término, tan popular actualmente, hace referencia a la tendencia actual en las empresas e instituciones de colecciónar datos de las bases de datos transaccionales y otras fuentes diversas para hacerlos accesibles para el análisis y la toma de decisiones.

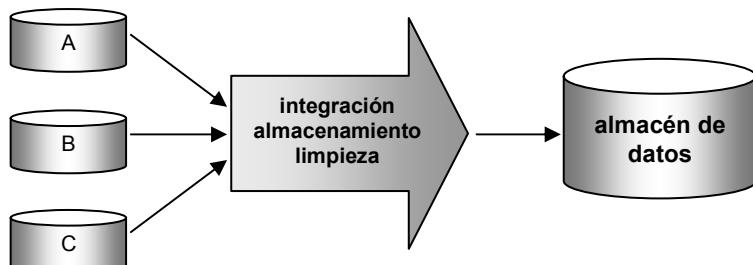


Figura 2.2. Integración en un almacén de datos.

Un almacén de datos es un repositorio de información coleccionada desde varias fuentes, almacenada bajo un esquema unificado que normalmente reside en un único emplazamiento.

to. Existen varias formas de mezclar las distintas bases de datos para crear el repositorio. Una posibilidad es simplemente hacer una copia de las bases de datos integrantes (probablemente eliminando inconsistencias y redundancias). Obviamente, esta aproximación limita las ventajas para acceder a bases de datos heterogéneas. Por ello, generalmente los almacenes de datos se construyen vía un proceso de integración y almacenamiento en un nuevo esquema integrado. En la Figura 2.2 se muestra este proceso de integración de un almacén de datos para tres fuentes de datos originales (A, B y C).

Esencialmente, los almacenes de datos se utilizan para poder agregar y cruzar eficientemente la información de maneras sofisticadas. Por ello, los datos se modelan con una estructura de base de datos multidimensional, donde cada dimensión corresponde a un atributo o conjunto de atributos en el esquema en torno a unos “hechos” que almacenan el valor de alguna medida agregada, como por ejemplo la cantidad vendida de un producto en un día concreto en una tienda. Esta visión multidimensional hace a los almacenes de datos adecuados para el **procesamiento analítico en línea** (*on-line analytical processing, OLAP*). Las operaciones OLAP permiten un análisis multidimensional de los datos, que es superior al SQL para computar resúmenes y desgloses en muchas dimensiones, pudiendo utilizar conocimiento previo sobre el dominio de los datos para permitir su presentación a diferentes niveles de abstracción, acomodando así diferentes puntos de vista del usuario.

Una cuestión importante para los profesionales del procesamiento de datos es la diferencia entre minería de datos y OLAP. El usuario de una herramienta OLAP utiliza la herramienta para obtener información agregada a partir de información detallada, combinando la información de manera flexible. Esto permite obtener informes y vistas sofisticadas en tiempo real. Además, las herramientas OLAP pueden utilizarse para comprobar rápidamente patrones y pautas hipotéticas sugeridas por el usuario con el objetivo de verificarlas o rechazarlas. Se trata, por lo tanto, de un proceso esencialmente deductivo. Por el contrario, la minería de datos, más que verificar patrones hipotéticos, usa los datos para encontrar estos patrones. Por lo tanto, es un proceso inductivo. Ambos tipos de herramientas se complementan: podemos usar OLAP al principio del proceso de KDD para explorar los datos (por ejemplo, para centrar nuestra atención en las variables importantes, identificar excepciones o encontrar interacciones), ya que cuanto más comprendamos los datos más efectivo será el proceso de descubrir conocimiento.

Como hemos dicho, un almacén de datos es muy aconsejable para la minería de datos, aunque no imprescindible. En algunos casos, en especial cuando el volumen no es muy grande, se puede trabajar con los datos originales o en formatos heterogéneos (archivos de texto, hojas de cálculo...).

2.3 Fase de selección, limpieza y transformación

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

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

La presencia de datos faltantes o perdidos (*missing values*) puede ser también un problema pernicioso que puede conducir a resultados poco precisos. No obstante, es necesario reflexionar primero sobre el significado de los valores faltantes antes de tomar ninguna decisión sobre cómo tratarlos ya que éstos pueden deberse a causas muy diversas, como a un mal funcionamiento del dispositivo que hizo la lectura del valor, a cambios efectuados en los procedimientos usados durante la colección de los datos o al hecho de que los datos se recopilen desde fuentes diversas. Por ello, existen muchas aproximaciones para manejar los datos faltantes, como veremos en el Capítulo 4.

Estos dos problemas son sólo dos ejemplos que muestran la necesidad de la limpieza de datos, es decir, de mejorar su calidad. Como hemos dicho, no es sólo suficiente con tener una buena calidad de datos, sino además poder proporcionar a los métodos de minería de datos el subconjunto de datos más adecuado para resolver el problema. Para ello es necesario seleccionar los datos apropiados.

La selección de atributos relevantes es uno de los preprocesamientos más importantes, ya que es crucial que los atributos utilizados sean relevantes para la tarea de minería de datos. Por ejemplo, supongamos que los jueces del torneo de Wimbledon desean determinar a partir de las condiciones climatológicas (nubosidad, humedad, temperatura, etc.) si se puede jugar o no al tenis. Para ello se cuenta con los datos recogidos de experiencias anteriores. Probablemente, la base de datos contenga un atributo que identifica cada uno de los días considerados (por ejemplo, la fecha). Si consideramos este atributo en el proceso de minería, un algoritmo de generación de reglas podría obtener reglas como

SI (fecha=10/06/2003) **ENTONCES** (jugar_tenis=sí)

que, aunque correcta, es inútil para realizar predicciones futuras.

Idealmente, uno podría usar todas las variables y dejar que la herramienta de minería de datos fuera probando hasta elegir las mejores variables predictoras. Obviamente, esta forma de trabajar no funciona bien, entre otras cosas porque el tiempo requerido para construir un modelo crece con el número de variables. Aunque en principio algunos algoritmos de minería de datos automáticamente ignoran las variables irrelevantes, en la práctica nuestro conocimiento sobre el dominio del problema puede permitirnos hacer correctamente muchas de esas selecciones.

Como en el caso de las variables, también podríamos construir el modelo usando todos los datos. Pero si tenemos muchos, tardaríamos mucho tiempo y probablemente también necesitaríamos una máquina más potente. Consecuentemente, una buena idea es usar una muestra (*sample*) a partir de algunos datos (o filas). La selección de la muestra debe ser hecha cuidadosamente para asegurar que es verdaderamente aleatoria.

Otra tarea de preparación de los datos es la construcción de atributos, la cual consiste en construir automáticamente nuevos atributos aplicando alguna operación o función a los atributos originales con objeto de que estos nuevos atributos hagan más fácil el proceso de minería. La motivación principal para esta tarea es fuerte cuando los atributos originales no tienen mucho poder predictivo por sí solos o los patrones dependen de variaciones lineales de las variables originales. Por ejemplo, el precio de las viviendas de una zona se puede estimar mucho mejor a partir de la densidad de población de la zona que de la población absoluta y de su superficie. Por tanto, es razonable derivar el atributo densidad de población de los otros dos.

El tipo de los datos puede también modificarse para facilitar el uso de técnicas que requieren tipos de datos específicos. Así, algunos atributos se pueden numerizar, lo que reduce el espacio y permite usar técnicas numéricas. Por ejemplo, podemos reemplazar los valores del atributo “tipo de vivienda” por enteros.

El proceso inverso consiste en discretizar los atributos continuos, es decir, transformar valores numéricos en atributos discretos o nominales. Los atributos discretizados pueden tratarse como atributos categóricos con un número más pequeño de valores. La idea básica es partir los valores de un atributo continuo en una pequeña lista de intervalos, tal que cada intervalo es visto como un valor discreto del atributo.

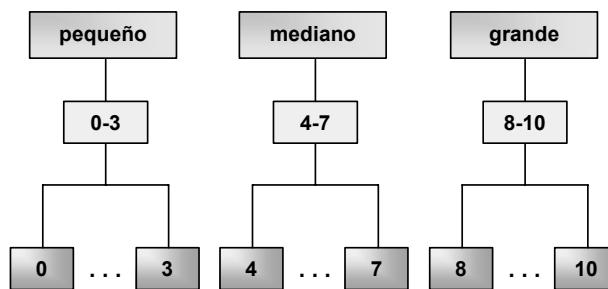


Figura 2.3. Ejemplo de discretización del atributo tamaño.

La Figura 2.3 ilustra una posible discretización para el atributo *tamaño*, con valores de 0 a 10. La parte inferior de la figura muestra la lista ordenada de los valores continuos, los cuales se han discretizado en tres intervalos a los que se les ha asignado los valores discretos *pequeño*, *mediano* y *grande*, como puede verse en la parte superior de la figura.

2.4 Fase de minería de datos

La fase de minería de datos es la más característica del KDD y, por esta razón, muchas veces se utiliza esta fase para nombrar todo el proceso. El objetivo de esta fase es producir nuevo conocimiento que pueda utilizar el usuario. Esto se realiza construyendo un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para

entender mejor los datos o para explicar situaciones pasadas. Para ello es necesario tomar una serie de decisiones antes de empezar el proceso:

- Determinar qué tipo de tarea de minería es el más apropiado. Por ejemplo, podríamos usar la clasificación para predecir en una entidad bancaria los clientes que dejarán de serlo.
- Elegir el tipo de modelo. Por ejemplo, para una tarea de clasificación podríamos usar un árbol de decisión, porque queremos obtener un modelo en forma de reglas.
- Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando. Esta elección es pertinente porque existen muchos métodos para construir los modelos. Por ejemplo, para crear árboles de decisión para clasificación podríamos usar CART o C5.0, entre otros. En los capítulos siguientes se presentarán los métodos más importantes para cada tipo de modelo.

En lo que resta de esta sección, describimos las tareas y modelos más utilizados, así como algunos conceptos relacionados con la construcción del modelo. Una descripción más completa se dará en el Capítulo 6.

2.4.1 Tareas de la minería de datos

Dentro de la minería de datos hemos de distinguir tipos de tareas, cada una de las cuales puede considerarse como un tipo de problema a ser resuelto por un algoritmo de minería de datos. Esto significa que cada tarea tiene sus propios requisitos, y que el tipo de información obtenida con una tarea puede diferir mucho de la obtenida con otra.

Tal y como comentamos en el capítulo anterior, las distintas tareas pueden ser predictivas o descriptivas. Entre las tareas predictivas encontramos la clasificación y la regresión, mientras que el agrupamiento (*clustering*), las reglas de asociación, las reglas de asociación secuenciales y las correlaciones son tareas descriptivas. Veamos en mayor detalle todas ellas.

La **clasificación** es quizá la tarea más utilizada. En ella, cada instancia (o registro de la base de datos) pertenece a una clase, la cual se indica mediante el valor de un atributo que llamamos la clase de la instancia. Este atributo puede tomar diferentes valores discretos, cada uno de los cuales corresponde a una clase. El resto de los atributos de la instancia (los relevantes a la clase) se utilizan para predecir la clase. El objetivo es predecir la clase de nuevas instancias de las que se desconoce la clase. Más concretamente, el objetivo del algoritmo es maximizar la razón de precisión de la clasificación de las nuevas instancias, la cual se calcula como el cociente entre las predicciones correctas y el número total de predicciones (correctas e incorrectas).

Ejemplo. Consideremos un oftalmólogo que desea disponer de un sistema que le sirva para determinar la conveniencia o no de recomendar la cirugía ocular a sus pacientes. Para ello dispone de una base de datos de sus antiguos pacientes clasificados en operados satisfactoriamente o no en función del tipo de problema que padecían (miopía y su grado, o astigmatismo) y de su edad. El modelo encontrado se utiliza para clasificar nuevos pacientes, es decir, para decidir si es conveniente operarlos o no.

Existen variantes de la tarea de la clasificación, como son el aprendizaje de “rankings”, el aprendizaje de preferencias, el aprendizaje de estimadores de probabilidad, etc.

La **regresión** es también una tarea predictiva que consiste en aprender una función real que asigna a cada instancia un valor real. Ésta es la principal diferencia respecto a la clasificación; el valor a predecir es numérico. El objetivo en este caso es minimizar el error (generalmente el error cuadrático medio) entre el valor predicho y el valor real.

Ejemplo. *Un empresario quiere conocer cuál es el costo de un nuevo contrato basándose en los datos correspondientes a contratos anteriores. Para ello usa una fórmula de regresión lineal, ajustando con los datos pasados la función lineal y usándola para predecir el costo en el futuro.*

El **agrupamiento** (*clustering*) es la tarea descriptiva por excelencia y consiste en obtener grupos “naturales” a partir de los datos. Hablamos de grupos y no de clases, porque, a diferencia de la clasificación, en lugar de analizar datos etiquetados con una clase, los analiza para generar esta etiqueta. Los datos son agrupados basándose en el principio de maximizar la similitud entre los elementos de un grupo minimizando la similitud entre los distintos grupos. Es decir, se forman grupos tales que los objetos de un mismo grupo son muy similares entre sí y, al mismo tiempo, son muy diferentes a los objetos de otro grupo. Al agrupamiento también se le suele llamar segmentación, ya que parte o segmenta los datos en grupos que pueden ser o no disjuntos. El agrupamiento está muy relacionado con la summarización, que algunos autores consideran una tarea en sí misma, en la que cada grupo formado se considera como un resumen de los elementos que lo forman para así describir de una manera concisa los datos.

Ejemplo. *Una librería que ofrece sus servicios a través de la red usa el agrupamiento para identificar grupos de clientes en base a sus preferencias de compras que le permita dar un servicio más personalizado. Así, cada vez que un cliente se interesa por un libro, el sistema identifica a qué grupo pertenece y le recomienda otros libros comprados por clientes de su mismo grupo.*

Las **correlaciones** son una tarea descriptiva que se usa para examinar el grado de similitud de los valores de dos variables numéricas. Una fórmula estándar para medir la correlación lineal es el coeficiente de correlación r , el cual es un valor real comprendido entre -1 y 1 . Si r es 1 (respectivamente, -1) las variables están perfectamente correlacionadas (perfectamente correlacionadas negativamente), mientras que si es 0 no hay correlación. Esto quiere decir que cuando r es positivo, las variables tienen un comportamiento similar (ambas crecen o decrecen al mismo tiempo) y cuando r es negativo si una variable crece la otra decrece. El análisis de correlaciones, sobre todo las negativas, puede ser muy útil para establecer reglas de ítems correlacionados, como se muestra en el siguiente ejemplo.

Ejemplo. *Un inspector de incendios que desea obtener información útil para la prevención de incendios probablemente esté interesado en conocer correlaciones negativas entre el empleo de distintos grosores de protección del material eléctrico y la frecuencia de ocurrencia de incendios.*

Las **reglas de asociación** son también una tarea descriptiva, muy similar a las correlaciones, que tiene como objetivo identificar relaciones no explícitas entre atributos *categóricos*. Pueden ser de muchas formas, aunque la formulación más común es del estilo “si el atributo X toma el valor a entonces el atributo Y toma el valor b ”. Las reglas de asociación no implican una relación causa-efecto, es decir, puede no existir una causa para que los datos estén asociados. Este tipo de tarea se utiliza frecuentemente en el análisis de la cesta

de la compra, para identificar productos que son frecuentemente comprados juntos, información esta que puede usarse para ajustar los inventarios, para la organización física del almacén o en campañas publicitarias. Las reglas se evalúan usando dos parámetros: precisión y soporte (cobertura)

Ejemplo. Una compañía de asistencia sanitaria desea analizar las peticiones de servicios médicos solicitados por sus asegurados. Cada petición contiene información sobre las pruebas médicas que fueron realizadas al paciente durante una visita. Toda esta información se almacena en una base de datos en la que cada petición es un registro cuyos atributos expresan si se realiza o no cada una de las posibles pruebas médicas que pueden ser realizadas a un paciente. Mediante reglas de asociación, un sistema encontraría aquellas pruebas médicas que frecuentemente se realizan juntas, por ejemplo que un 70 por ciento de las veces que se pide un análisis de orina también se solicita uno de sangre, y esto ocurre en dos de cada diez pacientes. La precisión de esta regla es del 70 por ciento y el soporte del 20 por ciento.

Un caso especial de reglas de asociación, que recibe el nombre de **reglas de asociación secuenciales**, se usa para determinar patrones secuenciales en los datos. Estos patrones se basan en secuencias temporales de acciones y difieren de las reglas de asociación en que las relaciones entre los datos se basan en el tiempo.

Ejemplo. Una tienda de venta de electrodomésticos y equipos de audio analiza las ventas que ha efectuado usando análisis secuencial y descubre que el 30 por ciento de los clientes que compraron un televisor hace seis meses compraron un DVD en los siguientes dos meses.

2.4.2 Técnicas de minería de datos

Dado que la minería de datos es un campo muy interdisciplinario, como vimos en la Sección 1.6, existen diferentes paradigmas detrás de las técnicas utilizadas para esta fase: técnicas de inferencia estadística, árboles de decisión, redes neuronales, inducción de reglas, aprendizaje basado en instancias, algoritmos genéticos, aprendizaje bayesiano, programación lógica inductiva y varios tipos de métodos basados en núcleos, entre otros. Cada uno de estos paradigmas incluye diferentes algoritmos y variaciones de los mismos, así como otro tipo de restricciones que hacen que la efectividad del algoritmo dependa del dominio de aplicación, no existiendo lo que podríamos llamar el método universal aplicable a todo tipo de aplicación.

A continuación revisamos los aspectos principales de algunas de las técnicas mencionadas. Para un estudio más detallado y profundo de paradigmas y algoritmos, remitimos al lector a la Parte III de este libro.

Existen muchos conceptos **estadísticos** que son la base de muchas técnicas de minería de datos. Ya hemos mencionado la regresión lineal, como un método simple pero frecuentemente utilizado para la tarea de regresión, como se muestra en la Figura 2.4.

En general, la fórmula para una regresión lineal es $y=c_0 + c_1x_1 + \dots + c_nx_n$, donde x_i son los atributos predictores e y la salida (la variable dependiente). Si los atributos son modificados en la función de regresión por alguna otra función (cuadrados, inversa, logaritmos, combinaciones de variables...), es decir $y=c_0 + f_1(x_1) + \dots + f_n(x_n)$, la regresión se dice no lineal. Se pueden incorporar variantes locales o transformaciones en las variables

predictoras y en la salida, permitiendo flexibilizar este tipo de técnicas. El abanico de técnicas se dispara aún más cuando consideramos técnicas no paramétricas.

Las técnicas estadísticas no son sólo útiles para regresión, sino que se utilizan también para discriminación (clasificación o agrupamiento). La inferencia de *funciones discriminantes* que separen clases o grupos, también se puede realizar de manera paramétrica o no paramétrica. El método más conocido es el análisis de discriminantes lineales de Fisher, que se verá, junto con algunas otras técnicas estadísticas, en los capítulos 7 y 8.

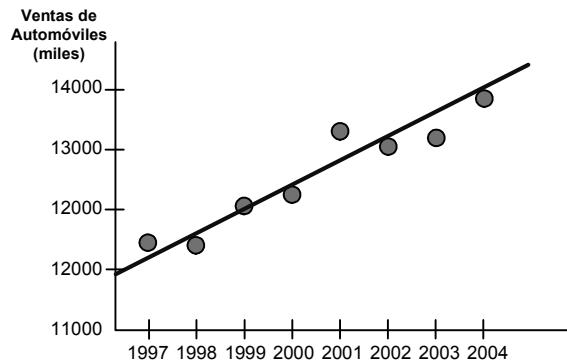


Figura 2.4. Ejemplo de regresión lineal.

Algunas de las técnicas de discriminantes no paramétricos tienen una relación muy estrecha con los **métodos basados en núcleo**, de los cuales las máquinas de vectores soporte son su ejemplo más representativo, en el que se busca un discriminante lineal que maximice la distancia a los ejemplos fronterizos de los distintos grupos o clases. Por ejemplo, la Figura 2.5 muestra el uso de un clasificador para distinguir en una cooperativa agrícola las naranjas que son aptas para consumo directo o para zumo (parte inferior derecha) y cuáles se separan para conservas y procesos industriales (parte superior izquierda), según su diámetro y peso. Para ello se han etiquetado previamente en las dos clases existentes un conjunto de muestras para los cuales se tenía claro su uso en la cooperativa.

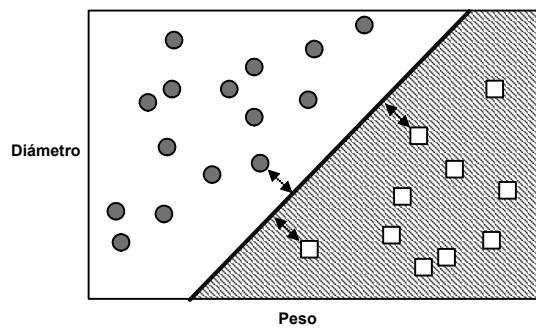


Figura 2.5. Ejemplo de un discriminante (clasificador) basado en vectores soporte.

En este caso, el discriminante lineal es muy fácil de encontrar. Lo interesante de estas técnicas es que, en problemas no lineales, este discriminante lineal también se puede encontrar porque se utilizan núcleos para convertir el problema en un problema de mayor dimensionalidad y porque se relajan ciertas condiciones, como veremos en el Capítulo 14.

En otras ocasiones, deseamos calcular, para una instancia dada sin clasificar, cuál es la probabilidad de que se le asigne cada una de las clases, y seleccionar la de mayor probabilidad. Ésta es la idea que subyace en los **métodos bayesianos**. Uno de los métodos más utilizados es el *Naive Bayes*, que se basa en la regla de Bayes y que “ingenuamente” asume la independencia de los atributos dada la clase. Este método funciona bien con bases de datos reales, sobre todo cuando se combina con otros procedimientos de selección de atributos que sirven para eliminar la redundancia.

La regla de Bayes establece que, si tenemos una hipótesis H sustentada para una evidencia E , entonces:

$$p(H | E) = \frac{p(E | H) \cdot p(H)}{p(E)}$$

donde $p(A)$ representa la probabilidad del suceso A , usando la notación $p(A|B)$ para denotar la probabilidad del suceso A condicionada al suceso B .

Veamos cómo funciona con un ejemplo. Una compañía de seguros dispone de los siguientes datos sobre sus clientes, clasificados en buenos y malos clientes (Tabla 2.1):

#Instancia	edad	hijos	practica_deporte	salario	buen_cliente
1	joven	sí	no	alto	sí
2	joven	no	no	medio	no
3	joven	sí	sí	medio	no
4	joven	sí	no	bajo	sí
5	mayor	sí	no	bajo	sí
6	mayor	no	sí	medio	sí
7	joven	no	sí	medio	sí
8	joven	sí	sí	alto	sí
9	mayor	sí	no	medio	sí
10	mayor	no	no	bajo	no

Tabla 2.1. Datos de una compañía de seguros.

Ahora supongamos que se tiene un nuevo ejemplo con los siguientes valores:

edad	hijos	practica_deporte	salario	buen_cliente
mayor	no	no	medio	?

La hipótesis H es que *buen-cliente* sea *sí* (o, alternativamente, *no*). La evidencia E es una combinación de los valores de los atributos *edad*, *hijos*, *practica_deporte* y *salario* del dato nuevo, por lo que su probabilidad se obtiene multiplicando las probabilidades de estos valores. Es decir,

$$p(\text{sí} | E) = \frac{[p(\text{edad}_E | \text{sí}) \cdot p(\text{hijos}_E | \text{sí}) \cdot p(\text{practica_deporte}_E | \text{sí}) \cdot p(\text{salario}_E | \text{sí})] \cdot p(\text{sí})}{p(E)}$$

El término $p(\text{edad}_E | \text{sí})$ se calcula dividiendo el número de instancias de la Tabla 2.1 que tienen el valor *mayor* en el atributo *edad* (de los que el *buen-cliente* es *sí*) dividido por el número de instancias cuyo valor del atributo *buen-cliente* es *sí*, es decir, $p(\text{edad}_E | \text{sí}) = p(\text{mayor}_E | \text{sí}) = 3/7$. De igual forma obtenemos el resto de probabilidades condicionadas en el numerador de la ecuación anterior. El término $p(\text{sí})$ se calcula como el número de instancias de la Tabla 2.1 cuyo valor del atributo *buen-cliente* es *sí* dividido por el número total de

instancias, es decir $p(\text{sí}) = 7/10$. Por último, el denominador $p(E)$ desaparece normalizando. Sustituyendo todos estos valores se obtiene que la probabilidad de que se asigne el valor *sí* al atributo *buen-cliente* del dato *E* es

$$p(\text{sí} | E) = \frac{3}{7} \cdot \frac{2}{7} \cdot \frac{4}{7} \cdot \frac{3}{7} \cdot \frac{7}{10} = 0,0210$$

Procediendo de igual forma para la clase *no* resulta $p(\text{no}|E) = 0,0296$, por lo que se asignará el valor *no* al atributo *buen-cliente* del dato *E*. Los métodos bayesianos se verán en el Capítulo 10.

Los árboles de decisión son una serie de decisiones o condiciones organizadas en forma jerárquica, a modo de árbol. Son muy útiles para encontrar estructuras en espacios de alta dimensionalidad y en problemas que mezclen datos categóricos y numéricos. Esta técnica se usa en tareas de clasificación, agrupamiento y regresión. Los árboles de decisión usados para predecir variables categóricas reciben el nombre de árboles de clasificación, ya que distribuyen las instancias en clases. Cuando los árboles de decisión se usan para predecir variables continuas se llaman árboles de regresión. Vamos a mostrar un ejemplo de árbol de clasificación. Para ello usaremos como ejemplo un típico problema muy utilizado en el aprendizaje automático. Se trata de un conjunto de datos ficticio que muestra las condiciones climatológicas (pronóstico, humedad y viento) adecuadas para jugar un cierto deporte (por ejemplo, tenis en Wimbledon). Los datos de los que disponemos son los siguientes:

#instancia	pronóstico	humedad	viento	jugar
1	soleado	alta	débil	No
2	cubierto	alta	débil	Sí
3	lluvioso	alta	débil	Sí
4	lluvioso	normal	fuerte	No
5	soleado	normal	débil	Sí
...

Usando un algoritmo de aprendizaje de árboles de decisión podríamos obtener el árbol que se muestra en la Figura 2.6:

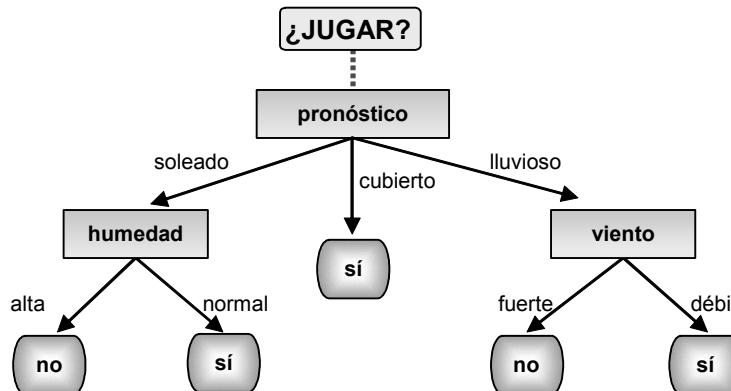


Figura 2.6. Árbol de decisión para determinar si se juega o no a un cierto deporte.

Los árboles de decisión siguen una aproximación “divide y vencerás” para partir el espacio del problema en subconjuntos. Encima del nodo raíz del árbol tenemos el problema a resolver. En nuestro ejemplo, se trata de decidir si *jugar* o no. Los nodos internos (nodos de

decisión) corresponden a particiones sobre atributos particulares, como por ejemplo *pronóstico*, y los arcos queemanan de un nodo corresponden a los posibles valores del atributo considerado en ese nodo (por ejemplo, *soleado*, *cubierto* o *lluvioso*). Cada arco conduce a otro nodo de decisión o a un nodo hoja. Los nodos hoja representan la predicción (o clase) del problema para todas aquellas instancias que alcanzan esa hoja. Para clasificar una instancia desconocida, se recorre el árbol de arriba hacia abajo de acuerdo a los valores de los atributos probados en cada nodo y, cuando se llega a una hoja, la instancia se clasifica con la clase indicada por esa hoja.

Existen muchos métodos de árboles de decisión que difieren entre sí en la forma de crear el árbol. En el Capítulo 11 se presenta esta técnica en detalle.

Los árboles de decisión pueden considerarse una forma de aprendizaje de reglas, ya que cada rama del árbol puede interpretarse como una regla, donde los nodos internos en el camino desde la raíz a las hojas definen los términos de la conjunción que constituye el antecedente de la regla, y la clase asignada en la hoja es el consecuente. La Figura 2.7 muestra el conjunto de reglas que corresponde al árbol de la Figura 2.6, en la que se han agrupado en una regla por defecto (“EN OTRO CASO”) todas las ramas del árbol cuya hoja asigna la clase *no*.

SI pronóstico=soleado **Y** humedad=normal **ENTONCES** jugar=sí

SI pronóstico=cubierto **ENTONCES** jugar=sí

SI pronóstico=lluvioso **Y** viento=débil **ENTONCES** jugar=sí

EN OTRO CASO jugar=no.

Figura 2.7. Reglas para el árbol de decisión de la Figura 2.6.

En general, la **inducción de reglas** es un conjunto de métodos para derivar un conjunto de reglas comprensibles de la forma:

SI cond₁ **Y** cond₂ **Y** ... **Y** cond_n **ENTONCES** pred.

El antecedente de la regla (la parte **SI**) contiene una conjunción de *n* condiciones sobre los valores de los atributos independientes, mientras que el consecuente de la regla (la parte **ENTONCES**) contiene una predicción sobre el valor de un atributo objetivo. La semántica de este tipo de reglas de predicción es: si se satisfacen todas las condiciones especificadas en el antecedente de la regla para los atributos independientes de un registro de datos (una instancia) entonces se predice que el atributo objetivo de este registro tendrá el valor especificado en el consecuente de la regla.

Aunque los árboles de decisión pueden también producir un conjunto de reglas (tal y como hemos visto anteriormente), los métodos de inducción de reglas son diferentes ya que:

- las reglas son independientes y no tienen por qué formar un árbol.
- las reglas generadas pueden no cubrir todas las situaciones posibles.
- las reglas pueden entrar en conflicto en sus predicciones; en este caso, es necesario elegir qué regla se debe seguir. Un método para resolver los conflictos consiste en asignar un valor de confianza a las reglas y usar la que tenga mayor confianza.

Algunos métodos de obtención de reglas, en especial para la tarea de reglas de asociación, se basan en el concepto de conjuntos de ítems frecuentes (*frequent itemsets*) y utilizan **técnicas de conteo y soporte mínimo** para obtener las reglas. Estos métodos se verán en el Capítulo 9.

Las condiciones en el antecedente de las reglas pueden ser comparaciones entre un atributo y uno de los valores de su dominio, o bien entre un par de atributos (siempre que sean del mismo dominio o de dominios compatibles). Usando la terminología de la lógica, las expresiones del primer tipo forman condiciones proposicionales, mientras que las del segundo tipo forman condiciones de primer orden. Por ejemplo, las expresiones *suelto=60.000* o *gasto > ingreso* son posibles condiciones proposicional y de primer orden, respectivamente. En general, las condiciones en lógica de primer orden pueden contener predicados más complejos que las simples comparaciones. El propósito de la programación lógica inductiva (del inglés, *Inductive Logic Programming*, ILP) es permitir una representación más rica que los métodos proposicionales. Los métodos ILP incorporan de forma natural conocimiento de base y pueden usarse para descubrir patrones que afecten a varias relaciones. Estos métodos se verán en el Capítulo 12.

Las **redes neuronales artificiales** son todo un paradigma de computación muy potente que permite modelizar problemas complejos en los que puede haber interacciones no lineales entre variables. Como los árboles de decisión, las redes neuronales pueden usarse en problemas de clasificación, de regresión y de agrupamiento. Las redes neuronales trabajan directamente con datos numéricos. Para usarlas con datos nominales éstos deben numerizarse primero.

Una red neuronal puede verse como un grafo dirigido con muchos nodos (elementos del proceso) y arcos entre ellos (sus interconexiones). Cada uno de estos elementos funciona independientemente de los demás, usando datos locales (la entrada y la salida del nodo) para dirigir su procesamiento.

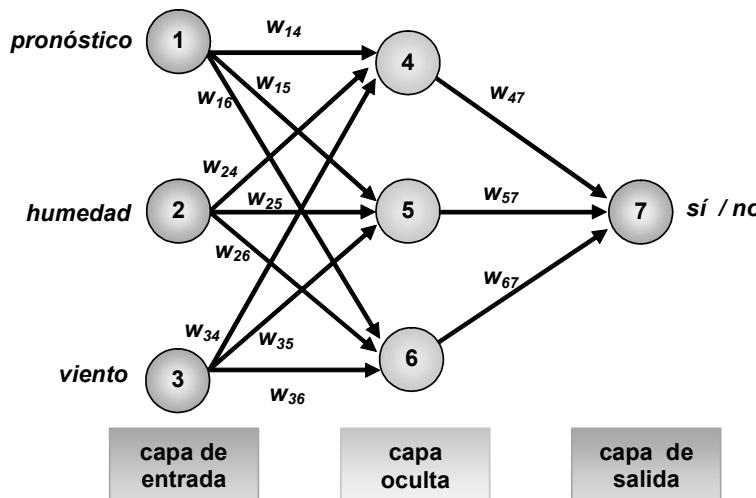


Figura 2.8. Red neuronal para el problema de jugar un cierto deporte.

La organización más popular de una red neuronal consta de una capa de entrada, en la que cada nodo corresponde a una variable independiente a examinar, unos nodos internos organizados en una o varias capas ocultas y una capa de salida con los nodos de salida (los

posibles valores de las variables objetivo). Cada nodo de la capa de entrada está conectado a cada nodo de la capa oculta. Los nodos de la capa oculta pueden estar conectados a nodos de otra capa oculta o a los nodos de la capa de salida. Cada arco está etiquetado por un peso de conexión (w) y en cada nodo hay una función de activación que indica el efecto de ese nodo sobre los datos que entran en él. Para usar una red neuronal ya entrenada se introducen los valores de los atributos de una instancia en los nodos de entrada y los nodos de salida determinan la predicción para dicha instancia.

Veamos de forma simplificada cómo funciona una red neuronal sobre el problema de determinar si se juega o no (Figura 2.8). Lo primero es determinar la estructura del grafo. Dado que hay tres atributos independientes (*pronóstico, humedad* y *viento*) vamos a asumir que en la capa de entrada hay tres nodos (en la figura etiquetados como 1, 2 y 3). Como el objetivo es efectuar una clasificación de un solo valor, la clase “sí” (la clase “no” se entiende como la negación de la clase “sí”, por lo que no se incluye en la capa de salida), la capa de salida constará de un nodo (en la figura, nodo 7). Finalmente, asumiremos que existe una única capa oculta (el número de capas ocultas de una red neuronal no es fácil de determinar) y arbitrariamente consideramos que contiene tres nodos (en la figura, nodos 4, 5 y 6). Todos los arcos están etiquetados con pesos w_{ij} , que indican el peso entre los nodos i y j , y cada nodo i tiene asociada una función de activación f_i . Durante el proceso, las funciones y los pesos actúan sobre las entradas de los nodos. Así, por ejemplo, dada una tupla de entrada (*pronóstico, humedad, viento*) con los valores de los tres atributos de entrada, la salida del nodo 1 sería $f_1(\text{pronóstico})$, la del nodo 2 sería $f_2(\text{humedad})$ y la del 3 sería $f_3(\text{viento})$. Similarmente, la salida del nodo 4 sería $f_4(w_{14}f_1(\text{pronóstico})+w_{24}f_2(\text{humedad})+w_{34}f_3(\text{viento}))$. Y así sucesivamente.

Los pesos de conexión son parámetros desconocidos que deben estimarse por un método de entrenamiento. El método más comúnmente utilizado es el de propagación hacia atrás (*backpropagation*). La idea básica es reducir el valor de error de la salida de la red.

Las redes neuronales tienen una gran capacidad de generalización para problemas no lineales, aunque requieren bastantes datos para su entrenamiento. Su mayor desventaja es, no obstante, que el modelo aprendido es difícilmente comprensible. Para una descripción detallada sobre las redes neuronales remitimos al lector al Capítulo 13 de este libro.

En el **aprendizaje basado en instancias o casos**, las instancias se almacenan en memoria, de tal forma que cuando llega una nueva instancia cuyo valor es desconocido se intenta relacionar ésta con las instancias almacenadas (cuya clase o valor es conocida) buscando las que más se parecen, con el objetivo de usar los valores de estas instancias similares para estimar los valores a obtener de la nueva instancia en cuestión. Por lo tanto, más que intentar crear reglas, se trabaja directamente con los ejemplos.

Todo el trabajo en el aprendizaje basado en instancias se hace cuando llega una instancia a clasificar y no cuando se procesa el conjunto de entrenamiento. En este sentido se trata de un método retardado o perezoso, ya que retraza el trabajo real tanto como sea posible, a diferencia de los otros métodos vistos hasta el momento que tienen un comportamiento anticipativo o voraz, produciendo generalizaciones en cuanto reciben los datos de entrenamiento.

En el aprendizaje basado en instancias, cada nueva instancia se compara con las existentes usando una métrica de distancia, y la instancia más próxima se usa para asignar

su clase a la instancia nueva. La variante más sencilla de este método de clasificación es conocido como “el vecino más próximo” (*nearest-neighbor*). Otra variante, conocida como el método de los “ k vecinos más próximos” (*k -nearest-neighbors*), usa los k vecinos más próximos, en cuyo caso la clase mayoritaria de estos k vecinos se asigna a la nueva instancia. En la Figura 2.9 se observa que a la nueva instancia N se le ha de asignar la clase a ya que, entre los vecinos más próximos (marcados con un círculo), hay más instancias de la clase a que de la b .

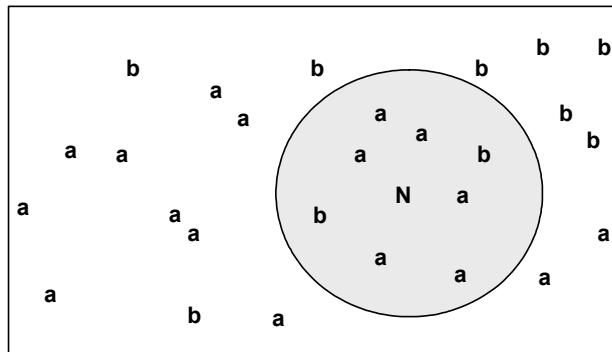


Figura 2.9. K -vecinos más próximos.

El aprendizaje basado en instancias es muy útil para trabajar sobre tipos de datos no estándar, como los textos o multimedia. El único requerimiento para incluir un tipo de datos es la existencia de una métrica apropiada de distancia para formalizar el concepto de similitud.

Calcular la distancia entre dos ejemplos o instancias es trivial cuando tienen un atributo numérico: basta con calcular la diferencia entre sus valores. Cuando hay varios atributos numéricos se puede usar la distancia euclídea, asumiendo que los atributos están normalizados y que son de igual importancia. Estas suposiciones no siempre se corresponden con la realidad y existen variantes que ponderan más los atributos importantes, así como otras métricas de distancia que veremos en el Capítulo 16.

Los atributos nominales deben tratarse de forma especial definiendo “distancias” entre los diferentes valores. Normalmente, se asigna una distancia 0 si los valores son idénticos, y una distancia de 1 en cualquier otro caso. Así, la distancia entre *joven* y *joven* es 0, mientras que la distancia entre *joven* y *adulto* es de 1. En algunos casos es conveniente usar medidas más sofisticadas. Por ejemplo, podríamos usar una medida que estableciera que *joven* es más próximo a *adulto* que a *mayor*. Finalmente, algunos atributos son más importantes que otros, lo cual puede reflejarse en la métrica a través de pesos.

Los **algoritmos evolutivos** son métodos de búsqueda colectiva en el espacio de soluciones. Dada una población de potenciales soluciones a un problema, la computación evolutiva expande esta población con nuevas y mejores soluciones. El nombre se debe a que siguen los patrones de la evolución biológica. Los cromosomas proporcionan la representación o codificación de un individuo. Parte de los cromosomas, los genes, se usan para definir diferentes rasgos del individuo. Durante la reproducción (cruce) los genes de los padres se combinan para producir los genes de los hijos. La población va mejorando de generación en generación, porque los individuos que representan las soluciones más adecuadas al problema tienen más posibilidades de sobrevivir.

En la minería de datos, los algoritmos genéticos se pueden usar para el agrupamiento, la clasificación y las reglas de asociación, así como para la selección de atributos. En cualquiera de estos casos, se comienza con un modelo o solución inicial y, a través de múltiples iteraciones, los modelos se combinan para crear nuevos modelos. Para ello, se usa una función de adaptación o de optimalidad (*fitness function*), que selecciona los mejores modelos que sobrevivirán o serán cruzados. Los distintos algoritmos genéticos difieren en la forma en que se representan los modelos, cómo se combinan los individuos en el modelo, si existen mutaciones y cómo son éstas, y cómo se usa la función de adaptación.

Los algoritmos genéticos también pueden usarse para guiar a otros algoritmos de minería de datos en el proceso de aprendizaje. Así, por ejemplo, en las redes neuronales los algoritmos genéticos pueden usarse como un medio para ajustar los pesos reemplazando a la propagación hacia atrás. En este caso, los cromosomas contienen la información de los pesos. Los algoritmos genéticos, así como el uso de la lógica difusa en el aprendizaje, se ven en el Capítulo 15.

2.4.3 Construcción del modelo

Es en la construcción del modelo donde vemos mejor el carácter iterativo del proceso de KDD, ya que será necesario explorar modelos alternativos hasta encontrar aquel que resulte más útil para resolver nuestro problema. Así, una vez obtenido un modelo y a partir de los resultados obtenidos para el mismo, podríamos querer construir otro modelo usando la misma técnica pero otros parámetros, o quizás usar otras técnicas o herramientas. En esta búsqueda del “buen modelo” puede que tengamos que retroceder hasta fases anteriores y hacer cambios en los datos que estamos usando o incluso modificar la definición del problema. Es más, la elección de la tarea a realizar y del algoritmo a usar puede influir en la preparación de los datos (por ejemplo, un determinado algoritmo o técnica puede requerir que los datos se presenten en un formato determinado).

El proceso de construcción de modelos predictivos requiere tener bien definidas las etapas de entrenamiento y validación para asegurar que las predicciones serán robustas y precisas. La idea básica es estimar (o entrenar) el modelo con una porción de los datos (*training dataset*) y luego validarla con el resto de los datos (*test dataset*), como pasamos a ver en la sección siguiente.

2.5 Fase de evaluación e interpretación

Medir la calidad de los patrones descubiertos por un algoritmo de minería de datos no es un problema trivial, ya que esta medida puede añadir a varios criterios, algunos de ellos bastante subjetivos. Idealmente, los patrones descubiertos deben tener tres cualidades: ser precisos, comprensibles (es decir, inteligibles) e interesantes (útiles y novedosos). Según las aplicaciones puede interesar mejorar algún criterio y sacrificar ligeramente otro, como en el caso del diagnóstico médico que prefiere patrones comprensibles aunque su precisión no sea muy buena.

2.5.1 Técnicas de evaluación

Tal y como hemos dicho en el apartado anterior, para entrenar y probar un modelo se parten los datos en dos conjuntos: el conjunto de entrenamiento (*training set*) y el conjunto de prueba o de test (*test set*). Esta separación es necesaria para garantizar que la validación de la precisión del modelo es una medida independiente². Si no se usan conjuntos diferentes de entrenamiento y prueba, la precisión del modelo será sobreestimada, es decir, tendremos estimaciones muy optimistas.

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

El método de evaluación más básico, la **validación simple**, reserva un porcentaje de la base de datos como conjunto de prueba, y no lo usa para construir el modelo. Este porcentaje suele variar entre el cinco por ciento y el 50 por ciento. La división de los datos en estos dos grupos debe ser aleatoria para que la estimación sea correcta.

Si tenemos una cantidad no muy elevada de datos para construir el modelo, puede que no podamos permitirnos el lujo de reservar parte de los mismos para la etapa de evaluación. En estos casos se usa un método conocido como validación cruzada (*cross validation*). Los datos se dividen aleatoriamente en dos conjuntos equitativos con los que se estima la precisión predictiva del modelo. Para ello, primero se construye un modelo con el primer conjunto y se usa para predecir los resultados en el segundo conjunto y calcular así un ratio de error (o de precisión). A continuación, se construye un modelo con el segundo conjunto y se usa para predecir los resultados del primer conjunto, obteniéndose un segundo ratio de error. Finalmente, se construye un modelo con todos los datos, se calcula un promedio de los ratios de error y se usa para estimar mejor su precisión.

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

² Algunos algoritmos de aprendizaje utilizan *internamente* un tercer conjunto que extraen del conjunto de aprendizaje, denominado conjunto de validación (*validation dataset*), para refinar el modelo o elegir entre posibles modelos antes de la salida final del algoritmo. No hemos de confundir esta pre-validación o evaluación interna con la verdadera evaluación y, por tanto, el conjunto de validación con el conjunto de test.

Otra técnica para estimar el error de un modelo cuando se disponen de pocos datos es la conocida como *bootstrapping*. Ésta consiste en construir primero un modelo con todos los datos iniciales. Entonces, se crean numerosos conjuntos de datos, llamados *bootstrap samples*, haciendo un muestreo de los datos originales con reemplazo, es decir, se van seleccionando instancias del conjunto inicial, pudiendo seleccionar la misma instancia varias veces. Nótese que los conjuntos construidos de esta forma pueden contener datos repetidos. A continuación se construye un modelo con cada conjunto y se calcula su ratio de error sobre el conjunto de test (que son los datos sobrantes de cada muestreo). El error final estimado para el modelo construido con todos los datos se calcula promediando los errores obtenidos para cada muestra. Esta técnica se verá con más detalle en el Capítulo 17.

2.5.2 Medidas de evaluación de modelos

Dependiendo de la tarea de minería de datos existen diferentes medidas de evaluación de los modelos. Por ejemplo, en el contexto de la **clasificación**, lo normal es evaluar la calidad de los patrones encontrados con respecto a su *precisión predictiva*, la cual se calcula como el número de instancias del conjunto de prueba clasificadas correctamente dividido por el número de instancias totales en el conjunto de prueba. Tal y como hemos indicado en la sección anterior, el objetivo es obtener la mayor precisión posible sobre el conjunto de test, ya que obtener un 100 por cien de precisión sobre el conjunto de entrenamiento es trivial, bastaría con generar una regla para cada instancia usando una conjunción de sus variables-valores como antecedente de la regla (parte “**SI**”) y el valor a predecir como consecuente (parte “**ENTONCES**”). Por ejemplo, para la cuarta instancia del ejemplo de la Tabla 2.1 de la página 29 podríamos generar una regla como:

SI #instancia=4 **Y** edad=joven **Y** hijos=sí **Y** practica_deporte=no **Y** salario=bajo
ENTONCES buen_cliente=sí

O incluso, usando sólo algún atributo que pueda servir como clave primaria, por ejemplo:

SI #instancia=4 **ENTONCES** buen_cliente=sí

Si procedemos de igual forma para todas las instancias tendremos un 100 por cien de precisión predictiva respecto al conjunto de entrenamiento (ya que tendremos una regla por cada ejemplo). Sin embargo, su precisión con respecto a un conjunto de test que contenga instancias diferentes de las de entrenamiento será baja.

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

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

Siguiendo con el ejemplo de determinar si se juega a un deporte, consideremos:

#Instancia	Pronóstico	Humedad	Viento	Jugar
1	soleado	alta	débil	no
2	cubierto	alta	débil	sí
3	lluvioso	alta	débil	sí
4	lluvioso	normal	fuerte	no
5	soleado	normal	débil	sí

que son los mismos datos vistos anteriormente. Entonces, la regla

SI pronóstico=soleado **Y** viento=débil **ENTONCES** jugar=sí

tendrá una cobertura de 1, es decir, el número de días soleados y con viento débil en los que se recomienda jugar (instancia 5); y una confianza de 1/2 (ya que la regla también se puede aplicar a la instancia 1).

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

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

Además de las medidas comentadas, existen otras medidas más subjetivas, como el interés, la novedad, la simplicidad o la comprensibilidad que serán tratadas en el punto 17.8.

2.5.3 Interpretación y contextualización

Pese a todas las medidas vistas anteriormente, en muchos casos hay que evaluar también el contexto donde el modelo se va a utilizar. Por ejemplo, en el caso de la clasificación y las reglas de asociación, usar la precisión como medida de calidad tiene ciertas desventajas. En primer lugar, no tiene en cuenta el problema de tener distribuciones de clases no balanceadas, es decir, tener muchas instancias de unas clases y muy pocas o ninguna de otras. Esta situación es habitual en la detección de fraudes y en el diagnóstico médico. Un simple ejemplo puede ilustrar este punto. El servicio de urgencias de un hospital desea mejorar su sistema de admisión usando técnicas de aprendizaje inductivo. Cuando un paciente acude a las urgencias, una vez evaluado y realizadas las primeras atenciones, se pueden dar tres posibles situaciones que dependen de su estado: ser dado de alta, permanecer hospitalizado en observación o ser ingresado en la UCI (Unidad de Cuidados Intensivos). El porcentaje de casos de cada una de estas tres clases es, por ejemplo, del 86,5 por ciento, 13 por ciento y 0,5 por ciento respectivamente. Usando un algoritmo de minería sobre los datos referentes a pacientes pasados podríamos obtener un modelo que siempre diera de alta a todos los pacientes. En términos de precisión, éste sería un buen modelo ya

que su precisión sería del 86,5 por ciento. Sin embargo, este modelo es inútil. Más importante todavía, puede ser peligroso ya que pacientes cuyas condiciones de salud requerirían un ingreso en la UCI serían enviados a sus casas, lo cual es una política desastrosa para el hospital.

El ejemplo anterior pone de manifiesto que necesitamos conocer mejor el tipo de errores y su coste asociado. En los problemas de clasificación se usa una **matriz de confusión**, la cual muestra el recuento de casos de las clases predichas y sus valores actuales. Si se dispone de información sobre el coste de cada error/acierto en la clasificación, entonces las celdas de la matriz pueden asociarse con el coste de cometer un cierto error de clasificación o de efectuar una clasificación correcta. En este caso, la matriz suele denominarse **matriz de costes** de clasificación. Con estas dos matrices podemos evaluar los modelos con sus costes de error de clasificación y, por ejemplo, buscar un modelo que minimice el coste global. Veremos ejemplos de matrices de confusión y de costes en el Capítulo 17.

La consideración de que todos los errores no son iguales puede incluso tenerse en cuenta en situaciones donde los costes de error suelen ser difíciles de estimar o incluso desconocidos para muchas aplicaciones. En estos casos, se usan estrategias alternativas como el **análisis ROC** (*Receiver Operating Characteristic*) que también veremos en el Capítulo 17.

Como hemos dicho anteriormente, la precisión de un modelo no garantiza que refleje el mundo real. Normalmente, esta situación se produce cuando al construir el modelo no hemos tenido en cuenta algunos parámetros que implícitamente influyen en él. Por ejemplo, un modelo para predecir el gasto energético de una población puede fallar al no haber considerado que un período de recesión económica mundial puede implicar subidas considerables en el precio del petróleo, lo que condiciona el gasto que de esta fuente de energía hace la población.

En cualquier caso deberemos contrastar el conocimiento que éste nos proporciona con el conocimiento previo que pudiéramos tener sobre el problema para detectar y en su caso resolver los posibles conflictos.

2.6 Fase de difusión, uso y monitorización

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

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

También es importante medir lo bien que el modelo evoluciona. Aun cuando el modelo funcione bien debemos continuamente comprobar las prestaciones del mismo. Esto se debe

principalmente a que los patrones pueden cambiar. Por ejemplo, todos los vendedores saben que las ventas se ven afectadas por factores externos como la tasa de inflación, la cual altera el comportamiento de compra de la gente. Por lo tanto, el modelo deberá ser monitorizado, lo que significa que de tiempo en tiempo el modelo tendrá que ser re-evaluado, re-entrenado y posiblemente reconstruido completamente.

PARTE II

PREPARACIÓN

DE DATOS

En esta parte se detallan las primeras fases: recopilación, limpieza y transformación. Para ello se introducen una serie de tecnologías: almacenes de datos, OLAP, técnicas simples del análisis multivariante, tratamiento de la dimensionalidad, de datos faltantes y anómalos, visualización básica, lenguajes de consulta, primitivas de minería de datos, etc.

CAPÍTULOS

- 3. Recopilación. Almacenes de datos**
- 4. Limpieza y transformación**
- 5. Exploración y selección**

Capítulo 3

RECOPILACIÓN.

ALMACENES DE DATOS

Para poder comenzar a analizar y extraer algo útil de los datos es preciso, en primer lugar, disponer de ellos. Esto en algunos casos puede parecer trivial; se parte de un simple archivo de datos que analizar. En otros, la diversidad y tamaño de las fuentes hace que el proceso de recopilación de datos sea una tarea compleja, que requiere una metodología y una tecnología propias. En general, el problema de reunir un conjunto de datos que posibilite la extracción de conocimiento requiere decidir, entre otros aspectos, de qué fuentes, internas y externas, se van a obtener los datos, cómo se van a organizar, cómo se van a mantener con el tiempo y, finalmente, de qué forma se van a poder extraer parcial o totalmente, en detalle o agregados, con distintas “vistas minables” a las que podamos aplicar las herramientas concretas de minería de datos.

En este capítulo³ nos centramos en las metodologías y tecnologías para realizar esta recopilación e integración. En particular introducimos la tecnología de los almacenes de datos y algunos conceptos relacionados, como las herramientas OLAP (*On-Line Analytical Processing*). Los almacenes de datos no son estrictamente necesarios para realizar minería de datos, aunque sí extremadamente útiles si se va a trabajar con grandes volúmenes de datos, que varían con el tiempo y donde se desea realizar tareas de minerías de datos variadas, abiertas y cambiantes. Es importante destacar las diferencias entre el análisis que se puede realizar con técnicas OLAP y con minería de datos (aunque exista un cierto solapamiento entre ambas), así como comprender que ambas tecnologías son complementarias. Finalmente, la selección, transformación y limpieza de datos serán tratadas en el capítulo siguiente, aunque algunas de estas operaciones puedan hacerse antes o durante el proceso de recopilación e integración.

³ Para la comprensión de este capítulo se asumen unos conocimientos básicos de bases de datos, especialmente bases de datos relacionales. Un buen libro introductorio es [Celma et al. 2003].

3.1 Introducción

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

Al igual que la tecnología de bases de datos ha desarrollado una serie de modelos de datos (como el relacional), de lenguajes de consulta y actualización, de reglas de actividad, etc., para trabajar con la información transaccional de una organización, veremos que existe una tecnología relativamente reciente, denominada “almacenes de datos” (*data warehouses*) que pretende proporcionar metodologías y tecnología para recopilar e integrar los datos históricos de una organización, cuyo fin es el análisis, la obtención de resúmenes e informes complejos y la extracción de conocimiento. Esta tecnología está diseñada especialmente para organizar grandes volúmenes de datos de procedencia generalmente estructurada (bases de datos relacionales, por ejemplo), aunque el concepto general es útil para la organización de pequeños conjuntos de datos en aplicaciones de minería de datos más modestas.

Supóngase que en una compañía bien implantada en el ámbito europeo queremos analizar aquellos países y gamas de productos en los que las ventas vayan excepcionalmente bien (con el objetivo, por ejemplo, de premiar a las oficinas comerciales de cada gama y producto) o, dicho de una manera más técnica, averiguar si la penetración relativa (teniendo en cuenta la permeabilidad del país en cuestión) de una gama de productos es significativamente mayor que la media de penetración en el conjunto del continente. La compañía dispone, por supuesto, de una base de datos transaccional sobre la que operan todas las aplicaciones de la empresa: producción, ventas, facturación, proveedores, nóminas, etc. Lógicamente, de cada venta se registra la fecha, la cantidad y el comprador y, de éste, el país. Con toda esta información histórica nos podemos preguntar: ¿es esta información suficiente para realizar el análisis anterior? La respuesta, a primera vista, quizás de manera sorprendente, es negativa. Pero, aparentemente, si tenemos detalladas las ventas de tal manera que una consulta SQL puede calcular las ventas por países de todos los productos y gamas, ¿qué más puede faltar?

Sencillamente, la respuesta hay que buscarla fuera de la base de datos, en el contexto donde se motiva el análisis. La penetración de un producto depende de las ventas *por habitante*. Si no tenemos en cuenta la población de cada país la respuesta del análisis estará sesgada; será muy probable que entre los países con mayor penetración siempre esté Alemania, y entre los países con menor penetración se encuentre San Marino. Pero no sólo eso, es posible que, si deseamos hacer un análisis más perspicaz, nos interese saber la renta per cápita de cada país o incluso la distribución por edad de cada país. Dependiendo de la gama, nos puede interesar información externa verdaderamente específica. Por ejemplo, las

horas de sol anuales de cada país pueden ser una información valiosísima para una compañía de cosméticos. Lógicamente es más difícil vender bronzeadores en Lituania que en Grecia o, dicho más técnicamente, Lituania tiene menos permeabilidad a la gama de bronzeadores que Grecia. Pero este hecho, que nos parece tan lógico, sólo podrá ser descubierto por nuestras herramientas de minería de datos si somos capaces de incorporar información relativa a las horas de sol o, al menos, cierta información climática de cada país.

Evidentemente, cada organización deberá recoger diferente información que le pueda ser útil para la tarea de análisis, extracción de conocimiento y, en definitiva, de toma de decisiones. En la Figura 3.1 se muestran las fuentes de datos que pueden ser requeridas en el caso anterior para un proceso de extracción de conocimiento satisfactorio. Sólo conociendo el contexto de cada organización o de cada problema en particular se puede determinar qué fuentes externas van a ser necesarias. Además, este proceso es generalmente iterativo. A medida que se va profundizando en un estudio, se pueden ir determinando datos externos que podrían ayudar y se pueden ir añadiendo a nuestro “repositorio de datos”. Por tanto, la tarea de mantener un “repositorio” o un “almacén” con toda la información necesaria cobra mayor relevancia y complejidad.

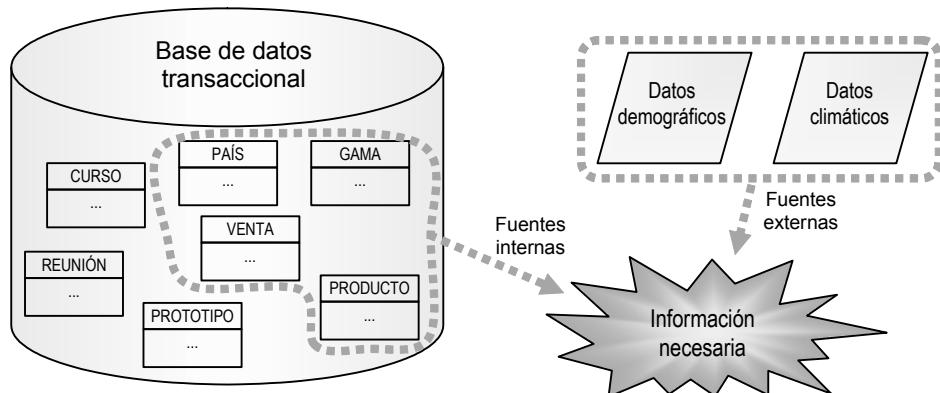


Figura 3.1. Fuentes de datos requeridas para responder “países con mayor penetración de bronzeadores”.

El mantenimiento de esta información plantea cuestiones técnicas. En primer lugar, se requerirá añadir, puede que frecuentemente, nueva información a nuestro repositorio, tanto proveniente de actualizaciones de la propia organización como de fuentes externas, ya sean actualizaciones como nuevas incorporaciones. En segundo lugar, y la que resulta la cuestión principal, ¿hay que almacenar toda esta información en la base de datos transaccional? Puestos en el ejemplo anterior, ¿requieren las aplicaciones diarias de la organización almacenar en una tabla de la base de datos la temperatura media de Lituania?

Estas y otras cuestiones, como veremos a continuación, han motivado el desarrollo de una tecnología nueva y específica, denominada “almacenes de datos” (*data warehouses*⁴).

⁴ La traducción en castellano no es unánime. En partes de Latinoamérica se conocen también como “bodegas de datos”.

3.2 Necesidad de los almacenes de datos

La proliferación de sistemas de información sustentados en bases de datos ha generalizado el uso de herramientas que permiten obtener informes complejos, resúmenes e incluso estadísticas globales sobre la información almacenada con el objetivo de asistir en la toma de decisiones. La mayoría de sistemas comerciales de gestión de bases de datos incluyen herramientas de “informes avanzados”, “inteligencia de negocio” (*business intelligence*), sistemas de información ejecutivos (EIS, *Executive Information Systems*) y otras, que pese a sus nombres variados intentan realizar un procesamiento analítico de la información, más que el procesamiento transaccional habitual realizado por las aplicaciones del día a día de la organización.

Por tanto, cada día es más necesario distinguir dos usos diferentes del sistema de información: el procesamiento transaccional y el procesamiento analítico.

3.2.1 OLTP y OLAP

Con las siglas OLTP y OLAP se denominan dos tipos de procesamiento bien diferentes:

- **OLTP (*On-Line Transactional Processing*)**. El procesamiento transaccional en tiempo real constituye el trabajo primario en un sistema de información. Este trabajo consiste en realizar transacciones, es decir, actualizaciones y consultas a la base de datos con un objetivo operacional: hacer funcionar las aplicaciones de la organización, proporcionar información sobre el estado del sistema de información y permitir actualizarlo conforme va variando la realidad del contexto de la organización. Muestras de este tipo de trabajo transaccional son, por ejemplo, en el caso de una empresa, la inserción de un nuevo cliente, el cambio de sueldo de un empleado, la tramitación de un pedido, el almacenamiento de una venta, la impresión de una factura, la baja un producto, etc. Es el trabajo diario y para el que inicialmente se ha diseñado la base de datos.
- **OLAP (*On-Line Analytical Processing*)**. El procesamiento analítico en tiempo real engloba un conjunto de operaciones, exclusivamente de consulta, en las que se requiere agregar y cruzar gran cantidad de información. El objetivo de estas consultas es realizar informes y resúmenes, generalmente para el apoyo en la toma de decisiones. Ejemplos de este tipo de trabajo analítico pueden ser resúmenes de ventas mensuales, los consumos eléctricos por días, la espera media de los pacientes en cirugía digestiva de un hospital, el producto cuyas ventas han crecido más en el último trimestre, las llamadas por horas, etc. Este tipo de consultas suelen emanarse de los departamentos de dirección, logística o prospectiva y requieren muchos recursos.

Una característica de ambos procesamientos es que se pretende que sean “on-line”, es decir, que sean relativamente “instantáneos” y se puedan realizar en cualquier momento (en tiempo real). Esto parece evidente e imprescindible para el OLTP, pero no está tan claro que esto sea posible para algunas consultas muy complejas realizadas por el OLAP.

La práctica general, hasta hace pocos años, y todavía existente en muchas organizaciones y empresas, es que ambos tipos de procesamiento (OLTP y OLAP) se realizaran sobre la misma base de datos transaccional. De hecho, una de las máximas de la tecnología de

base de datos era la eliminación de redundancia, con lo que parecía lo más lógico que ambos procesamientos trabajaran sobre una única base de datos general (aunque pudiera tener diferentes vistas para diferentes aplicaciones, procesamientos o servicios).

Esta práctica plantea dos problemas fundamentales:

- Las consultas OLAP perturban el trabajo transaccional diario de los sistemas de información originales. Al ser consultas complejas y que involucran muchas tablas y agrupaciones, suelen consumir gran parte de los recursos del sistema de gestión de base de datos. El resultado es que durante la ejecución de estas consultas, las operaciones transaccionales normales (OLTP), se resienten: las aplicaciones van más lentas, las actualizaciones se demoran muchísimo y el sistema puede incluso llegar a colapsarse. De este hecho viene el nombre familiar que se les da a las consultas OLAP: *“killer queries”* (consultas asesinas). Como consecuencia, muchas de estas consultas se deben realizar por la noche o en fines de semana, con lo que en realidad dejan de ser *“on-line”*.
- La base de datos está diseñada para el trabajo transaccional, no para el análisis de los datos. Esto significa que, aunque tuviéramos el sistema dedicado exclusivamente para realizar una consulta OLAP, dicha consulta puede requerir mucho tiempo, pero no sólo por ser compleja intrínsecamente, sino porque el esquema de la base de datos no es el más adecuado para este tipo de consultas.

Ambos problemas implican que va a ser prácticamente imposible (a un coste de *hardware* razonable, lógicamente) realizar un análisis complejo de la información en tiempo real si ambos procesamientos se realizan sobre la misma base de datos.

Afortunadamente, debido a que los costes de almacenamiento masivo y conectividad se han reducido drásticamente en los últimos años, parece razonable recoger (copiar) los datos en un sistema unificado y diferenciado del sistema tradicional transaccional u operacional. Aunque esto vaya contra la filosofía general de bases de datos, son muchas más las ventajas que los inconvenientes, como veremos a continuación. Desde esta perspectiva, se separa definitivamente la base de datos con fines transaccionales de la base de datos con fines analíticos. Nacen los almacenes de datos.

3.2.2 Almacenes de datos y bases de datos transaccionales

Un almacén de datos es un conjunto de datos históricos, internos o externos, y descriptivos de un contexto o área de estudio, que están integrados y organizados de tal forma que permiten aplicar eficientemente herramientas para resumir, describir y analizar los datos con el fin de ayudar en la toma de decisiones estratégicas.

La ventaja fundamental de un almacén de datos es su diseño específico y su separación de la base de datos transaccional. Un almacén de datos:

- Facilita el análisis de los datos en tiempo real (OLAP).
- No disturba el OLTP de las bases de datos originales.

A partir de ahora, por tanto, diferenciaremos claramente entre bases de datos transaccionales (u operacionales) y almacenes de datos. Dicha diferencia, además, se ha ido marcando más profundamente a medida que las tecnologías propias de ambas bases de datos (y en

especial la de almacenes de datos) se han ido especializando. De hecho, hoy en día, las diferencias son claras, como se muestra en la Tabla 3.1.

Las diferencias mostradas en la tabla, como veremos, distinguen claramente la manera de estructurar y diseñar almacenes de datos respecto a la forma tradicional de hacerlo con bases de datos transaccionales.

	BASE DE DATOS TRANSACCIONAL	ALMACÉN DE DATOS
Propósito	Operaciones diarias. Soporte a las aplicaciones.	Recuperación de información, informes, análisis y minería de datos.
Tipo de datos	Datos de funcionamiento de la organización.	Datos útiles para el análisis, la sumarización, etc.
Características de los datos	Datos de funcionamiento, cambiantes, internos, incompletos...	Datos históricos, datos internos y externos, datos descriptivos...
Modelo de datos	Datos normalizados.	Datos en estrella, en copo de nieve, parcialmente desnormalizados, multidimensionales...
Número y tipo de usuarios	Cientos/miles: aplicaciones, operarios, administrador de la base de datos.	Decenas: directores, ejecutivos, analistas (granjeros, mineros).
Acceso	SQL. Lectura y escritura.	SQL y herramientas propias (<i>slice & dice, drill, roll, pivot...</i>). Lectura.

Tabla 3.1. Diferencias entre la base de datos transaccional y el almacén de datos.

Aunque ambas fuentes de datos (transaccional y almacén de datos) están separadas, es importante destacar que gran parte de los datos que se incorporan en un almacén de datos provienen de la base de datos transaccional. Esto supone desarrollar una tecnología de volcado y mantenimiento de datos desde la base de datos transaccional al almacén de datos. Además, el almacén de datos debe integrar datos externos, con lo que en realidad debe estar actualizándose frecuentemente de diferentes fuentes. El almacén de datos pasa a ser un integrador o recopilador de información de diferentes fuentes, como se observa en la Figura 3.2.

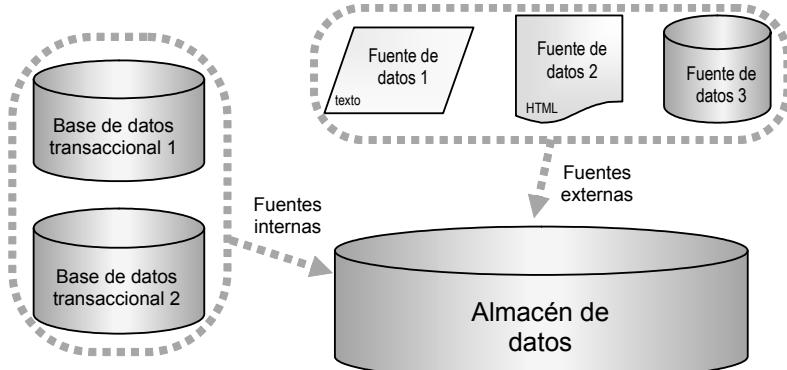


Figura 3.2. El almacén de datos como integrador de diferentes fuentes de datos.

La organización y el mantenimiento de esta información plantea cuestiones técnicas, fundamentalmente sobre cómo diseñar el almacén de datos, cómo cargarlo inicialmente, cómo mantenerlo y preservar su consistencia. No obstante, son muchas más las ventajas de esta separación que sus inconvenientes. Además, esta separación facilita la incorporación de fuentes externas, que, en otro caso, sería muy difícil de encajar en la base de datos transaccional.

3.3 Arquitectura de los almacenes de datos

Un almacén de datos recoge, fundamentalmente, datos históricos, es decir, *hechos*, sobre el contexto en el que se desenvuelve la organización. Los hechos son, por tanto, el aspecto central de los almacenes de datos. Esta característica determina en gran medida la manera de organizar los almacenes de datos.

3.3.1 Modelo multidimensional

El modelo *conceptual* de datos más extendido para los almacenes de datos es el **modelo multidimensional**. Los datos se organizan en torno a los *hechos*, que tienen unos atributos o *medidas* que pueden verse en mayor o menor detalle según ciertas *dimensiones*. Por ejemplo, una gran cadena de supermercados puede tener como hechos básicos las *ventas*. Cada venta tiene unas medidas: importe, cantidad, número de clientes, etc., y se puede detallar o agregar en varias dimensiones: tiempo de la venta, producto de la venta, lugar de la venta, etc. Es esclarecedor comprobar que las medidas responden generalmente a la pregunta “cuánto”, mientras que las dimensiones responderán al “cuándo”, “qué”, “dónde”, etc.

Lo realmente interesante del modelo es que ha de permitir, de una manera sencilla, obtener información sobre hechos a diferentes niveles de agregación. Por ejemplo, el hecho “El día 20 de mayo de 2003 la empresa vendió en España 12.327 unidades de productos de la categoría *insecticidas*” representa una medida (cantidad, 12.327 unidades) de una venta con granularidad día para la dimensión tiempo (20 de mayo de 2003), con granularidad país para la dimensión lugar (España) y con granularidad categoría (*insecticidas*) para la dimensión de productos. Del mismo modo, el hecho “El primer trimestre de 2004 la empresa vendió en Valencia por un importe de 22.000 euros del producto *Androbrío 33 cl.*” representa una medida (importe, 22.000 euros) de una venta con granularidad trimestre para la dimensión tiempo (primer trimestre de 2004), con granularidad ciudad para la dimensión lugar (Valencia) y con granularidad artículo (*Androbrío 33 cl.*) para la dimensión de productos.

En la Figura 3.3 se representa parte de un almacén de datos con estructura multidimensional de donde se pueden extraer estos dos hechos.

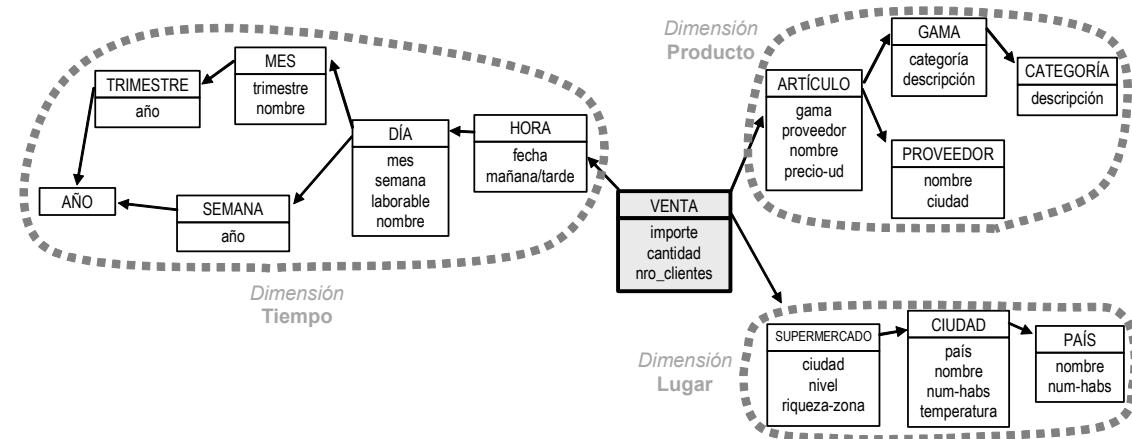


Figura 3.3. Información sobre ventas en un almacén de datos representado bajo un modelo multidimensional.

La Figura 3.3 no se basa en ningún modelo de datos en particular (por ejemplo el relacional). Nótese que no estamos hablando de que cada rectángulo de la figura sea una tabla o que las flechas sean claves ajenas. Al contrario, simplemente estamos representando datos de una manera conceptual. Mostramos los hechos “venta” y tres dimensiones con varios niveles de agregación. Las flechas se pueden leer como “se agrega en”. Como se observa en la figura, cada dimensión tiene una estructura jerárquica pero no necesariamente lineal. Por ejemplo, en las dimensiones *tiempo* y *producto* hay más de un camino posible de agregación (ruta de agregación). Incluso, en el caso de los productos, el nivel de agregación mayor puede ser diferente (hacia categoría o hacia proveedor). Esto permite diferentes niveles y caminos de agregación para las diferentes dimensiones, posibilitando la definición de hechos agregados con mucha facilidad. La forma que tienen estos conjuntos de hechos y sus dimensiones hace que se llamen popularmente almacenes de datos en “estrella simple” (cuando no hay caminos alternativos en las dimensiones) o de “estrella jerárquica” o “copo de nieve” (cuando sí hay caminos alternativos en las dimensiones, como el ejemplo anterior).

Cuando el número de dimensiones no excede de tres (o se agregan completamente el resto) podemos representar cada combinación de niveles de agregación como un cubo. El cubo está formado por casillas, con una casilla para cada valor entre los posibles para cada dimensión a su correspondiente nivel de agregación. Sobre esta “vista”, cada casilla representa un hecho. Por ejemplo, en la Figura 3.4 se representa un cubo tridimensional donde las dimensiones producto, lugar y tiempo se han agregado por artículo, ciudad y trimestre. La representación de un hecho como el visto anteriormente corresponde, por tanto, a una casilla en dicho cubo. El valor de la casilla es la medida observada (en este caso, importe de las ventas).

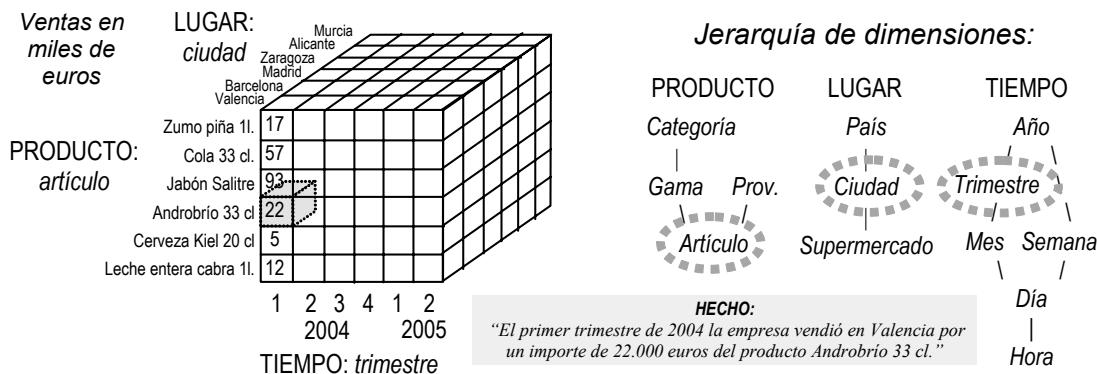


Figura 3.4. Visualización de un hecho en un modelo multidimensional.

Esta visualización hace que, incluso cuando tengamos más de tres dimensiones, se hable de un “cubo” (o más propiamente de “hipercubo”) como un conjunto de niveles de agregación para todas las dimensiones.

Esta estructura permite ver de una manera intuitiva la sumarización/agregación (varias casillas se fusionan en casillas más grandes), la disagregación (las casillas se separan en casillas con mayor detalle) y la navegación según las dimensiones de la estrella.

3.3.2 Datamarts

En algunos casos puede parecer intuitivo organizar la información en dimensiones. El caso de las ventas es el ejemplo más ilustrativo. En general, cierta información es más fácilmente representable de esta forma, pero siempre se puede llegar a una estructura de este tipo. Lo que no es posible, en general, es la representación de todo el almacén de datos como una sola estrella, ni siquiera jerárquica. Por ejemplo, la información de personal de una empresa (empleados, departamentos, proyectos, etc.) es difícilmente integrable en la misma estrella que las ventas. Incluso, en ámbitos más relacionados de una organización (por ejemplo ventas y producción) esto tampoco es posible. La idea general es que para cada subámbito de la organización se va a construir una estructura de estrella. Por tanto, el almacén de datos estará formado por muchas estrellas (jerárquicas o no), formando una “constelación”. Por ejemplo, aparte de la estrella jerárquica para las ventas, podríamos tener otra estrella para personal. En este caso, los hechos podrían ser que un empleado ha dedicado ciertos recursos en un proyecto durante un período en un departamento. Los hechos podrían llamarse “participaciones”. Las medidas o atributos podrían ser “horas de participación”, “número de participantes”, “presupuesto”, “nivel de éxito del proyecto”, etc. y las dimensiones podrían ser “tiempo” (para representar el período en el que ha estado involucrado), “departamento” (para representar un empleado, equipo, departamento o división en la que se ha desarrollado) y el “proyecto” (subproyecto, proyecto o programa).

Cada una de estas estrellas que representan un ámbito específico de la organización se denomina popularmente “datamart” (mercado de datos). Lógicamente, cada datamart tendrá unas medidas y unas dimensiones propias y diferentes de los demás. La única dimensión que suele aparecer en todos los datamarts es la dimensión *tiempo*, ya que el almacén de datos representa información histórica y, por tanto, siempre es de interés ser capaz de agregarlo por intervalos de diferente detalle.

En la Figura 3.5 se muestra un almacén de datos compuesto de varios datamarts.

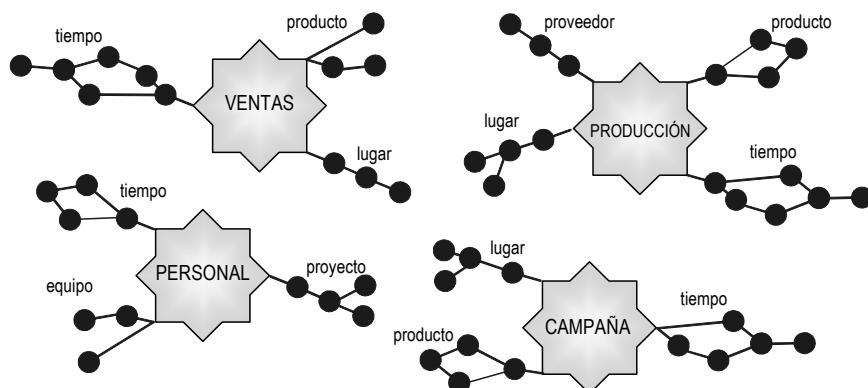


Figura 3.5. Representación icónica de un almacén de datos compuesto por varios datamarts.

Aparentemente, da la impresión de que el almacén de datos puede contener mucha información redundante, especialmente sobre las dimensiones. Aunque, en general, los almacenes de datos contienen información redundante, la estructura anterior es la estructura externa, visible o conceptual. Esta estructura no determina la manera de implementarlo ni lógica ni físicamente, como veremos.

3.3.3 Explotación de un almacén de datos. Operadores

En realidad, un modelo de datos se compone de unas estructuras y unos operadores sobre dichas estructuras. Acabamos de ver que el modelo multidimensional se basa en un conjunto de datamarts, que, generalmente, son estructuras de datos en estrella jerárquica. Para completar el modelo multidimensional debemos definir una serie de operadores sobre la estructura. Los operadores más importantes asociados a este modelo son:

- **Drill:** se trata de disagregar los datos (mayor nivel de detalle o desglose, menos sumarización) siguiendo los caminos de una o más dimensiones.
- **Roll:** se trata de agregar los datos (menor nivel de detalle o desglose, más sumarización o consolidación) siguiendo los caminos de una o más dimensiones.
- **Slice & Dice:** se seleccionan y se proyectan datos.
- **Pivot:** se reorientan las dimensiones.

Normalmente, estos operadores se llaman operadores OLAP, operadores de análisis de datos u operadores de almacenes de datos. Para explicar estos operadores hemos de pensar que partimos, además, de unos operadores genéricos básicos, que permiten realizar consultas, vistas o informes sobre la estructura estrella, generalmente de una forma gráfica. Estos operadores básicos permiten realizar las mismas consultas de proyección, selección y agrupamiento que se pueden hacer en SQL. En muchos casos, de hecho, se puede editar la consulta SQL correspondiente, aunque ésta se haya hecho gráficamente.

Por tanto, el primer paso para poder utilizar los operadores propios del modelo multidimensional es definir una consulta. En realidad, como veremos a continuación, los operadores *drill*, *roll*, *slice & dice* y *pivot*, son modificadores o refinadores de consulta y sólo pueden aplicarse sobre una consulta realizada previamente.

Consideremos por ejemplo la consulta “obtener para cada categoría y trimestre el total de ventas” para el datamart de la Figura 3.3. En un entorno gráfico, dicha consulta se podría realizar eligiendo el nivel “categoría” para la dimensión “producto” (obteniendo además sólo dos categorías: “refrescos” y “congelados”), el nivel “trimestre” para la dimensión “tiempo” y no escogiendo la dimensión “lugar” (o considerando que se considera el nivel más agregado, es decir, todo el datamart). Además, se elegiría la propiedad que se desea (“importe”).

Dependiendo del sistema o de la manera que hayamos elegido, el resultado se nos puede mostrar de manera tabular o de manera matricial, como se observa en la Figura 3.6, aunque la información mostrada es la misma. En este caso, como hay pocas dimensiones, la representación matricial parece más adecuada.

La existencia de dimensiones y atributos facilita, en gran medida, la realización de consultas y éstas se suelen hacer arrastrando con el ratón las medidas y dimensiones deseadas. No es necesario mucho más para realizar informes sencillos sobre ese datamart. No obstante, lo interesante empieza justamente cuando intentamos modificar el informe (una consulta, al fin y al cabo). A veces, querremos mayor nivel de detalle, otras veces menos, o bien desearemos añadir o quitar alguna dimensión, o modificar el informe en cualquier otro sentido.

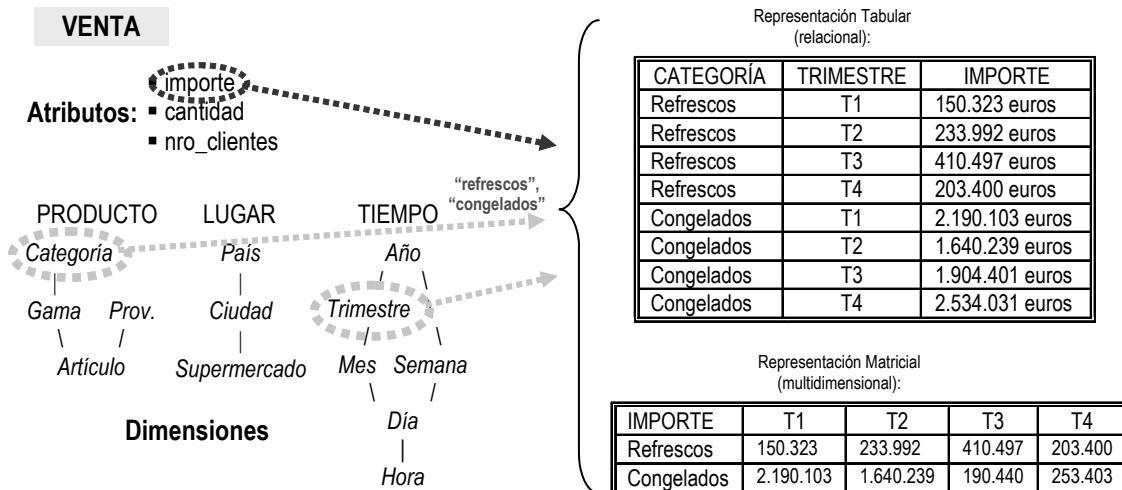


Figura 3.6. Construcción de una consulta seleccionando niveles de dimensiones.

Por ejemplo, supongamos que queremos ahora ver sólo las ventas de refrescos y desglosarlo por ciudades (en particular por dos: Valencia y León) con el objetivo de ver si los hábitos estacionales (hay más consumo de refrescos en estaciones calurosas) son generales en todas las áreas geográficas. Una forma obvia de hacer esto sería realizar un nuevo informe. Lo interesante de los nuevos operadores *drill*, *roll*, *slice & dice* y *pivot*, es que permiten modificar la consulta realizada, sin necesidad de realizar otra. En realidad son "navegadores" de informes, más que operadores por sí mismos. Por ejemplo, en el caso anterior, podemos utilizar el operador *drill*. Este operador permite entrar más al detalle en el informe. En particular, sólo es necesario que desglose la información por ciudades (en concreto, restringiéndose a sólo Valencia y León) y además seleccionando sólo la categoría "refrescos". La transformación producida tras esta operación se ilustra en la Figura 3.7, en la que se muestra cómo cambia la vista tanto en la representación relacional como en la multidimensional (la información mostrada en ambas representaciones siempre es la misma). En el resultado se puede observar que la distribución de ventas en Valencia (claramente estacional) difiere claramente de la de León (prácticamente no estacional).

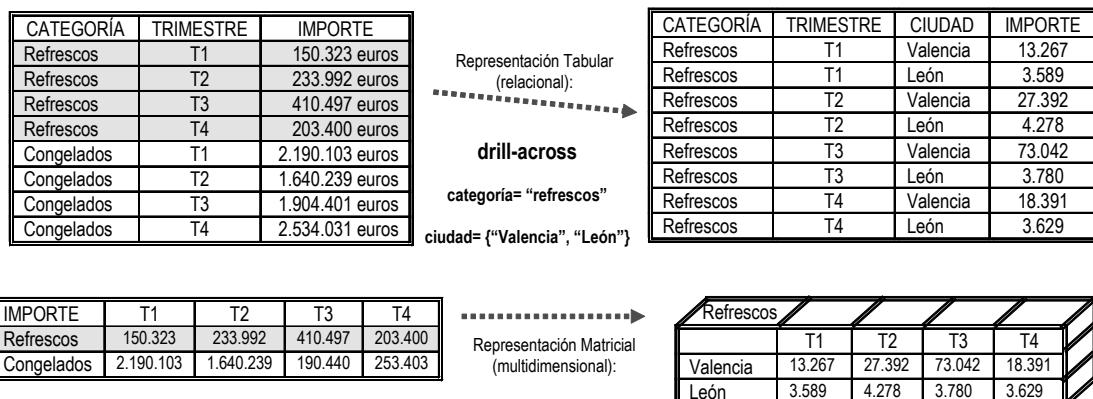


Figura 3.7. Ejemplo del operador "drill".

Lo importante de estos operadores es que modifican el informe en tiempo real y no generan uno nuevo. Lógicamente, para que esto sea eficiente el almacén de datos ha de estar diseñado e implementado para que este tipo de operaciones utilicen ciertas estructuras intermedias que permitan agregar y disagregar con facilidad.

Veamos ahora un ejemplo de la operación *roll*. Simplemente la operación *roll* es la inversa del *drill* y el objetivo es obtener información más agregada.

Representación Tabular (relacional):

CATEGORÍA	TRIMESTRE	IMPORTE
Refrescos	T1	150.323 euros
Refrescos	T2	233.992 euros
Refrescos	T3	410.497 euros
Refrescos	T4	203.400 euros
Congelados	T1	2.190.103 euros
Congelados	T2	1.640.239 euros
Congelados	T3	1.904.401 euros
Congelados	T4	2.534.031 euros

roll-across

un nivel por "tiempo"

CATEGORÍA	IMPORTE
Refrescos	998.212 euros
Congelados	10.458.877 euros

Figura 3.8. Ejemplo del operador *roll*.

Por ejemplo, si quisiéramos obtener los totales de las categorías “refrescos” y “congelados”, simplemente sería necesario aplicar el operador *roll-across* a la consulta original, sin necesidad de crear una nueva, como se observa en la Figura 3.8.

Vistos los operadores *drill* y *roll*, cabe preguntarse por qué a veces se utiliza la notación “-across” (como hemos hecho nosotros) y a veces la notación “-up” (que incluso es más frecuente). Aunque en realidad es una cuestión meramente terminológica y no universalmente respetada, las correspondencias son las siguientes:

- *Drill-down* y *roll-up*: representan agregaciones o disagregaciones dentro de una dimensión ya definida inicialmente en la consulta.
- *Drill-across* y *roll-across*: representan agregaciones o disagregaciones en otras dimensiones de las definidas inicialmente en la consulta o hacen desaparecer alguna de las dimensiones.

Finalmente, veamos los otros dos operadores: *pivot* y *slice & dice*. Estos dos operadores se utilizan exclusivamente cuando se hace una representación matricial, o al menos una representación mixta.

Veamos en primer lugar el operador *pivot*. Supongamos que tenemos la consulta en la situación en la que estamos mostrando el importe para las categorías “refrescos” y “congelados”, las ciudades “Valencia” y “León”, y todos los trimestres. La posible representación (mixta, entre tabular y multidimensional) es la que se muestra en la parte izquierda de la Figura 3.9.

Representación Mixta:

CATEGORÍA	TRIMESTRE	Valencia	León
Refrescos	T1	13.267	3.589
Refrescos	T2	27.392	4.278
Refrescos	T3	73.042	3.780
Refrescos	T4	18.391	3.629
Congelados	T1	150.242	4.798
Congelados	T2	173.105	3.564
Congelados	T3	163.240	4.309
Congelados	T4	190.573	4.812

pivot

categoría x ciudad

CATEGORÍA	TRIMESTRE	Refrescos	Congelados
Valencia	T1	13.267	150.242
Valencia	T2	27.392	173.105
Valencia	T3	73.042	163.240
Valencia	T4	18.391	190.573
León	T1	3.589	4.798
León	T2	4.278	3.564
León	T3	3.780	4.309
León	T4	3.629	4.812

Figura 3.9. Ejemplo del operador *pivot*.

El operador *pivot* permite cambiar algunas filas por columnas. Esta operación, aparentemente sencilla, no está generalizada en muchos sistemas de bases de datos (en SQL-92 no existía, por ejemplo). No obstante, su inclusión es prácticamente imprescindible para poder realizar análisis de datos y, muy en particular, minería de datos. Como veremos, este cambio permite que valores de columnas pasen a ser nombre de nuevas columnas y viceversa. Como se ve en la parte derecha de la Figura 3.9 esto supone que ciertos métodos de aprendizaje proposicionales sean capaces de extraer patrones sobre la consulta de la izquierda y no sean capaces de hacerlo sobre la segunda y viceversa.

CATEGORÍA	TRIMESTRE	Valencia	León
Refrescos	T1	13.267	3.589
Refrescos	T2	27.392	4.278
Refrescos	T3	73.042	3.780
Refrescos	T4	18.391	3.629
Congelados	T1	150.242	4.798
Congelados	T2	173.105	3.564
Congelados	T3	163.240	4.309
Congelados	T4	190.573	4.812

Representación Mixta

slice & dice

CATEGORÍA	Trimestre	Valencia
Refrescos	T1	13.267
Refrescos	T4	18.391
Congelados	T1	150.242
Congelados	T4	190.573

trimestre = {T1, T4}
ciudad = Valencia

Figura 3.10. Ejemplo del operador slice & dice.

Veamos finalmente el operador *slice & dice*. En realidad este operador permite escoger parte de la información mostrada, no por agregación sino por selección. En la Figura 3.10 se muestra un ejemplo de este operador.

Los operadores vistos son los básicos para refinar una consulta o informe, aunque distintos sistemas propietarios pueden añadir más operadores, maneras diferentes de representar los datos, de interpretar la petición de aplicación de operadores (mediante arrastre de dimensiones utilizando el ratón), etc. En los dos capítulos siguientes veremos cómo estos operadores pueden facilitar en gran medida la transformación y adecuación de datos de cara a obtener una “vista minable” que sea idónea para aplicar técnicas de minería de datos.

3.3.4 Implementación del almacén de datos. Diseño

Recordemos que una de las razones para crear un almacén de datos separado de la base de datos operacional era conseguir que el análisis se pudiera realizar de una manera eficiente. El hecho de que la estructura anterior y los operadores vistos permitan trabajar sencillamente y combinar dimensiones, detallar o agregar informes, etc., y todo ello de manera gráfica, no asegura que esto sea eficiente.

Con el objetivo de obtener la eficiencia deseada, los sistemas de almacenes de bases de datos pueden implementarse utilizando dos tipos de esquemas físicos⁵:

- **ROLAP (Relational OLAP)**: físicamente, el almacén de datos se construye sobre una base de datos relacional.
- **MOLAP (Multidimensional OLAP)**: físicamente, el almacén de datos se construye sobre estructuras basadas en matrices multidimensionales.

Las ventajas del ROLAP son, en primer lugar, que se pueden utilizar directamente sistemas de gestión de bases de datos genéricos y herramientas asociadas: SQL, restricciones,

⁵ Existen sistemas mixtos, denominados HOLAP (*Hybrid OLAP*).

disparadores, etc. En segundo lugar, la formación y el coste necesario para su implementación es generalmente menor. Las ventajas del MOLAP son su especialización, la correspondencia entre el nivel lógico y el nivel físico. Esto hace que el MOLAP sea generalmente más eficiente, incluso aunque en el caso de ROLAP se utilicen ciertas técnicas de optimización, como comentaremos más abajo.

No todos los sistemas, libros y manuales son consistentes respecto a si la diferencia ROLAP/MOLAP se produce a nivel físico o a nivel lógico. En algunos textos se habla de que si el sistema representa los resultados de los informes/consultas como tablas, el sistema es ROLAP y si los representa como matrices el sistema es MOLAP. Según nuestra definición (y la de muchos otros autores) tanto ROLAP como MOLAP se refieren a la implementación y son independientes de la manera en la que, externamente, se vean las herramientas del sistema de almacenes de datos o el sistema OLAP. Por tanto un sistema puede tener una representación de las consultas relacional y estar basado en un MOLAP o puede tener una representación completamente multidimensional y estar basado en un ROLAP. Algunos ejemplos de sistemas ROLAP son *Microstrategy*, *Informix Metacube* u *Oracle Discoverer*. El primero, por ejemplo, tiene una interfaz completamente multidimensional mientras que por debajo existe un sistema relacional. Ejemplos de sistemas MOLAP son el *Oracle Express* o el *Hyperion Enterprise*.

Como hemos dicho, la ventaja de los ROLAP es que pueden utilizar tecnología y nomenclatura de los sistemas de bases de datos relacionales. Esto tiene el riesgo de que en algunos casos se pueda decidir mantener parte de la base de datos transaccional o inspirarse en su organización (manteniendo claves ajenas, claves primarias, conservando parte de la normalización, etc.). En general, aunque esto pueda ser cómodo inicialmente, no es conveniente a largo plazo. De hecho, una de las maneras más eficientes de implementar un datamart multidimensional mediante bases de datos relacionales se basa en ignorar casi completamente la estructura de los datos en las fuentes de origen y utiliza una estructura nueva denominada *starflake* [Kimball 1996]. Esta estructura combina los esquemas en estrella, *star* y en estrella jerárquica o copo de nieve, *snowflake*.

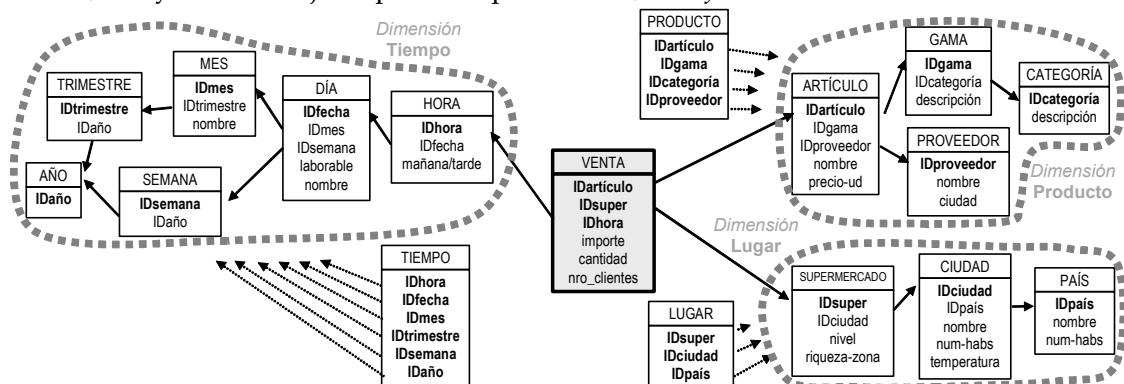


Figura 3.11. Implementación de un datamart utilizando tecnología relacional (ROLAP).

Para construir esta estructura se construyen tres tipos de tablas:

- Tablas copo de nieve (*snowflake tables*): para cada nivel de agregación de una dimensión se crea una tabla. Cada una de estas tablas tiene una clave primaria (señalada en la Figura 3.11 en negrita) y tantas claves ajenas como sean necesarias pa-

ra conectar con los niveles de agregación superiores. En la Figura 3.11, las tablas “mes”, “día”, “artículo”, “ciudad” y “país” (entre otras) son tablas copo de nieve.

- Tabla de hechos (*fact tables*): se crea una única tabla de hechos por datamart. En esta tabla se incluye un atributo para cada dimensión, que será clave ajena (*foreign key*⁶) a cada una de las tablas copo de nieve de mayor detalle de cada dimensión. Además, todos estos atributos forman la clave primaria. Adicionalmente, pueden existir atributos que representen información de cada hecho, denominados generalmente *medidas*. En la Figura 3.11, la tabla “VENTA” es la tabla de hechos.
- Tablas estrella (*star tables*): para cada dimensión se crea una tabla que tiene un atributo para cada nivel de agregación diferente en la dimensión. Cada uno de estos atributos es una clave ajena que hace referencia a tablas copo de nieve. Todos los atributos de la tabla forman la clave primaria (señalados en negrita). En la Figura 3.11, las tablas “TIEMPO”, “PRODUCTO” y “LUGAR” son tablas estrella.

Las tablas estrella son, en realidad, tablas de apoyo, ya que no representan ninguna información que no esté en las demás.

Este diseño proporciona la realización de consultas OLAP de una manera eficiente, así como la aplicación de los operadores específicos:

- Las tablas copo de nieve permiten realizar vistas o informes utilizando diferentes grados de detalle sobre varias dimensiones. Al estar normalizadas permiten seleccionar datos dimensionales de manera no redundante. Esto es especialmente útil para los operadores *drill*, *slice & dice* y *pivot*.
- Las tablas estrella son, como hemos dicho, tablas de apoyo, que representan “pre-concatenaciones” o “pre-junciones” (*pre-joins*) entre las tablas copo de nieve. El propósito de las tablas estrella es evitar concatenaciones costosas cuando se realizan operaciones de *roll-up*.

Además de la estructura anterior, los sistemas ROLAP se pueden acompañar de estructuras especiales: índices de mapa de bits, índices de JOIN, optimizadores de consultas, extensiones de SQL (por ejemplo “CUBE”), etc., así como técnicas tan variadas como el precálculo y almacenamiento de valores agregados que vayan a utilizarse frecuentemente (totales por año, por producto, etc.). Además, se pueden desactivar los *locks* de lectura/escritura concurrente (ya que sólo hay lecturas), muchos índices dinámicos se pueden sustituir por estáticos o por *hashing* (ya que las tablas no van a crecer frecuentemente), etc. Todas estas extensiones y ajustes hacen que el sistema de gestión de bases de datos subyacente se adapte mejor a su nuevo cometido que ya no es una base de datos operacional sino un almacén de datos y proporcione la eficiencia necesaria.

Por el contrario los sistemas MOLAP almacenan físicamente los datos en estructuras multidimensionales de forma que la representación externa e interna coincidan. Las estructuras de datos utilizadas para ello son bastante específicas, lo que permite rendimientos mayores que los ROLAP. En cambio, los sistemas MOLAP tienen algunos inconvenientes:

⁶ También llamada clave/llave externa o foránea o secundaria.

- Se necesitan sistemas específicos. Esto supone un coste de *software* mayor y generalmente compromete la portabilidad, al no existir estándares sobre MOLAP tan extendidos como los estándares del modelo relacional.
- Al existir un gran acoplamiento entre la visión externa y la implementación, los cambios en el diseño del almacén de datos obligan a una reestructuración profunda del esquema físico y viceversa.
- Existe más desnormalización que en las ROLAP. En muchos casos un almacén de datos MOLAP ocupa más espacio que su correspondiente ROLAP.

Tanto para los sistemas ROLAP y MOLAP existen numerosos aspectos que influyen en el diseño físico. Además, existen metodologías y modelos conceptuales para asistir en el diseño conceptual y lógico y, de ahí, al diseño físico. Existen extensiones del modelo entidad-relación (ER) [Sapia et al. 1999; Tryfona et al. 1999] o de modelos orientados a objetos [Trujillo et al. 2001], así como modelos específicos [Golfarelli et al. 1998]. Respecto a la representación, en especial si se utiliza una metodología orientada a objetos o se es familiar con el UML (*Unified Modeling Language*), existe un estándar *Common Warehouse Metadata* (CWM) del OMG (*Object Management Group*, <http://www.omg.org>). Se trata de una extensión del UML para modelar almacenes de datos.

Quizá la parte de diseño de almacenes de datos es una de las áreas más abiertas y donde existe menos convergencia. Las razones son múltiples pero, fundamentalmente, se resumen en que los almacenes de datos se han originado principalmente desde el ámbito industrial y no académico, que el fin inicial del almacén de datos era realizar OLAP eficiente, con lo que el énfasis recaía fundamentalmente en los niveles lógico y físico. A pesar de todo esto, podemos identificar cuatro pasos principales a la hora de diseñar un almacén de datos (en realidad estos pasos se han de seguir para cada datamart):

1. Elegir para modelar un “proceso” o “dominio” de la organización sobre el que se deseen realizar informes complejos frecuentemente, análisis o minería de datos. Por ejemplo, se puede hacer un datamart sobre pedidos, ventas, facturación, etc.
2. Decidir el hecho central y el “gránulo” (nivel de detalle) máximo que se va a necesitar sobre él. Por ejemplo, ¿se necesita información horaria para el tiempo?, ¿se necesita saber las cajas del supermercado o es suficiente el supermercado como unidad mínima?, etc. En general, siempre hay que considerar gránulos finos por si más adelante se fueran a necesitar, a no ser que haya restricciones de tamaño importantes. Precisamente, el almacén de datos se realiza, entre otras cosas, para poder agregar eficientemente, por lo que un almacén de datos demasiado detallado no compromete, en principio, la eficacia.
3. Identificar las dimensiones que caracterizan el “dominio” y su grafo o jerarquía de agregación, así como los atributos básicos de cada nivel. No se deben incluir atributos descriptivos más que lo imprescindible para ayudar en la visualización. En cambio, atributos informativos del estilo “es festivo”, “es fin de semana”, “es estival”, etc., son especialmente interesantes de cara a agregaciones y selecciones que detecten patrones. Las dimensiones varían mucho de un dominio a otro, aunque respondan a preguntas como “qué”, “quién”, “dónde”, “de dónde”, “cuándo”, “cómo”, etc. El tiempo siempre es una (o más de una) de las dimensiones presentes.

4. Determinar y refinar las medidas y atributos necesarios para los hechos y las dimensiones. Generalmente las medidas de los hechos son valores numéricos agregables (totales, cuentas, medias...) y suelen responder a la pregunta “cuánto”. Revisar si toda la información que se requiere sobre los hechos está representada en el almacén de datos.

Existen muchas otras consideraciones que hay que tener en cuenta durante el diseño (véase, por ejemplo [Inmon 2002; Kimball 1996]). Por ejemplo, no hay que obsesionarse por el espacio (algunas normalizaciones no van a mejorar la eficiencia y el ahorro en espacio no es considerable). Tampoco hay que orientarse demasiado en la estructura de la base de datos transaccional. Por ejemplo, no se debe utilizar la misma codificación de claves primarias que en la base de datos transaccional.

3.4 Carga y mantenimiento del almacén de datos

Finalmente, si se ha decidido diseñar un almacén de datos, y ya esté implementado mediante tecnología ROLAP o MOLAP, el siguiente paso es cargar los datos. El proceso tradicional de base de datos más parecido a la carga de un almacén de datos es el proceso de “migración”, aunque a diferencia de él, existe un “mantenimiento” posterior.

En realidad, la carga y mantenimiento de un almacén de datos es uno de los aspectos más delicados y que más esfuerzo requiere (alrededor de la mitad del esfuerzo necesario para implantar un almacén de datos), y, de hecho, suele existir un sistema especializado para realizar estas tareas, denominado **sistema ETL** (*Extraction, Transformation, Load*)⁷. Dicho sistema no se compra en el supermercado ni se descarga de Internet, sino que:

- La construcción del ETL es responsabilidad del equipo de desarrollo del almacén de datos y se realiza específicamente para cada almacén de datos.

Afortunadamente, aunque un ETL se puede construir realizando programas específicos, también se puede realizar adaptando herramientas genéricas (por ejemplo *triggers*), herramientas de migración o utilizando herramientas más específicas que van apareciendo cada vez más frecuentemente.

El sistema ETL se encarga de realizar muchas tareas:

- Lectura de datos transaccionales: se trata generalmente de obtener los datos mediante consultas SQL sobre la base de datos transaccional. Generalmente se intenta que esta lectura sea en horarios de poca carga transaccional (fines de semana o noches). Para la primera carga los datos pueden encontrarse en históricos y es posible que en distintos formatos. Este hecho condiciona muchas veces el número de años que se puede incluir en el almacén de datos.
- Incorporación de datos externos: generalmente aquí se deben incorporar otro tipo de herramientas, como *wrappers*, para convertir texto, hojas de cálculo o HTML en XML o en tablas de base de datos que se puedan integrar en el almacén de datos.

⁷ Existen traducciones diversas en castellano, como ETC (Extracción, Transformación, Carga) o ETT (Extracción, Transformación, Transporte).

- Creación de claves: en general se recomienda crear claves primarias nuevas para todas las tablas que se vayan creando en el almacenamiento intermedio o en el almacén de datos.
- Integración de datos: consiste en muchos casos en la fusión de datos de distintas fuentes, detectar cuándo representan los mismos objetos y generar las referencias y restricciones adecuadas para conectar la información y proporcionar integridad referencial.
- Obtención de agregaciones: si se sabe que cierto nivel de detalle no es necesario en ningún caso, una primera fase de agregación se puede realizar aquí.
- Limpieza y transformación de datos: aunque de estas dos tareas nos dedicaremos en el capítulo siguiente, parte de la limpieza y la transformación necesaria para organizar el almacén se realiza por el ETL. Se trata, como veremos, de evitar datos redundantes, inconsistentes, estandarizar medidas, formatos, fechas, tratar valores nulos, etc.
- Creación y mantenimiento de metadatos: para que todo el ETL pueda funcionar es necesario crear y mantener metadatos sobre el propio proceso ETL y los pasos realizados y por realizar.
- Identificación de cambios: esto se puede realizar de muy distintas maneras: mediante una carga total cada vez que haya un cambio, mediante comparación de instancias (uso de archivos delta), mediante marcas de tiempo (*time stamping*) en los registros, mediante disparadores, mediante el archivo de *log* o mediante técnicas mixtas. Algunas son muy ineficiente (carga total o uso de disparadores) y otras son muy complejas de implementar (archivo de *log*). Generalmente, por tanto, se utilizan técnicas mixtas.
- Planificación de la carga y mantenimiento: consiste en definir las fases de carga, el orden, para evitar violar restricciones de integridad, del mismo modo que se realizan las migraciones, y las ventanas de carga, con el objetivo de poder hacer la carga sin saturar ni la base de datos transaccional, así como el mantenimiento sin paralizar el almacén de datos.
- Indización: finalmente se han de crear índices sobre las claves y atributos del almacén de datos que se consideren relevantes (niveles de dimensiones, tablas de hechos, etc.).
- Pruebas de calidad: en realidad se trata de definir métricas de calidad de datos del almacén de datos, así como implantar un programa de calidad de datos, con un responsable de calidad que realice un seguimiento, especialmente si el almacén de datos se desea utilizar para el apoyo en decisiones estratégicas o especialmente sensibles.

Generalmente, para realizar todas estas tareas, los sistemas ETL se basan en un repositorio de datos intermedio, como se muestra en la Figura 3.12. Esto puede parecer que ya es abusar de recursos, al tener además de la base de datos transaccional y el almacén de datos un tercer repositorio de datos de similar magnitud. Sin embargo, este almacenamiento intermedio es extremadamente útil, ya que hay tareas que no se pueden realizar en el sistema transaccional ni en el almacén de datos. Por ejemplo, la limpieza y transformación de datos se pueden realizar tranquilamente en este repositorio intermedio, ciertos

metadatos pueden almacenarse ahí y valores agregados intermedios también pueden residir ahí, así como los valores integrados de fuentes externas. Con ello, muchos procesos del ETL, incluidos el mantenimiento, se pueden realizar en gran medida sin paralizar ni la base de datos transaccional ni el almacén de datos.

Esta estructura basada en un “almacenamiento intermedio” se muestra en la Figura 3.12 y sitúa más claramente las siglas del acrónimo ETL.

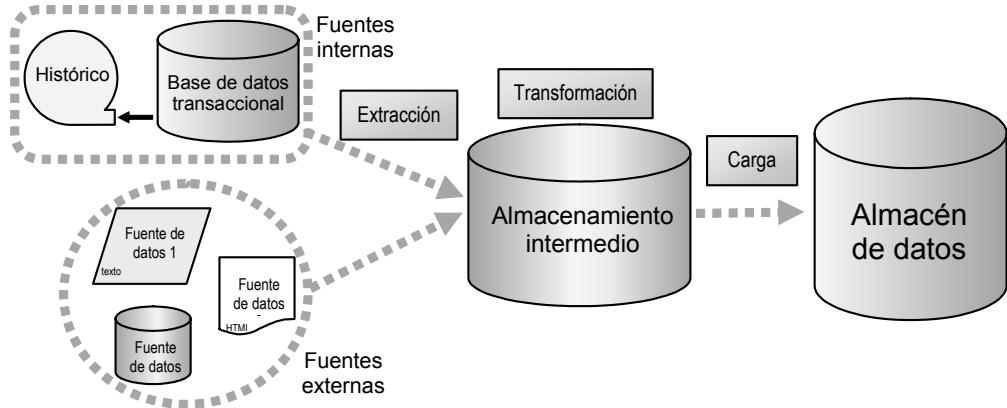


Figura 3.12. El sistema ETL basado en un repositorio intermedio.

La organización con almacenamiento intermedio es especialmente indicada para integrar la información externa. Como dijimos en la introducción, dicha información externa es especialmente importante para encontrar patrones o aspectos significativos en muchos casos, por lo que no nos podemos limitar a la información de la base de datos transaccional. Por ejemplo, la Figura 3.13 muestra que se pueden malinterpretar los datos si no se compara (o se integra) con información externa; una tendencia que parece atisbar una recuperación puede verse como una pérdida de mercado si se compara con la competencia. Del mismo modo, sin esta información externa, puede ser prácticamente imposible extraer patrones.

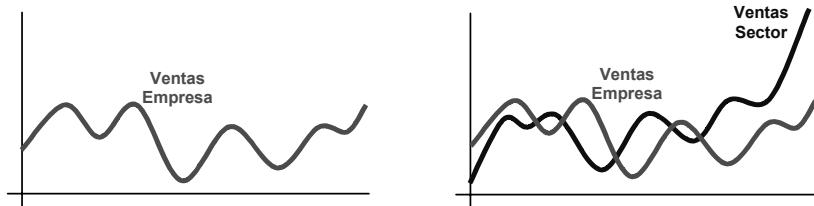


Figura 3.13. La importancia de usar fuentes externas.

En general, existe información que suele ser apropiada para muchos almacenes de datos: demografías (censo), datos resumidos de áreas geográficas, distribución de la competencia, evolución de la economía, información de calendarios y climatológicas, programaciones televisivas-deportivas, catástrofes, páginas amarillas, psicografías sociales, información de otras organizaciones, datos compartidos en una industria o área de negocio, organizaciones y colegios profesionales, catálogos, etc.

El valor de estas fuentes externas ha propiciado la existencia de un mercado de este tipo de base de datos. En aquellos casos en los que no se puede obtener de una manera

gratuita (por ejemplo, desde algún organismo público), algunas bases de datos se pueden comprar a compañías especializadas. Gran parte de estas bases de datos externas no tienen información personal y por tanto no infringen leyes de protección de datos. En el caso de que esto pudiera ser así, es muy importante estar al tanto de la legalidad al respecto (de este tema trataremos en el Capítulo 23).

3.5 Almacenes de datos y minería de datos

El concepto de almacenes de datos nace hace más de una década [Inmon 1992] ligado al concepto de EIS (*Executive Information System*), el sistema de información ejecutivo de una organización. En realidad, cuando están cubiertas las necesidades operacionales de las organizaciones se plantean herramientas informáticas para asistir o cubrir en las necesidades estratégicas.

La definición original de almacén de datos es una “colección de datos, orientada a un dominio, integrada, no volátil y variante en el tiempo para ayudar en las decisiones de dirección” [Inmon 1992; Inmon 2002]. A raíz de esta definición, parecería que los almacenes de datos son sólo útiles en empresas o instituciones donde altos cargos directivos tengan que tomar decisiones. A partir de ahí, y de la difusión cada vez mayor de las herramientas de *business intelligence* y OLAP, podríamos pensar que los almacenes de datos no se aplican en otros ámbitos: científicos, médicos, ingenieriles, académicos, donde no se tratan con las variables y problemáticas típicas de las organizaciones y empresas.

Al contrario, en realidad, los almacenes de datos pueden utilizarse de muy diferentes maneras, y pueden agilizar muchos procesos diferentes de análisis. En la Figura 3.14 se pueden observar las distintas aplicaciones y usos que se puede dar a un almacén de datos: herramientas de consultas e informes, herramientas EIS, herramientas OLAP y herramientas de minería de datos.

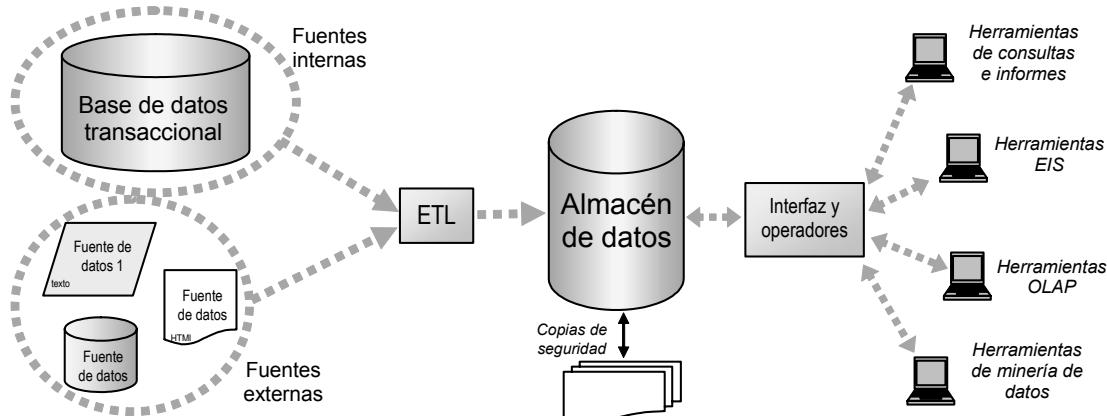


Figura 3.14. Perspectiva general y usos de un almacén de datos.

La variedad de usos que se muestran en la figura anterior sugiere también la existencia de diferentes grupos de usuarios: analistas, ejecutivos, investigadores, etc. Según el carácter de estos usuarios se les puede catalogar en dos grandes grupos:

- “picapedreros” (o “granjeros”): se dedican fundamentalmente a realizar informes periódicos, ver la evolución de indicadores, controlar valores anómalos, etc.

- “exploradores”: encargados de encontrar nuevos patrones significativos utilizando técnicas OLAP o de minería de datos. La estructura del almacén de datos y sus operadores facilita la obtención de diferentes vistas de análisis o “vistas minables”.

Esta diferencia, y el hecho de que se catalogue como “exploradores” a aquellos que utilizan técnicas OLAP o minería de datos, no nos debe hacer confundir las grandes diferencias de un análisis clásico, básicamente basado en la agregación, la visualización y las técnicas descriptivas estadísticas con un uso genuino de minería de datos que no transforma los datos en otros datos (más o menos agregados) sino que transforma los datos en conocimiento (o más humildemente, en reglas o modelos).

Un aspecto a destacar es que el nivel de agregación para los requerimientos de análisis OLAP puede ser mucho más grueso que el necesario para la minería de datos. Por ejemplo, para el análisis OLAP puede ser suficiente usar como unidad mínima de lugar el supermercado. En cambio, para la minería de datos puede ser interesante tener un nivel más fino (por caja o por cajera).

Los almacenes de datos no son imprescindibles para hacer extracción de conocimiento a partir de datos. En realidad, se puede hacer minería de datos sobre un simple archivo de datos. Sin embargo, las ventajas de organizar un almacén de datos se amortizan sobradamente a medio y largo plazo. Esto es especialmente patente cuando nos enfrentamos a grandes volúmenes de datos, o éstos aumentan con el tiempo, o provienen de fuentes heterogéneas o se van a querer combinar de maneras arbitrarias y no predefinidas. Tampoco es cierto que un almacén de datos sólo tenga sentido si tenemos una base de datos transaccional inicial. Incluso si todos los datos originalmente no provienen de bases de datos puede ser conveniente la realización de un almacén de datos.

En gran medida, un almacén de datos también facilita la limpieza y la transformación de datos (en especial para generar “vistas minables” en tiempo real). La limpieza y transformación de datos se tratan precisamente en el capítulo siguiente.

Capítulo 4

LIMPIEZA Y TRANSFORMACIÓN

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

Si bien es cierto que gran parte de los procesos de limpieza e integración se pueden realizar durante la construcción de un “almacén de datos”, como vimos en el capítulo anterior, hemos preferido separar estos dos subprocesos (y describirlos conjuntamente con algunos procesos de transformación) porque, en principio, no es necesario tener un almacén de datos para hacer minería de datos y, además, porque la problemática y herramientas de los procesos de integración, limpieza y transformación están estrechamente relacionadas. En concreto, en este capítulo se relatan una serie de técnicas para la integración y la limpieza (histogramas, detección de valores anómalos, otros tipos de visualización...), algunas transformaciones (discretización, numerización, etc.) y otra serie de técnicas clásicas del análisis multivariante (análisis de componentes principales, análisis de correspondencias, escalado multidimensional...) o análisis factorial.

En el capítulo siguiente se continúa con procesos estrechamente relacionados: la selección (muestreo, selección de atributos) para extraer aquellos datos exclusivamente necesarios para posibilitar una minería eficiente y otros procesos de transformación para presentar los datos de la manera más idónea (sumarización, pivotación, generalización) para las herramientas de minería.

4.1 Introducción

El concepto de “calidad de datos” se asocia cada vez más frecuentemente a los sistemas de información. Aunque se ha avanzado mucho en el diseño y desarrollo de restricciones de integridad de los sistemas de información, éstos han crecido de tal manera en las últimas décadas, que el problema de calidad de datos, en vez de resolverse, en muchos casos se ha acentuado.

En la mayoría de bases de datos existe mucha información que es incorrecta respecto al dominio de la realidad que se desea cubrir y un número menor, pero a veces también relevante, de datos inconsistentes. Estos problemas se acentúan cuando realizamos la integración de distintas fuentes. No obstante, mientras los datos erróneos crecen de manera lineal respecto al tamaño de los datos recopilados, los datos inconsistentes se multiplican; varias fuentes diferentes pueden afirmar cosas distintas sobre el mismo objeto.

La integración también produce una disparidad de formatos, nombres, rangos, etc., que podría no existir, o en menor medida, en las fuentes originales. Esto dificulta en gran medida los procesos de análisis y extracción de conocimiento. Para resolver esta disparidad se presentan una serie de consejos para la integración, para la limpieza y algunas transformaciones para convertir los datos en otros más apropiados para la minería.

Estos procesos (o algunos de ellos, o junto a otros, como la selección de datos) reciben nombres bastante variados: preparación de datos, *data cooking*, preprocessamiento, etc. Conjuntamente, la preparación de datos tiene como objetivo la eliminación del mayor número posible de datos erróneos o inconsistentes (limpieza) e irrelevantes (criba), y trata de presentar los datos de la manera más apropiada para la minería de datos. Del proceso de selección, y de otras técnicas de transformación, también se hablará en el capítulo siguiente, como fase final (más exploratoria y de menos preprocessamiento de datos) antes de la minería de datos propiamente dicha.

Al igual que en el capítulo siguiente, en estas fases se utilizan técnicas estadísticas, de visualización y de consulta (si es posible, OLAP). Algunas de ellas son de uso bastante común y se encuentran no sólo en paquetes estadísticos, sino en hojas de cálculo y en muchas herramientas de minería de datos, como por ejemplo, los histogramas para la detección de datos anómalos, gráficos de dispersión, cálculos de medias, varianzas, correlaciones, etc.

El análisis multivariante (*multivariate analysis*) clásico engloba, tradicionalmente, una serie de técnicas estadísticas para tratar con un conjunto de variables, que tienen sus orígenes en el álgebra lineal y la geometría, y que, por esta razón, pese a su diversidad, se agrupan bajo el mismo término. Dentro del análisis multivariante se incluye generalmente el análisis de componentes principales (*principal components analysis*), el análisis de correspondencias (*correspondence analysis*), el escalado multidimensional (*multidimensional scaling*) y ciertas técnicas de análisis de grupos (*cluster analysis*) y descomposición de mezclas o mixturas (*mixture decomposition*). Muchas de ellas sirven especialmente para la preparación de datos y, por ello, se verán en este capítulo. Exceptuamos el análisis de grupos que se ve en varios capítulos y la descomposición de mezclas que se comenta muy brevemente. De modo similar, ciertas técnicas muy relacionadas, dentro de lo que se conoce como análisis factorial, serán comentadas muy brevemente.

Como hemos dicho, muchas técnicas descritas en este capítulo (y la mayoría de las descritas en los capítulos 7 y 8) se pueden encontrar en paquetes estadísticos clásicos. Cada día más frecuentemente también, los paquetes de minería de datos incluyen además algunas de estas herramientas.

Otras técnicas son más artesanales y más variadas. Por ejemplo, la redefinición de atributos (mediante creación de nuevos atributos o mediante la separación) o incluso la discretización o numerización.

El orden en el que se presentan las técnicas en este capítulo (y en el siguiente) no implica necesariamente que se deban realizar en este orden. Por ejemplo, un muestreo puede ser previo a una discretización o viceversa. Se puede hacer un aumento de dimensionalidad (mediante introducción de relaciones cuadráticas) seguido de una reducción de dimensionalidad mediante análisis de componentes principales. En general, el orden dependerá mucho del problema y de las características de los datos. En muchos casos algunas técnicas se pueden repetir varias veces, intercalándose con otras.

Un aspecto muy importante a la hora de realizar los procesos de integración, limpieza, selección y transformación es que se debe conocer el dominio de donde provienen los datos. Por ejemplo, un histograma puede ayudar a detectar los datos anómalos más flagrantes pero no podrá ayudarnos para determinar otros casos que sólo pueden detectarse con seguridad si conocemos el dominio de los datos. En otros casos, conocer el dominio es imprescindible, como por ejemplo para la redefinición de atributos (mediante creación o separación).

En este capítulo y en el siguiente trataremos de datos de distinto tipo, aunque, en general, para los propósitos de limpieza, transformación, selección y minería de datos, podemos clasificarlos en tres tipos: **numéricos** (sean enteros o reales, abiertos o cerrados por un intervalo, circulares), **nominales sin orden** (incluyendo valores lógicos o booleanos, con valores prefijados o abiertos) y **nominales con orden** u ordinales (del estilo `{ bajo, medio, alto }`). Existen numerosas nomenclaturas alternativas, como pueden ser continuos y discretos, cuantitativos y cualitativos/categóricos, que no son exactamente equivalentes a la terminología de datos numéricos y nominales, que es la que utilizaremos, en general, a continuación. Sobre la terminología para nombrar los atributos (variables, componentes, características, campos, etc.) seremos más flexibles y utilizaremos muchos de estos nombres indistintamente.

4.2 Integración y limpieza de datos

Como hemos dicho, existen problemas de calidad de datos en los sistemas de información. Estos problemas, además, pueden verse agravados por el proceso de integración de distintas fuentes, especialmente si no se hace con esmero. Por ejemplo, existen datos faltantes que suelen ser originados muchas veces al integrar fuentes diferentes, para los cuales no existen soluciones fáciles, pero hay otros casos, como los valores duplicados, que sí pueden y deben ser detectados durante la integración.

La integración es generalmente un proceso que se realiza durante la recopilación de datos y, si se realiza un almacén de datos, durante el proceso de carga, mediante el sistema ETL visto en el capítulo anterior. La limpieza de datos (*data cleaning / cleansing*) puede, en muchos casos, detectar y solucionar problemas de datos no resueltos durante la integración,

como los valores anómalos y faltantes. Por tanto, parece lógico que la limpieza tenga lugar durante esta integración o inmediatamente después de ella.

Lógicamente, estos procesos de integración y limpieza pueden ser más rudimentarios cuando no se desee crear un almacén de datos. En cualquier caso, es un aspecto que no hay que descuidar, particularmente si se desea realizar minería de datos de una manera sistemática.

4.2.1 Integración

El primer problema a la hora de realizar una integración de distintas fuentes de datos es identificar los objetos, es decir, conseguir que datos sobre el mismo objeto se unifiquen y datos de diferentes objetos permanezcan separados. Este problema se conoce como el problema del esclarecimiento de la identidad. Existen dos tipos de errores que pueden ocurrir en esta integración:

- Dos o más objetos diferentes se unifican: los datos resultantes mezclarán patrones de diferentes individuos y serán un problema para extraer conocimiento. Esto será más grave cuanto más diferentes sean los dos objetos unificados.
- Dos o más fuentes de objetos iguales se dejan separadas: los patrones del mismo individuo aparecerán repartidos entre varios individuos parciales. Este problema genera menos “ruido” que el anterior, aunque es especialmente problemático cuando se usan valores agregados (el total de compras será mucho menor si consideramos un individuo real como dos individuos en la base de datos, por ejemplo).

En general el primer problema es menos frecuente que el segundo, ya que la unificación se realiza generalmente por identificadores *externos* a la base de datos: número de identidad, número de pólizas, matrículas, tarjeta de crédito o de fidelización, etc. Además, se suele ser conservador a la hora de unificar; si no se está seguro no se hace. Esta tarea es más difícil de lo que pueda parecer, ya que si se utilizan claves internas para identificar objetos (por ejemplo autonumeradas), hay que mirar los identificadores externos y éstos, muchas veces, varían de formato (por ejemplo, un ciudadano español puede identificarse por el DNI, el NIF y el pasaporte, que coinciden en ocho dígitos, pero el NIF añade una letra y el pasaporte añade alguna letra y dígito adicional). Este proceso de identificación se muestra en la parte superior izquierda de la Figura 4.1.

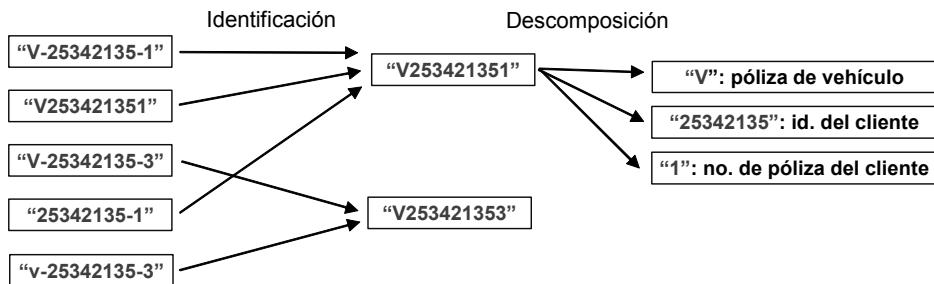


Figura 4.1. Ejemplos de integración: identificación y descomposición.

Las claves internas de sistemas mal diseñados pueden entrañar información no normalizada, que es preciso detectar en el proceso de integración. Este proceso se denomina *descomposición de claves*. La parte derecha de la Figura 4.1 muestra un ejemplo.

Cuando se integran (correctamente) dos fuentes diferentes de datos de distintos objetos suele suceder que puedan aparecer datos faltantes (el dato se registra en una fuente pero no en la otra) o datos inconsistentes (el dato es diferente en una fuente y otra). Esto se puede observar en la Figura 4.2.

Figura 4.2. Ejemplos de integración de atributos de distintas fuentes.

Fuente 1:

DNI	EDAD	COD.POSTAL	ESTADO	AÑOS_CARNÉ
...
25342135	35	46019	Casado	13
98525925	23	28004	Soltero	1
...

Fuente 2:

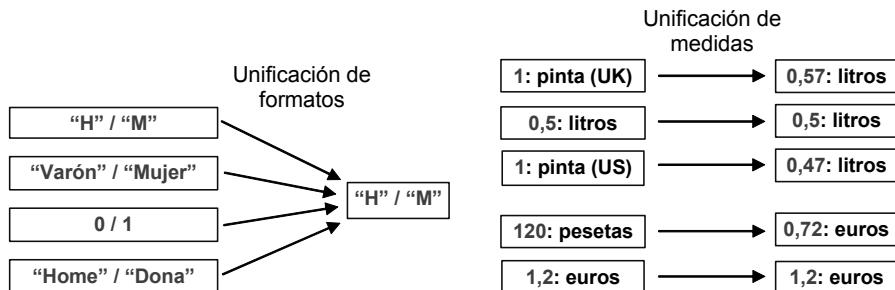
DNI	FECHA_NAC	CIUDAD	CASADO	CARNÉ
...
77775252	1/1/1950	Benitatxell	Sí	A2
25342135	18/11/1971	Valencia	No	B1
...

Integración:

DNI	EDAD	FECHA_NAC	CIUDAD	COD_POSTAL	ESTADO	CASADO	AÑOS_CARNÉ	CARNÉ
...
25342135	35	18/11/1971	Valencia	46019	Casado	No	13	B1
98525925	23	-	-	28004	Soltero	-	1	-
77775252	-	1/1/1950	Benitatxell	-	-	Sí	-	A2
...

Lógicamente aparecen campos redundantes total o parcialmente: "edad" y "fecha_nac", "ciudad" y "cod_postal", "estado" y "casado". Cuando sea posible se intentarán fusionar. En muchos casos los datos inconsistentes se convierten en faltantes; por ejemplo si el mismo cliente tiene estados civiles diferentes en cada fuente, es preferible dejar el valor a faltante que elegir al azar uno de los dos (o hacer la media).

Otro caso muy frecuente es la integración de formatos diferentes, que se produce si tenemos codificaciones diferentes (casado / matrimonio), idiomas diferentes, medidas diferentes, etc.



En general existen muchas otras situaciones, algunas de ellas relativamente típicas en procesos de migración y fusión de bases de datos, pero hay otras tantas que son específicas del dominio y de las bases de datos integradas y que pueden necesitar soluciones muy específicas.

4.2.2 Reconocimiento

Cuando tenemos integrados todos los datos lo primero que podemos realizar es un resumen de las características (o informe de estado) de atributos (ya sea tabla a tabla o para toda la base o almacén de datos). En este tipo de tablas se muestran las características

generales de los atributos (medias, mínimos, máximos, posibles valores). Se puede distinguir entre valores nominales y numéricos (y hacer dos tablas) o integrarlo todo en la misma tabla.

Por ejemplo, para una compañía de seguros, tenemos los datos referidos a las pólizas de vehículos. La siguiente tabla muestra (parcialmente) un resumen de los atributos de la base de datos:

Atributo	Tabla	Tipo	# total	# nulos	# dists	Media	Desv.e.	Moda	Min	Max
Código postal	Cliente	Nominal	10320	150	1672	-	-	"46003"	"01001"	"50312"
Sexo	Cliente	Nominal	10320	23	6	-	-	"V"	"E"	"M"
Estado civil	Cliente	Nominal	10320	317	8	-	-	Casado	"Casado"	"Viudo"
Edad	Cliente	Numérico	10320	4	66	42,3	12,5	37	18	87
Total póliza p/a	Póliza	Numérico	17523	1325	142	737,24€	327€	680€	375€	6200€
Asegurados	Póliza	Numérico	17523	0	7	1,31	0,25	1	0	10
Matrícula	Vehículo	Nominal	16324	0	16324	-	-	"A-0003-BF"	"Z-9835-AF"	
Modelo	Vehículo	Nominal	16324	1321	2429	-	-	"O. Astra"	"Audi A3"	"VW Polo"
...

Tabla 4.1. Tabla resumen de atributos.

La tabla anterior es sencilla de construir (se puede hacer incluso a partir de un conjunto de consultas SQL) y da mucha información de un simple vistazo. Además de ver cuántos clientes, pólizas y vehículos tenemos, podemos observar el total de nulos de cada atributo, después el número de valores distintos (incluyendo los nulos), la media y la desviación típica para los valores numéricos, la moda (el valor más frecuente), el mínimo y el máximo (para los valores nominales, el mínimo y el máximo se interpretan como en SQL, alfabéticamente).

Hay aspectos respecto a la calidad de los datos que saltan a la vista; por ejemplo, ¿cómo es posible que haya cinco valores (más el nulo) diferentes para el atributo "sexo"? Para observar los valores podemos utilizar una de las herramientas gráficas más básicas, el *histograma*, que muestra la distribución de los valores posibles para el atributo "sexo" (véase Figura 4.4):

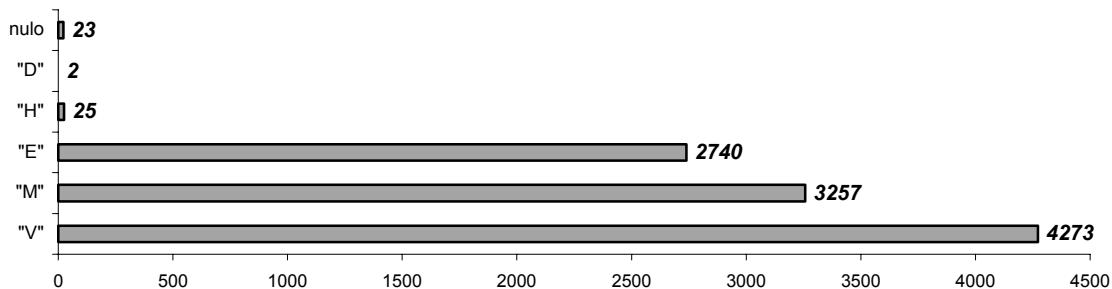


Figura 4.4. Histograma representando las frecuencias de un atributo nominal.

Al observar los datos, podemos darnos cuenta de que en realidad el nombre "sexo" para el atributo no está muy bien elegido, porque el valor "E" viene a representar que no es una persona sino una empresa la que asegura el vehículo. A partir de esa aclaración, los datos "V", "M" y "E" parecen claros: "varón", "mujer" y "empresa". El problema lo presentan los valores "H" y "D". Tras un análisis de la procedencia de los datos, la mayoría de "H" se

supone que vienen de “hombre” pero algunos pueden venir, equivocadamente, del término erróneo “hembra”, especialmente en las fichas antiguas, que se realizaban por papel y no a través de una aplicación informática con mayores restricciones de integridad. Debido a esta ambigüedad, todos los valores “H” se dejan como valores nulos (luego se verá que muchos de ellos se podrán llenar, mirando los nombres de los clientes). Finalmente, el valor “D” puede deberse a que las aplicaciones de la empresa son bilingües (castellano / catalán) y “dona” es “mujer” en catalán, con lo que se decide unificar con “M”. Éste es un ejemplo muy simple de los aspectos que son necesarios para, en este caso, realizar una unificación de formatos: reconocer los datos y su distribución y, sobre todo, conocer las fuentes de los datos y el dominio (en este caso una empresa aseguradora).

Siguiendo con la tabla de resumen de atributos, la información sobre media, desviación típica, moda, máximo y mínimo proporciona bastante información sobre los atributos numéricos. En general, un segundo paso sobre estos valores es también un histograma (en el caso de los valores numéricos se realiza por intervalos, generalmente). Por ejemplo, observemos en la Figura 4.5 un histograma para los valores del atributo “total póliza por año” (los nulos no se muestran en el histograma):

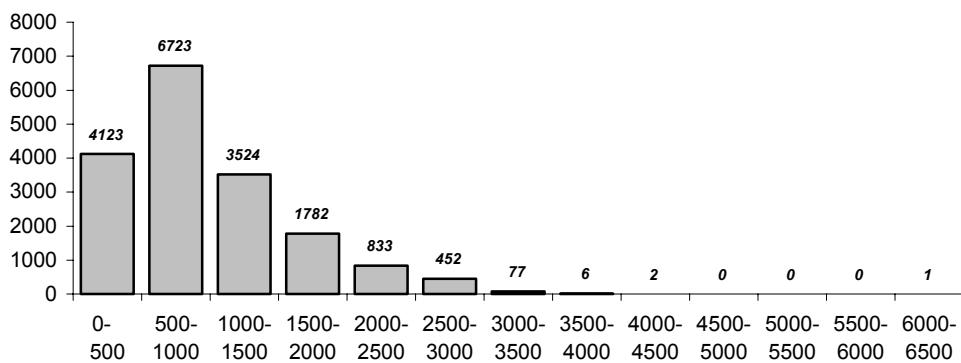


Figura 4.5. Histograma representando las frecuencias de un atributo numérico.

Los datos se distribuyen de una manera cercana a la normal y se confirma el pico alrededor de la media (737,24). La moda es menor (680), lo que coincide con la tendencia lateral (*skewness*) de la distribución (la cola de la izquierda se para por el mínimo, que está en 375). Quizá lo más significativo es que existe un dato entre 6.000 y 6.500 que podría considerarse anómalo. Sin conocimiento del dominio y sin observar el dato en detalle, no podemos determinar si dicho dato es simplemente un dato anómalo pero correcto (el seguro de un automóvil muy especial, como un Roll Royce) o un dato erróneo.

No vamos a entrar aquí en detalles de las herramientas existentes para analizar las distribuciones de frecuencias. Muchos sistemas calculan la distribución que se ajusta más a los datos (normal, uniforme...) y nos la dibujan sobreescrita en el histograma. Esto generalmente ayuda a entender mejor la distribución de los datos y permite determinar, si bajo esa distribución, los datos extremos se consideran estadísticamente anómalos. Volveremos más adelante sobre los datos anómalos.

En muchos casos en los histogramas pueden aparecer combinaciones de distribuciones simples. Esto a veces se observa viendo que los datos no son unimodales (una moda) y son en realidad bimodales (dos “jorobas” o dos modas), trimodal (tres modas), etc.

Una alternativa a los histogramas, a la hora de estudiar las frecuencias de una variable, son los **diagramas de caja** (*box plots*) o de bigotes (*whisker plots*).

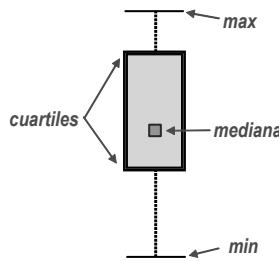


Figura 4.6. Diagrama de caja (o de bigotes).

La caja va del primer cuartil al tercer cuartil (del 25 por ciento de los datos al 75 por ciento de los datos). Es decir, la caja muestra el rango intercuartil, que contiene el 50 por ciento de los datos. La mediana se representa con un cuadrito (también se puede hacer con otro tipo de marcas) y muestra el segundo cuartil (el valor tal que la mitad de los datos estén por debajo y la mitad por encima). Los bigotes (o líneas acabadas en un segmento) muestran el resto de los datos hasta los valores más extremos. En cierto modo, los diagramas de caja son un resumen de los histogramas y permiten, en un mismo gráfico, representar varias variables.

Los histogramas se pueden extender para las frecuencias conjuntas de dos variables, dando un histograma tridimensional. Se puede realizar entre dos variables nominales, una nominal y una numérica o entre dos numéricas (las numéricas siempre agrupadas entre intervalos). En la Figura 4.7 se muestra un ejemplo de histograma de dos variables que muestra las frecuencias conjuntas del “sexo” respecto al total de la póliza de cada cliente.

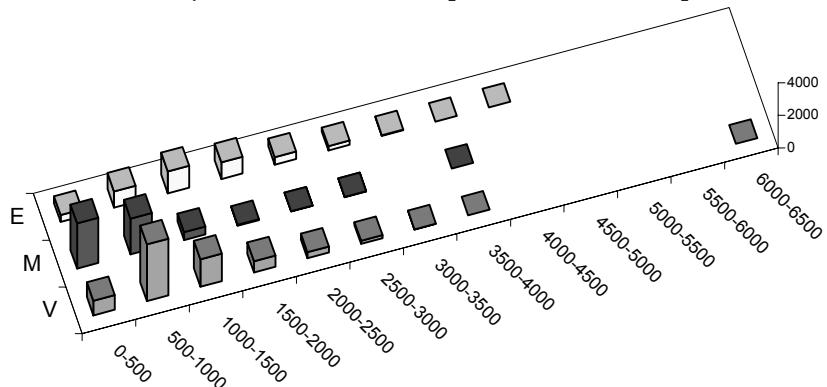


Figura 4.7. Histograma representando las frecuencias de un atributo numérico vs. un atributo nominal.

En la figura se observan tres distribuciones diferentes para los tres valores “E”, “M” y “V”, con jorobas diferentes dependiendo del tipo de cliente.

Una gráfica similar y especialmente útil cuando los dos atributos son numéricos es la **gráfica de dispersión** (*scatterplot*). En la Figura 4.8 se muestran dos variables numéricas (la edad en el eje de las X y los accidentes por quinquenio en el eje de las Y, de los clientes de la aseguradora en el municipio de Denia).

Esta gráfica no tiene el “sesgo” de los intervalos numéricos como los histogramas. El problema es que si existen muchos individuos (como ocurre, por ejemplo en la parte

central), éstos se “apiñan” y llega un punto en el que no se puede ver si hay más o menos densidad.

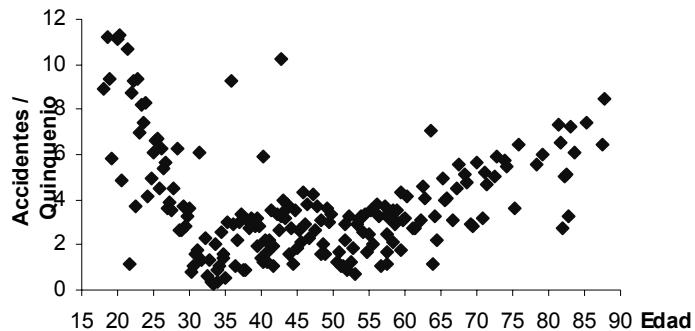


Figura 4.8. Gráfica de dispersión (diagrama bivariante).

En las gráficas de dispersión se puede mostrar una tercera dimensión. Si marcamos los puntos con distintos símbolos, se puede representar un tercer atributo nominal (por ejemplo la clase). Esta gráfica se llama “gráfica de dispersión etiquetada” y se muestra en la Figura 4.9.

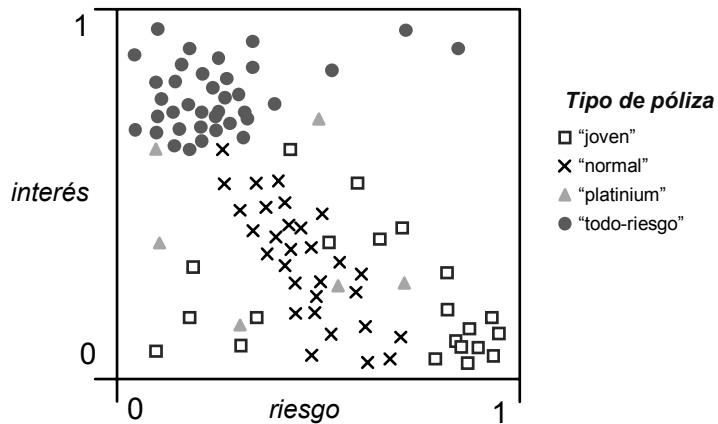


Figura 4.9. Gráfica de dispersión etiquetada.

La figura anterior, en concreto, representa dos variables numéricas: “riesgo”, que representa el riesgo del asegurado (de 0 a 1, en función de los años de carné, la juventud, los accidentes previos, etc.) y el “interés”, que representa la importancia del cliente para la compañía (de 0 a 1, en función de otras pólizas realizadas, familiares asegurados, etc.). El gráfico muestra que estas dos variables son muy significativas, especialmente para las pólizas “joven”, “normal” y “todo-riesgo”.

Cuando tenemos más de dos variables el gráfico anterior se puede repetir para todas las combinaciones posibles. Por ejemplo, la Figura 4.10 muestra todas las gráficas de dispersión etiquetadas, resultantes de las combinaciones de los cuatro atributos numéricos (*sepallength*, *sepawidth*, *petallength*, *petalwidth*) del conjunto de datos de lirios (“iris”, véase el Apéndice B). La clase o tipo de lirio (*setosa*, *versicolour*, *virginica*) se muestra por el tono de los círculos (más claros u oscuros).

Como se puede observar hay pares de atributos (por ejemplo *petallength* y *petalwidth*) en los que las tres clases aparecen más diferenciadas, y combinaciones de atributos donde esa diferenciación sería más difícil (por ejemplo *sepalwidth* y *sepallength*). De hecho, se puede observar que sólo con el valor de *petallength*, se podría conseguir una buena clasificación. Esto ayuda a ver patrones y a determinar qué variables pueden ser más relevantes para el problema. Más adelante en este capítulo (y en capítulos siguientes) trataremos técnicas para estimar la relevancia de variables.

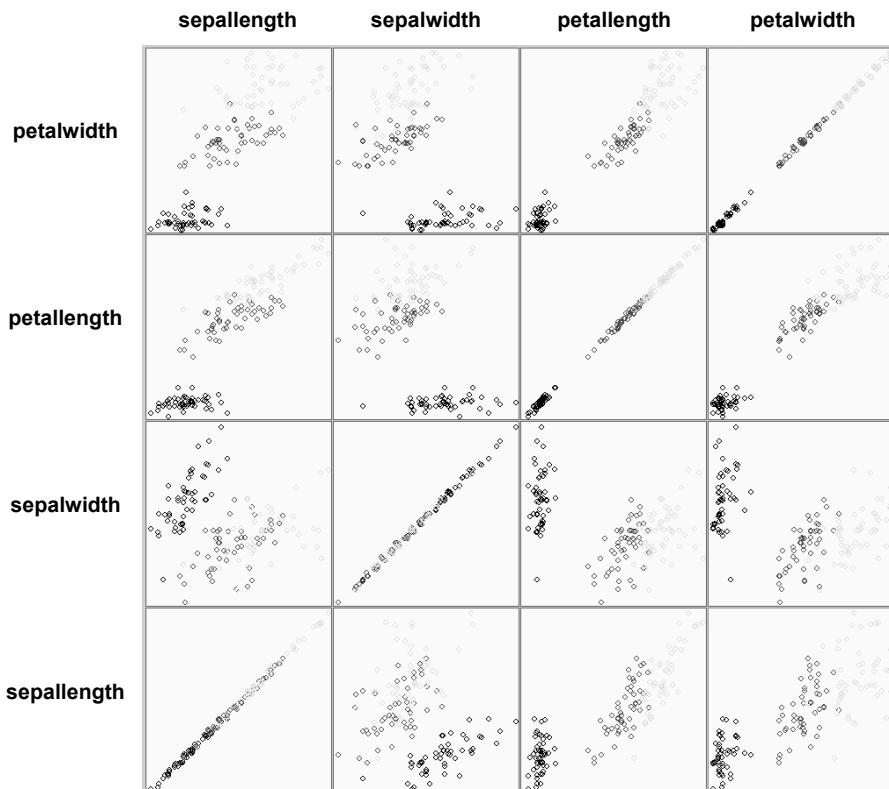


Figura 4.10. Matriz de gráficas de dispersión etiquetadas (plot matrix).

4.2.3 Valores faltantes

Los valores faltantes, perdidos o ausentes (*missing values*) pueden ser reemplazados por varias razones. En primer lugar, el método de minería de datos que utilicemos puede no tratar bien los campos faltantes. En segundo lugar, podemos querer agregar los datos (especialmente los numéricos) para realizar otras vistas minables y que los valores faltantes no nos permitan agregar correctamente (totales, medias, etc.). En tercer lugar, si el método es capaz de tratar campos faltantes es posible que ignore todo el ejemplo (produciendo un sesgo) o es posible que tenga un método de sustitución de campos faltantes que no sea adecuado debido a que no conoce el contexto asociado al atributo faltante.

A la hora de hablar de campos faltantes, debemos tratar de su detección y de su tratamiento. La detección de campos faltantes puede parecer sencilla. Si los datos proceden de una base de datos, basta mirar en la tabla de resumen de atributos/características y ver

la cantidad de nulos que tiene cada atributo. El problema es que a veces los campos faltantes no están representados como nulos. Por ejemplo, aunque hay campos en los que las restricciones de integridad del sistema evitan introducir códigos fuera del formato para representar valores faltantes, esto al final ocurre en muchos otros, especialmente en campos sin formato: direcciones o teléfono como “no tiene”, códigos postales o números de tarjeta de crédito con valor -1, etc. A veces son las propias restricciones de integridad las que causan el problema. Por ejemplo, ¿cuántos sistemas obligan a introducir necesariamente los dos apellidos y fuerzan que a los extranjeros, por tanto, les pongamos algún valor (“-”) en el segundo apellido? Este tipo de situaciones complica sobremanera la detección de valores faltantes. No obstante, son precisamente aquellos casos más difíciles en los que merece la pena realizar más esfuerzo, ya que muchas veces son este tipo de “nulos camuflados” los que pueden introducir sesgo en el conocimiento extraído.

Tanto para la detección, como para su tratamiento posterior, es importante saber el porqué de los valores faltantes.

- Algunos valores faltantes expresan características relevantes. Por ejemplo, la falta de teléfono puede representar en muchos casos un deseo de que no se moleste a la persona en cuestión, o un cambio de domicilio reciente.
- Valores no existentes. Muchos valores faltantes existen en la realidad, pero otros no. Por ejemplo el cliente que se acaba de dar de alta no tiene un registro de accidentes medio de los últimos años.
- Datos incompletos: si los datos vienen de fuentes diferentes, al combinarlos se suele hacer la unión y no la intersección de campos, con lo que muchos datos faltantes representan que esas tuplas vienen de una/s fuente/s diferente/s al resto.

Finalmente, si se han conseguido establecer los datos faltantes e, idealmente, sus causas, procederemos a su tratamiento. Las posibles acciones sobre datos faltantes son:

- Ignorar (dejar pasar): algunos algoritmos son robustos a datos faltantes (por ejemplo árboles de decisión).
- Eliminar (filtrar o reemplazar) toda la columna (es decir quitar el atributo para todos los ejemplos): solución extrema, pero a veces la proporción de nulos es tan alta que la columna no tiene arreglo. Otras veces, existe otra columna dependiente con datos de mayor calidad.
- Filtrar la fila: claramente sesga los datos, porque muchas veces las causas de un dato faltante están relacionadas con casos o tipos especiales.
- Reemplazar el valor: se puede intentar reemplazar el valor manualmente (en el caso de que no haya muchos) o automáticamente por un valor que preserve la media o la varianza (globales o por clases/grupos), en el caso de valores numéricos, o por el valor moda, en el caso de valores nominales. Una manera más sofisticada de estimar un valor es *predecirlo* a partir de los otros ejemplos (esto se llama a veces “imputación de datos perdidos”, utilizando cualquier técnica predictiva de aprendizaje automático (clasificación o regresión) o técnicas más específicas (por ejemplo, determinar el sexo a partir del nombre). También existen algoritmos que se usan tradicionalmente para esto, como el algoritmo EM, que veremos precisamente para esta aplicación en el Capítulo 10 (Sección 10.7).

- Segmentar: se segmentan las tuplas por los valores que tienen disponibles. Se obtienen modelos diferentes para cada segmento y luego se combinan.
- Modificar la política de calidad de datos y esperar hasta que los datos faltantes estén disponibles.

Quizás una de las soluciones anteriores más frecuentes cuando el algoritmo a utilizar no maneja bien los nulos sea reemplazar el valor. Si sustituimos un dato faltante por un dato estimado, hemos de tener en cuenta que, en primer lugar, perdemos información, ya que ya no se sabe que el dato era faltante y, en segundo lugar, inventamos información, con los riesgos que pueda tener de que sea errónea. El primer problema también ocurre en el caso de que eliminemos toda la columna.

La solución a ambos problemas pasa por crear un nuevo atributo lógico (booleano) indicando si el atributo original era nulo o no. Esto permite al proceso y al método de minería de datos, si son bastante perspicaces, saber que el dato era faltante y, por tanto, que el valor hay que tomarlo con cautela. En el caso en el que el atributo original sea nominal no es necesario crear un nuevo atributo, basta con añadir un valor adicional, denominado “faltante”. Esta es una solución (sin ninguna acción adicional) bastante frecuente con atributos nominales.

Existen casos excepcionales, por ejemplo, cuando tratamos con datos poco densos en el sentido de que casi todos los atributos tienen unos niveles muy altos de valores faltantes (por encima del 50 por ciento, por ejemplo). Reemplazar o filtrar no parece ser la solución. En estos casos existen soluciones particulares, por ejemplo intentar aglutinar varios ejemplos *similares* en uno solo y describir porcentajes de valores faltantes, valores medios, desviaciones, etc.

4.2.4 Valores erróneos. Detección de valores anómalos

Del mismo modo que para los campos faltantes, para los campos erróneos o inválidos, hemos de distinguir entre la detección y el tratamiento de los mismos. La detección de campos erróneos se puede realizar de maneras muy diversas, dependiendo del formato y origen del campo. En el caso de datos nominales, la detección dependerá fundamentalmente de conocer el formato o los posibles valores del campo. Por ejemplo, si estamos tratando con un atributo con un formato fijo, como la matrícula de un vehículo, podemos establecer si una determinada matrícula es errónea si no se ajusta al formato o los formatos permitidos en un cierto país. Parece evidente que aquellos datos erróneos que sí se ajusten al formato serán mucho más difíciles (o imposibles) de detectar.

Resulta especialmente patente el hecho de que si los sistemas de información originales fueron diseñados con buenas restricciones de integridad (no permitiendo, por ejemplo, matrículas con un formato no correcto), todos estos campos erróneos no habrán podido introducirse. Este caso, obviamente, es mucho más aconsejable. En general, en estos casos, la detección de campos erróneos por el formato no es posible, ya que todos los datos se ajustan a los datos correctos, y nos podremos concentrar en otros tipos de errores.

En algunos casos, afortunadamente, es posible detectar campos erróneos por su contenido. Siguiendo con el ejemplo de las matrículas, podemos comparar (preferiblemente de una manera automática o semiautomática) los modelos de vehículos con las matrículas. Si las matrículas se numeran incrementalmente (“BFG 8940” después de “BFG 8939”),

podemos observar si alguna matrícula supuestamente antigua está asociada a un modelo de vehículo muy moderno. Esta situación puede denotar un posible error.

También mediante la tabla de resumen de atributos o histogramas particular, podemos detectar valores erróneos. Por ejemplo, así se ha actuado anteriormente con un atributo nominal, en el ejemplo del sexo (véase Figura 4.4).

En el caso de la detección de valores erróneos en atributos numéricos, ésta suele empezar por buscar valores anómalos, atípicos o extremos (*outliers*), también llamados datos aislados, exteriores o periféricos. Es importante destacar que un valor erróneo y un valor anómalo no son lo mismo. Como hemos dicho, existen casos en los que los valores extremos se categorizan como anómalos estadísticamente pero son correctos, es decir, representan un dato fidedigno de la realidad. No obstante, así y todo, pueden ser un inconveniente para algunos métodos que se basan en el ajuste de pesos ejemplo por ejemplo, como las redes neuronales. De modo similar, y mucho más frecuentemente, puede haber datos erróneos que caen en la “normalidad” y, por tanto, no pueden ser detectados.

En el ejemplo anterior de la Figura 4.5 vemos que existe un valor por encima de los 6.000 euros, bastante lejano o aislado de la parte central de la “campana”. Dicho valor puede ser determinado como un valor anómalo desde el punto de vista estadístico. En general, existen herramientas estadísticas (basadas en tests de discordancia, véase por ejemplo [Barnett & Lewis 1994]) que sugieren los valores anómalos (bajo la distribución asumida o sugerida) y es el usuario el que finalmente debe determinar si son erróneos o no.

Otras técnicas para la detección de valores anómalos consisten en definir una distancia (incluyendo valores nominales y numéricos) y ver los individuos con mayor distancia media al resto de individuos (individuo cuyo vecino más próximo o cuyos k -vecinos más próximos está más lejos) o bien donde una fracción de los otros individuos está lejos. Ésta es una manera más general que los histogramas y se ha desarrollado en gran medida últimamente [Knorr & Ng 1998].

Esto se puede llevar todavía mucho más allá. En primer lugar, podemos utilizar técnicas de *clustering* parciales (no todos los elementos caen dentro de un grupo). Aquellos elementos aislados que no entran en un grupo o aquellos grupos minoritarios pueden ser considerados valores extremos. También se puede realizar una detección por predicción. Es decir, se realiza un modelo de clasificación o regresión (dependiendo de si el atributo es nominal o numérico) y se predice el valor del atributo para cada ejemplo. Aquellos ejemplos cuya predicción esté más lejos del valor que existe en la instancia (se puede utilizar un clasificador suave, en el caso de valores nominales) puede denotar un error a ser inspeccionado. Para profundizar en la gran cantidad de métodos para detectar valores anómalos se puede consultar [Peña & Prieto 2001; Peña 2002b; Barnett & Lewis 1994; Rousseeuw & Leroy 2003].

Antes de pasar al tratamiento de valores anómalos es importante destacar que no detectar un valor anómalo puede ser un problema importante si el atributo se normaliza posteriormente (entre cero y uno por ejemplo), ya que la mayoría de datos estarán en un rango muy pequeño (entre 0 y 0,1 por ejemplo) y puede haber poca precisión o sensibilidad para algunos métodos de minería de datos. Veremos, más adelante en este mismo capítulo, tipos especiales de normalización o escalado (por ejemplo el *softmax*) que intentan distribuir más uniformemente los datos para que los datos extremos sean menos extremos.

Finalmente, llegamos al tratamiento de valores anómalos o erróneos (los que decidamos tratar de ambos casos, visto que anómalo no indica siempre un error). Los tratamientos son muy similares a los de los datos faltantes:

- Ignorar (dejar pasar): algunos algoritmos son robustos a datos anómalos (por ejemplo árboles de decisión).
- Filtrar (eliminar o reemplazar) la columna: solución extrema, pero a veces existe otra columna dependiente con datos de mayor calidad. Preferible a eliminar la columna es reemplazarla por una columna discreta diciendo si el valor era normal u erróneo (por encima o por debajo). En el caso de los anómalos se puede sustituir por no anómalo, anómalo superior o anómalo inferior.
- Filtrar la fila: puede sesgar los datos, porque muchas veces las causas de un dato erróneo están relacionadas con casos o tipos especiales.
- Reemplazar el valor: por el valor ‘nulo’ si el algoritmo lo trata bien o por máximos o mínimos, dependiendo por dónde es el anómalo, o por medias. A veces se puede predecir a partir de otros datos, utilizando cualquier técnica de ML.
- Discretizar: transformar un valor continuo en uno discreto (por ejemplo muy alto, alto, medio, bajo, muy bajo) hace que los anómalos caigan en ‘muy alto’ o ‘muy bajo’ sin mayores problemas.

Los atributos erróneos son mucho más graves cuando afectan a un atributo que va a ser utilizado con clase o como valor de salida de una tarea predictiva de minería de datos. De estos casos, existe un tipo peculiar todavía más grave: todos los atributos de dos o más ejemplos son idénticos excepto la clase (esto es infrecuente si existen atributos numéricos). Este tipo de errores se consideran frecuentemente como “inconsistencias” e incluso algunos métodos de minería de datos no pueden “digerirlos” (llegando a dar errores de ejecución), con lo que es importante en este caso eliminarlos: o bien eliminando todos los ejemplos inconsistentes o todos menos uno (la duda queda en decidir cuál se deja).

4.3 Transformación de atributos. Creación de características

La transformación de datos engloba, en realidad, cualquier proceso que modifique la forma de los datos. Prácticamente todos los procesos de preparación de datos entrañan algún tipo de transformación.

En los apartados siguientes vamos a ver aquellas operaciones que *transforman* atributos. Veremos aquellas que transforman un conjunto de atributos en otros, o bien derivan nuevos atributos, o bien cambian el tipo (mediante numerización o discretización) o el rango (mediante escalado). La selección de atributos (eliminar los menos relevantes) en realidad no transforma atributos y, en consecuencia, se deja para el capítulo siguiente. Existen otras transformaciones (agregación, pivotamiento, etc.) que afectan a toda la tabla o a varias tablas y parcialmente se han visto en el capítulo anterior (junto a las herramientas OLAP). También volveremos sobre ellas en el capítulo siguiente.

Comenzamos por las transformaciones que crean nuevos atributos, inicialmente si el objetivo es reducir la dimensionalidad (sustituyendo estos nuevos, en un número menor,

por los viejos) o aumentar la dimensionalidad (manteniendo los viejos o sustituyéndolos por un número mayor de atributos).

4.3.1 Reducción de dimensionalidad por transformación

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

La reducción de dimensionalidad se puede realizar por selección de un subconjunto de atributos, como veremos en el capítulo siguiente, o bien se puede realizar por transformación, sustituyendo el conjunto de atributos iniciales por otros diferentes, que, geométricamente, se denomina proyección.

Existen muchas técnicas para realizar este tipo de proyección (o que pueden ayudar en este propósito): análisis de componentes principales, algunas técnicas del análisis factorial, uso de mapas auto-organizativos, etc.

Análisis de componentes principales

La técnica más tradicional, conocida y eficiente para reducir la dimensionalidad por transformación se denomina “análisis de componentes principales” (del inglés, “*principal component analysis*”, PCA, también llamada “método Karhunen-Loeve”) y consiste en transformar los atributos o variables originales x_1, x_2, \dots, x_m de los ejemplos en otro conjunto de atributos f_1, f_2, \dots, f_p , donde $p \leq m$. Este proceso se puede ver geométricamente como un cambio de ejes en la representación (proyección).

Lo interesante de esta transformación es que los nuevos atributos se generan de tal manera que sean independientes entre sí y, además, los primeros f 's tengan más relevancia (más contenido informacional) que los últimos. Es decir, f_1 es más relevante que f_2, f_3 más relevante que f_4 y así sucesivamente. Esto permite seleccionar los k -primeros atributos para diferentes valores de k . Por ejemplo, podremos elegir $k=3$ cuando queramos representar los datos en un gráfico tridimensional o podemos elegir un k mayor o menor según la información que queramos *ignorar*. La transformación nos asegura que si ignoramos los últimos $p-k$ atributos, estaremos descartando la información menos relevante.

Como hemos comentado en la introducción, el análisis de componentes principales se suele incluir generalmente, junto otras técnicas, en lo que se denominan análisis multivariante (*multivariate analysis*). De hecho, como el resto de estas técnicas se basa en conceptos algebraicos. Veamos, de la manera más sencilla posible, en qué consiste el análisis de componentes principales.

El objetivo es convertir de un vector original x_1, x_2, \dots, x_m de los ejemplos en otro vector f_1, f_2, \dots, f_p . Algebraicamente, si suponemos $m=p$ (aunque no variaría significativamente si $m>p$), esto se puede representar de la siguiente forma⁸:

$$\bar{f}_{(m \times 1)} = \begin{bmatrix} \bar{a}_1^T \\ \bar{a}_2^T \\ \vdots \\ \bar{a}_m^T \end{bmatrix}_{(m \times 1)} \bar{x}_{(m \times 1)} = A^T_{(m \times m)} \bar{x}_{(m \times 1)}$$

Es decir, necesitamos obtener una matriz $m \times m$ de coeficientes, tal que multiplicada por cada vector de atributos original (un ejemplo) nos de otro vector de atributos en el nuevo espacio de dimensiones. La pregunta, lógicamente, es la siguiente, ¿cómo calcular los vectores de coeficientes \bar{a}_i de tal manera que esta “proyección” consiga que los primeros atributos de los nuevos vectores f sean más relevantes que el resto?

La manera de conseguirlo en el análisis de componentes principales es asumir que queremos que la varianza de los nuevos atributos sea mayor para los primeros atributos que para los últimos. Se puede demostrar que el resultado se reduce a calcular, en primer lugar, la siguiente matriz de covarianzas:

$$S_{(m \times m)} = \frac{1}{n-1} \sum_{i=1}^n (\bar{u}_i)_{(m \times 1)} (\bar{u}_i)^T_{(1 \times m)}$$

Donde n es el número de ejemplos en el conjunto de datos, m es el número de atributos y los vectores de diferencia \bar{u}_i se calculan restando cada ejemplo i (vector de atributos x_1, x_2, \dots, x_m) con la media de los atributos de todos los ejemplos. Formalmente:

$$\bar{u}_i_{(m \times 1)} = \bar{x}_i_{(m \times 1)} - \bar{m}_{(m \times 1)} \quad \text{donde} \quad \bar{m}_{(m \times 1)} = \frac{1}{n} \sum_{i=1}^n \bar{x}_i_{(m \times 1)}$$

Una vez construida esta matriz de covarianzas S , los vectores de coeficientes se obtienen calculando los valores eigen (*eigenvalues*) de S :

$$e_1 \geq e_2 \geq \dots \geq e_m \geq 0$$

Los valores eigen son un conjunto especial de escalares asociados a un sistema lineal de ecuaciones [Marcus & Minc 1988]. El cómputo de los valores eigen y los vectores eigen (*eigenvectors*) de un sistema es de vital importancia en áreas como la física o la ingeniería. Calculando además los *eigenvectors* de S correspondientes con los e_j se obtienen los vectores \bar{a}_i , también llamados *ejes principales* de manera que tengan longitud 1 y sean ortogonales. Es importante mantener el orden de los *eigenvectors* marcados por los *eigenvalues*, ya que eso es lo que asegurará que el atributo f_1 sea más importante que el f_2 , el f_2 que el f_3 , etc.

Si después de lo anterior no se cree capacitado para calcular los valores f a partir de los valores x , no se preocupe, existen muchas herramientas que permiten realizar todo el proceso por usted. Lo importante es que el resultado de todo el proceso será que la representación de los ejemplos iniciales ha pasado de estar basada en los atributos originales a estar basada en otros nuevos atributos f . Lo relevante de la transformación,

⁸ Para facilitar la interpretación de las fórmulas, utilizamos la barra superior para indicar que se trata de un vector y añadimos en la parte inferior las dimensiones de vectores y matrices.

como hemos dicho al principio, es que los primeros atributos f_j tienen más relevancia que el resto. Además, todos los nuevos atributos son independientes entre sí. De alguna manera el proceso intenta concentrar las regularidades en los primeros atributos.

Evidentemente, la pregunta que uno se puede plantear es: "bien, tengo los nuevos atributos ordenados de mayor a menor relevancia y mi objetivo era reducir la dimensionaldad, ¿con cuántos me puedo quedar sin perder demasiada información?", o bien, ¿si me quedo con sólo tres qué parte de la información de los atributos iniciales estoy perdiendo?". Afortunadamente, el análisis de componentes principales puede responder a esta pregunta.

Para ello basta mirar el vector de *eigenvalues* \bar{e} . Por ejemplo, si dadas cinco dimensiones, hemos obtenido los valores para $\bar{e} = (3,65, 0,93, 0,22, 0,13, 0,07)$, podemos observar claramente la importancia relativa de cada uno de los nuevos atributos. Además, estos valores suman 5 (el total de dimensiones), con lo que es muy sencillo calcular porcentajes de relevancia (basta con dividir entre 5). Utilizando los valores anteriores, existen diferentes criterios para determinar con cuántas dimensiones quedarse. Por ejemplo, un criterio escoge tantas dimensiones necesarias para sumar al menos el 75 por ciento del valor total de la varianza (en el caso anterior deberían escogerse las dos primeras dimensiones). Otro criterio elige las dimensiones cuya varianza sea mayor que la media, o lo que es lo mismo, mayor que 1 (en el caso anterior se escogería sólo la primera dimensión). Finalmente, un último criterio representa gráficamente los valores \bar{e} en un gráfico denominado *scree diagram* y elige los atributos en el momento en el que la pendiente cambia más bruscamente. El *scree diagram* de los *eigenvalues* anteriores se muestra en la Figura 4.11:

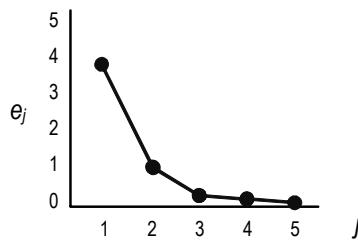


Figura 4.11. "Scree diagram" para representar la importancia de los componentes principales.

Antes de terminar con el análisis de componentes principales, es importante tener en cuenta que, en la mayoría de los casos, es imprescindible normalizar los datos antes de realizar el análisis. Cuando nos referimos a normalizar queremos decir, en primer lugar, centrar todos los atributos, que es equivalente a calcular el vector diferencia ya definido:

$$\bar{u}_i = \bar{x}_i - \bar{m} \quad \text{donde} \quad \bar{m} = \frac{1}{n} \sum_{i=1}^n \bar{x}_i$$

Y en segundo lugar, dividir cada elemento por su desviación típica (o estándar), es decir.

$$x'_{ij} = \frac{u_{ij}}{s_j} \quad \text{donde } s_j \text{ es la desviación típica del atributo } j$$

El único caso donde no hay que normalizar es cuando los atributos están expresados en la misma medida, y este hecho es importante. Por ejemplo, ingresos y gastos han de estar expresados en la misma moneda. Volveremos sobre este aspecto más adelante.

Lógicamente, lo que hemos relatado de análisis de componentes principales es sólo la punta del iceberg de una técnica que tiene muchas más aplicaciones y posibilidades. Un muy buen libro específico sobre el uso del análisis de componentes principales es [Aluja & Morineau 1999]. También se puede encontrar información en libros genéricos de análisis multivariante [Peña 2002b; Johnson 1999; Krzanowski 1988].

Otros métodos de reducción de dimensionalidad por transformación

Existen otros métodos para reducir la dimensionalidad que no se basan en una selección de atributos sino en una transformación de los mismos y pueden considerarse similares al análisis de componentes principales. Muchos de estos métodos se encuentran dentro de un área de técnicas más heterogéneas que se conoce como análisis factorial (*factor analysis*). Por ejemplo, existen otros métodos de **análisis factorial** para reducir el número de atributos (creando un conjunto menor de nuevos atributos) basados en mínimos cuadrados (ponderados o no), basados en máxima verisimilitud, basados en factorización de ejes principales, factorización alfa y factorización de imagen [Peña 2002b; Renche 2002].

Las técnicas anteriores, sin embargo, son sólo capaces de realizar una reducción atendiendo a las relaciones lineales entre las variables originales. Cualquier relación no lineal (por ejemplo $x_1 = \sin x_2$) no es capturada en las técnicas anteriores. Para capturar relaciones no lineales hay que optar por modelos no lineales.

Para empezar, por ejemplo, las redes de Kohonen o mapas autoasociativos (véase el Capítulo 13) se han utilizado generalmente para transformar los atributos existentes en otros (generalmente dos o pocos más) atributos diferentes. Estos métodos, al basarse en algoritmos de aprendizaje, son más lentos que el análisis de componentes principales clásico, pero generalmente permiten capturar relaciones no lineales.

En este sentido, un caso especialmente paradigmático es la relación entre el análisis de los componentes principales y las redes neuronales. En realidad, un perceptrón de p salidas y m entradas, produce las salidas según la ecuación siguiente:

$$\bar{o}_{(p \times 1)} = W^T \bar{x}_{(m \times 1)}$$

Donde la matriz W representa los pesos aprendidos. Esta ecuación es muy similar a la vista anteriormente para los componentes principales:

$$\bar{z}_{(m \times 1)} = A^T \bar{x}_{(m \times 1)}$$

Donde la matriz A era finalmente los *eigenvectors* calculados. Esta similitud permite utilizar redes neuronales y, en especial, redes neuronales multicapa (véase el Capítulo 13) para realizar análisis de componentes principales. Esto es particularmente interesante cuando la relación entre las variables no es lineal o sigue patrones complejos que el análisis de componentes principales no puede detectar (sólo transforma los patrones lineales, al basarse en la varianza).

De hecho, cuando en el análisis tenemos en cuenta *momentos estadísticos* de orden superior (la media y la varianza son los momentos de orden uno y dos, respectivamente), tenemos lo que se denomina **búsqueda de proyecciones exploratorias** (*Exploratory Projection Pursuit*, EPP) [Friedman 1987]. En estas técnicas se utilizan momentos estadísticos de orden superior, como el índice de *skewness* (orden tres) o el de *kurtosis* (orden cuatro), a

diferencia de los componentes principales, que sólo se basan en el momento de orden dos (la varianza). Veremos técnicas de EPP también en el Capítulo 13.

Otros métodos de reducción de dimensionalidad por transformación están basados en técnicas evolutivas, como se comenta en la Sección 15.3.3.

Una de las grandes limitaciones de la reducción de dimensionalidad por transformación es que los nuevos atributos obtenidos no son representativos, en el sentido de que no son los atributos originales del problema. Por tanto, un modelo comprensible obtenido a partir de atributos transformados de esta manera (por ejemplo, un árbol de decisión o un modelo de regresión) no puede ser interpretado por un experto, ya que las reglas están definidas en función de las z 's y no de las x 's originales.

Otra limitación es que el análisis de componentes principales clásico sólo es aplicable a atributos numéricos. Esto no es así para otros métodos (como las redes de Kohonen).

Ejemplo de reducción de dimensionalidad por transformación

Vamos a ver un ejemplo en el que vamos a ilustrar el uso de algunas de las técnicas anteriores para la reducción de dimensionalidad. El ejemplo que vamos a utilizar es el “Boston Housing” que, al igual que muchos otros conjuntos de datos que tratamos en este libro, se encuentra explicado en el Apéndice B. Este conjunto de datos tiene 14 atributos numéricos (en realidad uno de ellos es binario) y 506 instancias sobre las condiciones de viviendas en los suburbios de la ciudad de Boston. La proporción de ejemplos por atributo numérico es muy baja ($506 / 14 = 36,1$), con lo que podría ser interesante proceder a una reducción de dimensionalidad previa a la fase de minería de datos, para evitar así que los modelos se sobreajusten a los datos.

Para trabajar vamos a utilizar el sistema integrado de minería de datos SPSS Clementine (véase el Apéndice A), que incluye un nodo denominado “Factorial / PCA” que es similar a las características de análisis factorial y de componentes principales de muchos paquetes estadísticos. Una vez cargados los datos en el sistema, lo primero que podemos realizar es analizar la correlación entre atributos. Para ello, existe un nodo denominado “Estadísticos” (*Statistics* en la versión en inglés) que permite calcular los mínimos, máximos rangos, media, desviación típica, varianza, etc., para todos los atributos (de una forma similar a la Tabla 4.1, vista anteriormente con un resumen de atributos), así como la correlación de cada atributo con los demás. De la correlación entre atributos se ve que hay casi una decena de pares de variables con correlaciones positivas o negativas “fuertes” (mayores que 0,66 o menores que -0,66) y muchísimas correlaciones “medianas”. Esto sugiere claramente que existe redundancia entre las variables y que la reducción de dimensionalidad puede ser una buena opción.

A continuación, aplicamos un nodo “Factorial / PCA” y vamos a generar 14 componentes principales (para ello hemos de decir que el número máximo de componentes es 14 en las opciones *expertas*), usando el método de “Componentes Principales”. Obtenemos 14 nuevos atributos con nombres “\$F-Factor-PCA-X\$”, donde X va de 1 hasta 14, y en el propio nodo podemos analizar las funciones de definición de cada nuevo componente a partir de los atributos originales.

Pero quizás lo más interesante es si generamos el informe de “resultados del experto”, obteniendo la siguiente tabla:

VARIANZA TOTAL EXPLICADA			
Componente	Total	% de la varianza	% acumulado
1	6,546	46,757	46,757
2	1,650	11,782	58,539
3	1,349	9,635	68,174
4	0,887	6,332	74,507
5	0,851	6,078	80,585
6	0,660	4,714	85,299
7	0,535	3,824	89,123
8	0,403	2,879	92,003
9	0,277	1,980	93,983
10	0,252	1,802	95,785
11	0,213	1,520	97,305
12	0,183	1,307	98,612
13	0,134	0,957	99,569
14	6,033E-02	0,431	100,000

En esta tabla se muestran en la primera columna los *eigenvalues*, en la segunda el porcentaje de la varianza que cubre y en la última el porcentaje acumulado. Si utilizamos el criterio de quedarnos con tantas dimensiones como sean necesarias para sumar al menos el 75 por ciento del valor total de la varianza, deberíamos elegir los cinco primeros componentes.

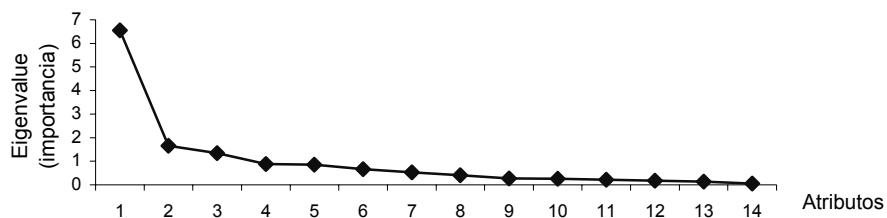


Figura 4.12. “scree diagram” de los componentes principales del problema “Boston Housing”.

El informe muestra mucha más información adicional, por ejemplo una matriz de componentes, donde se muestra la correlación entre los viejos atributos y los nuevos atributos.

Podemos intentar probar otro método de los muchos de que dispone este sistema. Por ejemplo, podemos probar el de “Mínimos cuadrados generalizados” (GMS). Con este método hay que tener cautela, ya que no podemos generar tantos atributos como queramos, ya que éste se basa en grados de libertad. Por ejemplo, si indicamos diez nuevos atributos, para este ejemplo no funcionaría. Vamos a probarlo con la generación de cinco nuevos atributos.

VARIANZA TOTAL EXPLICADA			
Componente	Total	% de la varianza	% acumulado
1	4,425	31,608	31,608
2	2,944	21,029	52,637
3	1,364	9,740	62,377
4	0,520	3,717	66,095
5	0,303	2,162	68,256

Como vemos, con este otro método, los cinco primeros componentes explican menos proporción de la varianza que el anterior, aunque (y esto se puede ver de manera positiva o negativa), la varianza está más repartida, en especial entre las dos primeras variables (en el caso del PCA, la primera variable capturaba casi el 47 por ciento de la varianza).

En cualquier caso, son métodos diferentes que encuentran relaciones lineales para hacer la proyección. Por ejemplo, si observamos la definición del primer componente usando PCA y usando GMS, vemos que ambas son funciones lineales de los atributos originales.

```

PCA1= 0.011009 * campo1 + -0.004113 * campo2 + 0.018907 * campo3 +
      -0.007736 * campo4 + 1.096869 * campo5 + -0.112823 * campo6 +
      0.004124 * campo7 + -0.055345 * campo8 + 0.01362 * campo9 +
      0.000751 * campo10 + 0.037494 * campo11 + -0.000842 * campo12 +
      0.017044 * campo13 + -0.011331 * campo14 + -0.964049
GMS1= 0.000139 * campo1 + -0.00005 * campo2 + 0.000535 * campo3 +
      -0.000033 * campo4 + 0.036886 * campo5 + -0.000842 * campo6 +
      0.000065 * campo7 + -0.001719 * campo8 + 0.112035 * campo9 +
      0.000075 * campo10 + 0.000643 * campo11 + -0.000007 * campo12 +
      0.000354 * campo13 + -0.00042 * campo14 + -1.12398

```

También se ve, y sólo con mostrar uno de los cinco nuevos componentes, que son dos proyecciones muy diferentes.

Si quisieramos buscar una proyección no lineal con una herramienta de minería de datos podríamos utilizar métodos de agrupamiento y utilizar las distancias a los grupos formados como dimensiones, o, directamente, como veremos en el Capítulo 13, utilizar redes de Kohonen si queremos dos componentes.

4.3.2 Aumento de la dimensionalidad por transformación o construcción

Aunque hay muchos casos donde, razonablemente, puede parecer que reducir la dimensionalidad es positivo, podemos ver otros casos donde nos interesa aumentar el número de atributos. Incluso, hablaremos en muchos casos de generar nuevos atributos que son combinación de otros y, en cierto modo, son atributos redundantes.

Aumento de la dimensionalidad mediante núcleos

Si en el apartado anterior hemos visto muchos métodos que proyectan los atributos originales en otro conjunto de atributos nuevos y diferentes, con la propiedad de que el número de atributos después de la proyección sea menor que el inicial, también se puede hacer una transformación similar en la que el número de atributos después de la proyección sea mayor que el original.

El objetivo, en estos casos, es aprovecharse de la “maldición de la dimensionalidad”. Como no hay mal que por bien no venga, si aumentamos la dimensionalidad conseguimos, como hemos dicho anteriormente, que los datos se separen en el espacio, facilitando, por ejemplo, fronteras lineales donde antes no las había. Esto quiere decir que, si realizamos un aumento de dimensionalidad adecuado, podemos convertir algunos problemas no lineales o incluso irresolubles en problemas lineales, al “aclararse” el espacio.

Esto no es sencillo, pero se ha avanzado mucho en los últimos años. El concepto fundamental en este tipo de aumento de dimensionalidad es la utilización de funciones de

núcleo (*kernels*), que permiten realizar estas proyecciones de manera adecuada. No vamos a entrar aquí en ver algunos ejemplos de núcleos, ni en desarrollar este tipo de métodos. Veremos funciones núcleo en varios capítulos de la Parte III, en especial, el Capítulo 14, que está dedicado a las máquinas de vectores soporte, que son, quizás, el método que más utiliza la técnica de aumento de dimensionalidad mediante núcleos.

Como hemos visto ya anteriormente, existe la posibilidad de utilizar un método de aprendizaje automático, por ejemplo, un agrupamiento, para generar un nuevo atributo, en este caso, el grupo al que pertenece el atributo. Ésta es la base del **análisis discriminante lineal** y de los núcleos utilizados para añadir o transformar atributos. Veremos los discriminantes lineales en los capítulos 7 y 8 de modelización estadística.

Uno de los problemas de la transformación de unos atributos en otros para aumentar la dimensionalidad, y al igual que ocurría para la disminución de dimensionalidad, es que el nuevo conjunto de atributos no es comprensible.

Creación de características

La creación, o agregación, de características consiste en crear nuevos atributos para mejorar la calidad, visualización o comprensibilidad del conocimiento extraído. La mayoría o todos los atributos originales se preservan, con lo que hablamos de *añadir* nuevos atributos y no de sustituirlos, como en el caso anterior. Este proceso recibe muchos nombres: **construcción, creación o descubrimiento de características**, inducción constructiva, invención de atributos, escoge y mezcla (*pick-and-mix*)...

La importancia de añadir atributos se muestra patente cuando existen patrones complejos en los datos que no pueden ser adquiridos por el método de minería de datos utilizado. Por ejemplo, supongamos que tenemos los datos de la evolución de la apertura de pólizas en los últimos 21 meses. Queremos obtener un modelo de los datos para estimar futuros de contratos de pólizas. Los datos se representan en la Figura 4.13 en forma de pequeños círculos.

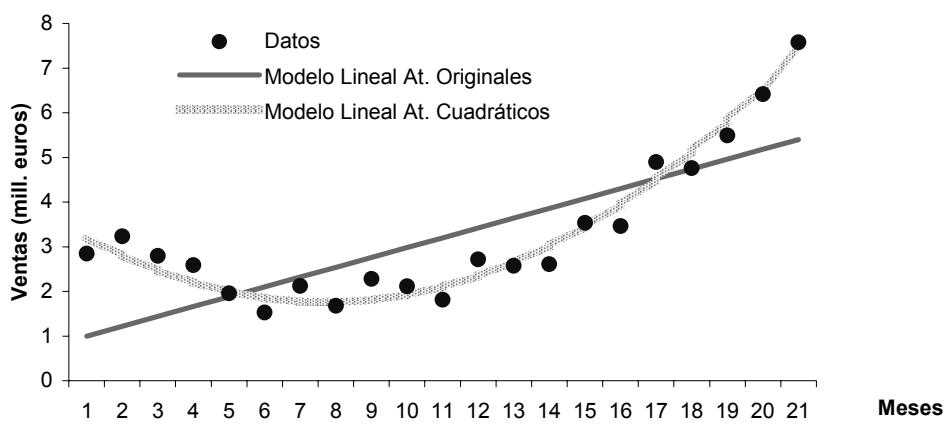


Figura 4.13. La importancia de crear características.

Podemos intentar realizar un modelo de regresión lineal con una variable de entrada y una de salida como sigue:

$$y = a_0 + a_1 x$$

El resultado del modelo obtenido ($a_0= 1$, $a_1= 2,2$) se muestra también en la Figura 4.13. Este modelo no se ajusta demasiado bien, como se puede observar. ¿Deberemos utilizar redes neuronales debido a la no linealidad de los datos?

En muchos casos no hay que llegar tan lejos. Si sospechamos que en los datos hay componentes cuadráticos, podemos añadirlos. La idea es crear un nuevo atributo $z=x^2$. Calculamos este atributo para todos los datos y ahora volvemos a realizar un modelo de regresión lineal con dos variables de entrada y una de salida como sigue:

$$y = a_0 + a_1 x + a_2 z$$

El resultado de este nuevo modelo “lineal” obtenido ($a_0= 3,2$, $a_1= 3,2$, $a_2= -4,3$) se muestra también en la Figura 4.13. Como se puede observar, el modelo con componentes cuadráticos se ajusta mucho mejor a los datos.

En el ejemplo anterior, un simple atributo adicional, como el cuadrado de un atributo original, ha sido suficiente para ajustar un buen modelo con una técnica relativamente sencilla. En general, sin embargo, disponemos de muchos atributos (tanto nominales como numéricos) y de métodos de aprendizaje más sofisticados. La “idea feliz” para crear nuevos atributos no resulta fácil cuando existen muchas posibles combinaciones.

Vamos a dar una serie de pautas para facilitar esta tarea, aunque, en muchos casos, la creación de buenas características o atributos depende no sólo de los propios datos, sino también del conocimiento que se tenga de los datos y del método de minería de datos que se vaya a utilizar. De hecho, a veces se categorizan las técnicas según están guiadas por los datos (*data-driven*), guiadas por la hipótesis o el modelo (*hypothesis-driven*), o guiadas por el conocimiento (*knowledge-driven*).

En primer lugar, nos podemos preguntar qué tipos de combinaciones podemos hacer. La respuesta no puede ser más tajante: cualquiera que nos imaginemos. Por ejemplo, el atributo $z= \pi(d/2)^2$, donde d es el diámetro de los discos que componen una unidad de almacenamiento, puede ayudar a la hora de detectar la capacidad máxima de almacenamiento del dispositivo.

Lógicamente, se pueden utilizar combinaciones muy complejas. Afortunadamente, es más útil (y más comprensible) el uso de relaciones simples entre los atributos. Dependiendo de si son atributos numéricos o nominales podemos distinguir las técnicas:

- Atributos numéricos: se utilizan, generalmente, operaciones matemáticas básicas de uno o más argumentos: suma, resta, producto, división, máximo, mínimo, media, cuadrado, raíz cuadrada, senos, cosenos, etc. Todas ellas retornan un valor numérico. También se pueden generar valores nominales a partir de valores numéricos, por ejemplo, crear una variable lógica que indique si un determinado valor numérico es mayor o menor que un determinado valor o que otro atributo, la igualdad estricta o aproximada entre atributos o valores, la desigualdad, etc.
- Atributos nominales: se utilizan, generalmente, operaciones lógicas: conjunción, disyunción, negación, implicación, condiciones M -de- N (M -de- N es cierto si y sólo si al menos M de las N condiciones son ciertas), igualdad o desigualdad. Todas ellas retornan un valor nominal. También se pueden generar valores numéricos a partir de valores nominales, por ejemplo, las variables X -de- N (se retorna el número X de las N condiciones que son ciertas).

Lo que no suele ser recomendable es la generación de atributos que el propio modelo sea capaz de obtener (en muchos casos no ganaremos ni siquiera en eficiencia, al tener más atributos). Por ejemplo, generar atributos “ $z = \text{'verdadero'}$ si $x < 27,2$, y $z = \text{'falso'}$ en caso contrario” será de poca utilidad si utilizamos como método un árbol de decisión, ya que, como veremos, los árboles de decisión son capaces de expresar condiciones del estilo “ $x < 27,2$ ”. Por el contrario, la condición “ $z = \text{'verdadero'}$ si $x_1 < x_2$ y $z = \text{'falso'}$ en caso contrario” puede ser muy útil, porque la mayoría de sistemas de aprendizaje de árboles de decisión no permiten comparar variables. En cambio, puede no aportar ninguna mejora significativa si usamos una red neuronal. De modo similar, generar un atributo $z = 2\pi x$ cuando estamos utilizando regresión lineal es también inútil, ya que, en este caso, z depende linealmente de x y no aportará nada al modelo. En cambio $z = x_1 \times x_2$ representa una relación no lineal, ya que multiplica dos variables de entrada, y, por tanto, puede ser útil para un modelo de regresión lineal.

Un tipo especial de atributos “numéricos” son las fechas. Hay que ser muy cautelosos con ellas, ya que generalmente la fecha (o incluso la hora) dan muy poca información. En general, las fechas hay que transformarlas en nuevos atributos más significativos. La primera idea es generar tres atributos: día, mes y año, aunque se pueden generar otros relacionados, como se ve en la Figura 4.14. De este tipo de atributos hablaremos en el capítulo siguiente, al tratar de las “jerarquías”.

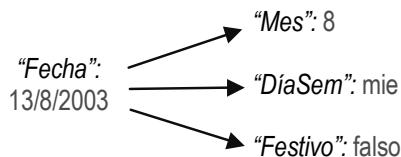


Figura 4.14. Convirtiendo fechas en atributos más significativos.

Otra de las cosas importantes es si tenemos información de fechas relativas, por ejemplo, la edad, o tenemos fechas absolutas, pero no la utilizamos apropiadamente. Si tenemos la fecha de nacimiento de los clientes y han realizado varias compras, nos interesa la *edad en el momento de la compra*, ya que si consideramos la edad actual, una persona que hubiera comprado con 16 años un producto, años después parecería que lo ha comprado con mucha más edad.

El caso anterior es un ejemplo de una transformación conocida como “separación de campos”. Por ejemplo, la dirección del viento puede estar codificada como “NE”, “SO”... Puede ser interesante crear dos atributos “componente_NS” con valores “N”, “S” o “-” y una “componente_EO” con valores “E”, “O” o “-”. Del mismo modo, se puede separar un código postal como “46019” en “46” (provincia), “460” (localidad), “46019” (barrio).

Tampoco hay que desestimar el uso de herramientas gráficas para ver si existen relaciones entre variables lineales o no lineales y crear derivados a partir de ellas. Por ejemplo, los gráficos de dispersión etiquetados, como el de la Figura 4.9, pueden sugerir la creación de una nueva variable como combinación lineal o no lineal de las representadas en el gráfico al verse zonas claramente marcadas en la gráfica.

Finalmente, el conocimiento del dominio es el factor que más determina la creación de buenos atributos derivados. La siguiente tabla muestra algunos ejemplos de atributos derivados en distintos dominios:

ATRIBUTO DERIVADO	FÓRMULA
Índice de obesidad	Altura ² / peso
Hombre familiar	Casado, varón e "hijos>0"
Síntomas SARS	3-de-5 (fiebre alta, vómitos, tos, diarrea, dolor de cabeza)
Riesgo póliza	X-de-N (edad < 25, varón, años de carné < 2, vehículo deportivo)
Beneficios brutos	Ingresos - Gastos
Beneficios netos	Ingresos – Gastos – Impuestos
Desplazamiento	Pasajeros * kilómetros
Duración media	Segundos de llamada / número de llamadas
Densidad	Población / Área
Retardo compra	Fecha compra – Fecha campaña

Tabla 4.2. Ejemplos de atributos derivados.

Una regla bastante útil a la hora de intentar que no se disparen las posibles combinaciones es no intentar poner varias combinaciones similares de los mismos atributos. Por ejemplo, no es conveniente añadir x/y si ya se ha puesto $x-y$.

Existen muchas otras técnicas (véase por ejemplo [Liu & Motoda 1998a]) para buscar dichas combinaciones (incluso evolutivas, como se ve en el Capítulo 15). También existen sistemas, como el LINUS, que se comenta en el Capítulo 12, que usan sistemáticamente la creación de atributos para extender la capacidad representacional de los modelos aprendidos. De hecho, LINUS permite aprender modelos relacionales utilizando técnicas proposicionales mediante la creación de características, del mismo modo que un atributo cuadrático derivado permite a una regresión lineal obtener modelos cuadráticos.

4.4 Discretización y numerización

Uno de los aspectos más importantes, sino el que más, de un atributo es el tipo. El hecho de que un atributo sea nominal o numérico determina en gran medida cómo va a ser tratado por las herramientas de minería de datos. En algunas situaciones puede ser conveniente convertir un numérico a nominal (discretización) o viceversa (numerización). Empecemos por la discretización.

4.4.1 Discretización

La discretización, o cuantización (también llamada “*binning*”) es la conversión de un valor numérico en un valor nominal ordenado (que representa un intervalo o “*bin*”). Por ejemplo, convertir una nota o calificación de 0 a 10 en una serie de valores ordenados {suspenso (0:4,99), aprobado (5:6,99), notable (7:8,49), sobresaliente (8,5:9,99), matrícula de honor (10)} es un ejemplo de discretización. En teoría, como hemos dicho, la discretización por intervalos genera un ordinal, es decir, un nominal ordenado. No obstante, el orden del atributo nominal puede ser preservado y utilizado por los pasos subsiguientes o bien puede olvidarse y tratarse el atributo como un valor nominal sin orden.

La discretización, en general, se realiza cuando el error en la medida puede ser grande o existen ciertos umbrales significativos. El caso de las notas es, probablemente, el más ilustrativo: la diferencia numérica entre 4,99 y 5 es mínima y equivalente a la diferencia numérica entre 3,99 y 4,00. Sin embargo, la primera diferencia (según el sistema de calificaciones tradicional en España) significa pasar de suspenso a aprobado, de

consecuencias más que jubilosas para los alumnos. Otra razón importante para discretizar es la integración de escalas diferentes. Por ejemplo, en ciertos países de Latinoamérica, como por ejemplo, El Salvador, el aprobado empieza en 6 y no en 5, como en España. En el caso de tener notas de varios países sería preferible integrarlas todas en algún tipo de escala común. Una escala discreta es, en este caso, más sencilla de realizar.

Existen más razones para discretizar como, por ejemplo, el hecho de que las diferencias en ciertas zonas del rango de valores sean más importantes que en otras o, dicho más técnicamente, cuando la interpretación de la medida no sea lineal. Por ejemplo, si tomamos las medidas de luz en una casa domótica, nos puede interesar más que tener una escala numérica de 0 a 1,000 lúmenes, tener una simple escala discreta: { oscuro (0:170), penumbroso (170:200), luminoso (200:1000) }, que puede ser mucho más práctico de cara a la obtención de modelos.

Una última razón y, quizá la más obvia, es cuando tenemos algunos atributos nominales y otros numéricos, y queremos que todos sean nominales para, por ejemplo, establecer reglas de asociación.

En general, si no conocemos el atributo que queremos discretizar o queremos realizar la discretización de muchos atributos nos podemos ayudar de técnicas que nos indican, principalmente, dónde hay que separar los intervalos y cuántos hay que obtener, ya que existen infinitas posibilidades. Cuando no existe una tarea asignada para los datos el proceso es más complejo, porque no se sabe muy bien con respecto a qué evaluar (qué discretizaciones son mejores o peores). Se pueden utilizar técnicas similares al análisis de correspondencias (véase la Sección 5.4.2) para ver si ciertas discretizaciones tienen un efecto mayor o menor sobre el resto de variables.

Cuando la tarea final que se pretende es clasificación, los métodos de discretización son más sencillos y se basan en medidas de separabilidad o entropía. Por ejemplo, si tenemos unos datos con 40 por ciento de elementos de la clase *a* y 60 por ciento de la clase *b* y queremos discretizar un atributo *x*, una discretización { $x < \alpha_1$, $x \geq \alpha_1$ } que deje una proporción de clases (5%, 95%) para $x < \alpha_1$, y una proporción (75%, 25%) para $x \geq \alpha_1$, es mucho mejor que una discretización { $x < \alpha_2$, $x \geq \alpha_2$ } que deje una proporción de clases (35%, 65%) para $x < \alpha_2$, y una proporción (50%, 50%) para $x \geq \alpha_2$. La primera ayuda a discriminar la clase mucho más que la segunda y, en este sentido, es preferible. Este tipo de criterios son los que se utilizan generalmente en los sistemas de aprendizajes de reglas y de árboles de decisión, ya que estos métodos inicialmente sólo trataban valores nominales y luego se adaptaron, mediante técnicas de discretización, para tratar valores numéricos. De hecho, muchas de las medidas de separabilidad que pueden utilizarse en discretización fueron inventadas propiamente para los árboles de decisión, como el índice GINI o la ganancia (Gain). Comentaremos estos índices en el Capítulo 11 de árboles de decisión y sistemas de reglas.

Ejemplo

Veamos un ejemplo sencillo de discretización. Para ello vamos a considerar de nuevo el conjunto de datos "iris" (véase Apéndice B), que consiste en 150 ejemplos de lirios de tres tipos, con cuatro atributos numéricos (*sepallength*, *sepalwidth*, *petallength*, *petalwidth*) y un quinto nominal, que representa la clase o tipo de lirio (*setosa*, *versicolour*, *virginica*). Vamos a

utilizar este ejemplo para ilustrar técnicas de discretización, utilizando la herramienta WEKA (véase el Apéndice A). Empecemos reconociendo los cuatro atributos numéricos:

	Min	Max	Media	Desv.Típ.
<i>sepallength</i>	4,3	7,9	5,84	0,83
<i>sepalwidth</i>	2,0	4,4	3,05	0,43
<i>petallength</i>	1,0	6,9	3,76	1,76
<i>petalwidth</i>	0,1	2,5	1,20	0,76

La discretización más sencilla (“*simple binning*”) es aquella que realiza intervalos del mismo tamaño y utilizando el mínimo y el máximo como referencia. Para ello se restan el máximo y el mínimo, y el valor resultante se divide por el número de intervalos deseados. Al primer y al último intervalo resultante se le añaden los valores desde $-\infty$ y hasta ∞ , respectivamente. Por ejemplo, si abrimos el archivo “iris.arff” en el WEKA y utilizamos el filtro “DiscretizeFilter”, utilizando cinco bins (intervalos) (“-B 5”), discretizando los cuatro primeros atributos (“-R 1,2,3,4”) y no utilizamos la opción por defecto “useMDL”, es decir “DiscretizeFilter -B 5 -R 1,2,3,4”, tenemos la siguiente discretización:

Atributo	1	2	3	4	5
<i>sepallength</i>	(-inf-5,02): 32	(5,02-5,74): 41	(5,74-6,46): 42	(6,46-7,18): 24	(7,18-inf): 11
<i>sepalwidth</i>	(-inf-2,48): 11	(2,48-2,96): 46	(2,96-3,44): 69	(3,44-3,92): 20	(3,92-inf): 4
<i>petallength</i>	(-inf-2,18): 50	(2,18-3,36): 3	(3,36-4,54): 34	(4,54-5,72): 47	(5,72-inf): 16
<i>petalwidth</i>	(-inf-0,58): 49	(0,58-1,06): 8	(1,06-1,54): 41	(1,54-2,02): 29	(2,02-inf): 23

Observando, por ejemplo, el atributo *sepallength*, tenemos que los intervalos tienen un tamaño fijo de $(7,9 - 4,3) / 5 = 0,72$. En la tabla anterior también se muestra la frecuencia absoluta (el número de individuos que caen en cada intervalo). Este mismo ejemplo nos muestra que utilizar el mismo número de intervalos para todos los atributos y establecer los puntos de corte con una regla tan sencilla puede ser limitado, ya que el número de intervalos es fijo.

Vamos a utilizar ahora una discretización más adaptable. Esta discretización la presentaron [Fayyad & Irani 1993] y se basa en el principio MDL, teniendo en cuenta, además, la discriminación realizada sobre la clase a predecir (a diferencia del anterior, este método no se puede utilizar si no tenemos una clase definida). Utilicemos WEKA con la opción “useMDL” usando el atributo 5 como clase (“-c 5”) y además la opción “findNumBins” (“-O”) para que optimice el número de intervalos “WEKA.filters.DiscretizeFilter -O -R 1,2,3,4 -c 5”. Si volvemos a realizar la discretización sobre los datos originales, tenemos un resultado bastante diferente:

Atributo	Discretización
<i>sepallength</i>	10 intervalos, el primero (-inf-4,66), y después de tamaño fijo 0,36
<i>sepalwidth</i>	10 intervalos, el primero (-inf-2,24), y después de tamaño fijo 0,24
<i>petallength</i>	7 intervalos, el primero (-inf-1,59), y después de tamaño fijo 0,59
<i>petalwidth</i>	7 intervalos, el primero (-inf-0,34), y después de tamaño fijo 0,24

Esta discretización, aunque puede ser mejor de cara a la separación de clases, no es buena o no es posible para otros cometidos, por ejemplo, reglas de asociación donde no tenemos clase. En cualquier caso, en ambos tipos de discretización se producen intervalos con muy pocos individuos, mostrando que la discretización por intervalos fijos no se adecua bien a estos atributos. Una discretización que, en algunos casos, puede resultar más adecuada es aquella en la que cada intervalo tiene un número constante de individuos o aquella discretización que mantiene una distribución similar a la normal. En muchos de estos casos la discretización ya no puede ser automática, como hemos visto en los ejemplos anteriores, sino que se tiene que realizar de forma manual.

Para saber más sobre discretización se puede consultar [Lui et al. 2002].

4.4.2 Numerización

La numerización es el proceso inverso a la discretización. Aunque es menos común que la discretización, también existen casos donde puede ser extremadamente útil. Un primer caso obvio donde la numerización es útil es cuando el método de minería de datos que vamos a utilizar no admite datos nominal. Un claro ejemplo es si pretendemos utilizar regresión lineal para obtener la influencia de cada “factor”. En general, es preciso para muchos de los métodos de modelización estadística, como la regresión logística (o multinomial), el análisis ANOVA, los modelos *log-linear*, los discriminantes paramétricos o no paramétricos, etc.

En todos estos casos lo que se suele hacer es lo que se denomina **numerización “1 a n ”**, que es la creación de varias variables indicadoras, *flag* o *dummy*. Si una variable nominal x tiene posibles valores $\{a_1, a_2, \dots, a_n\}$ creamos n variables numéricas, con valores 0 o 1 dependiendo de si la variable nominal toma ese valor o no. En general, esta aproximación genera más variables de las necesarias, ya que si tenemos que $a_1=0, a_2=0, \dots, a_{n-1}=0$ sabemos con certeza que $a_n=1$. Por tanto, es suficiente con generar $n-1$ variables numéricas en vez de n . Esto se denomina numerización “1 a $n-1$ ”.

Esta numerización “1 a n ” (o “1 a $n-1$ ”) es también útil cuando se espera que sólo uno de los posibles valores sea significativo, en especial cuando el método de minería de datos a utilizar puede generar una condición para cada valor. Por ejemplo, si en una inmobiliaria tenemos un atributo para el tipo de inmueble, que puede tomar el valor “adosado / chalet / piso / negocio / solar”, si queremos estudiar qué tipo de inmuebles son más interesantes para una constructora, la partición de dos reglas formada por las condiciones “inmueble_solar = V” e “inmueble_solar = F” es mucho más breve y general que la partición en cinco reglas dada por las condiciones “inmueble = solar”, “inmueble = adosado”, “inmueble = chalet”, “inmueble = piso” e “inmueble = negocio”. Por ejemplo, un árbol de decisión sin particiones nominales binarias repetiría prácticamente cuatro modelos idénticos para los casos en los que el inmueble no es solar, cuando todo esto se fusionaría para el caso de las etiquetas numerizadas.

En algunos casos es posible identificar un cierto orden o magnitud en los valores de un atributo nominal y entonces podemos realizar una numerización denominada **numerización “1 a 1”**. Por ejemplo, si tenemos categorías del estilo {niño, joven, adulto, anciano} podemos numerar los valores de 0 a 3, por ejemplo. En otros casos es interesante numerizar cuando se tiene un atributo nominal para el cual se puede discernir un orden, aunque sea

ficticio. Por ejemplo, supongamos que tenemos el nivel de estudios de un conjunto de individuos, con valores posibles { sin estudios, primarios, secundarios, bachillerato, universitario, doctor }. Nos puede interesar convertirlo en una escala de { 0, 1, 2, 3, 4, 5 }. Tanto en un caso como en otro, esto puede ser útil para obtener modelos más precisos y generales, incluso aunque nuestro método de minería de datos trate directamente con valores discretos.

4.5 Normalización de rango: escalado y centrado

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

Sin embargo, en otros casos, en especial para muchas técnicas vistas anteriormente (análisis multivariante, por ejemplo) es necesario normalizar todos los atributos al mismo rango. Por ejemplo, antes se ha visto que es preciso normalizar a la misma medida cuando el mismo atributo proviene de distintas fuentes (euros, pesetas). En otras técnicas, como el análisis de componentes principales, se requiere normalizar el rango entre cero y uno. También es necesario en la mayoría de algoritmos basados en distancias, ya que las distancias debidas a diferencias de un atributo que van entre 0 y 100 serán mucho mayores que las distancias debidas a diferencias de un atributo que va entre 0 y 10. Otras veces es necesario realizar escalados no lineales por la presencia de datos anómalos.

La normalización más común es la **normalización lineal uniforme** y se normaliza a una escala genérica entre cero y uno utilizando la siguiente fórmula.

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

El resultado de esta normalización es que la relación (el cociente) entre valores se mantiene. Para realizar esta normalización sólo es necesario conocer el máximo y mínimo de los valores dados para ese atributo. Determinar el máximo y el mínimo puede parecer trivial o directo. En algunos casos, sin embargo, no se escoge el máximo y mínimo natural sino que se establece agrandando o disminuyendo el intervalo. Para ello se puede mirar la media y la varianza o hacer un estudio por deciles. Por ejemplo, si los datos se distribuyen en una normal que aparece cortada por los lados, se puede decidir elegir un máximo mayor y un mínimo menor para capturar el 99 por ciento, digamos, de los datos de la distribución original. El proceso inverso también se puede realizar, es decir, disminuir el intervalo, cuando se sospecha que el máximo y el mínimo calculados se han podido ver afectados por valores anómalos (*outliers*). Esto produce, como consecuencia, valores por debajo de 0 y por encima de 1.

Una solución a este problema es el escalado *softmax* o **escalado sigmoidal**, en el que no se usa una transformación lineal, sino una transformación que es más pronunciada en el centro y más aplanada en los bordes. Para ello se utiliza una función sigmoidal (o alguna parecida a ella), como por ejemplo: $1/\exp(-20(x-0,5))$.

La siguiente figura muestra una posible función para realizar un escalado *softmax* (suponiendo que se le ha hecho un escalado lineal previamente o la variable original va entre cero y uno).

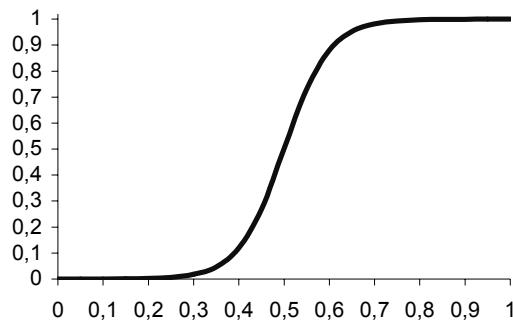


Figura 4.15. Función sigmoidal (o logística) para realizar un escalado softmax.

Por ejemplo los valores entre 0 y 0,25 irán a parar prácticamente al 0, mientras que los valores centrales (entre 0,3 y 0,7) se “ensancharán” cubriendo el rango 0,1 a 0,9. De modo similar los valores muy altos (entre 0,75 y 1) se equiparán con 1.

Existen otras normalizaciones, como el centrado o normalización de media cero, en el que a cada valor se le resta la media de los valores, quedando, por tanto, los datos centrados en cero. Si además del centrado, dividimos todos los valores por su desviación típica tenemos lo que se denomina **tipificación**. La tipificación es especialmente útil si los datos tienen una distribución normal. En este caso, el resultado de los datos tipificados se distribuyen normalmente alrededor de 0 y con desviación típica es 1.

Hasta ahora hemos comentado la normalización de un atributo, independientemente del resto. Una cuestión final es si tenemos varias dimensiones con las mismas unidades y otras con diferentes unidades. Por ejemplo, podemos tener tres atributos: atributo1 en euros, atributo2 en euros y atributo3 en metros. Se puede optar por normalizar independientemente los tres entre cero y uno, o se puede optar por normalizar el atributo1 y el atributo2 usando el máximo y el mínimo encontrados en ambos atributos, para mantener la relación.

4.6 Otras transformaciones

Existen muchas otras transformaciones de los datos. Podemos comenzar por el **escalado multidimensional** (*multidimensional scaling*). Ésta es otra técnica del análisis multivariante que se utiliza cuando tenemos, como datos, las distancias entre los individuos en vez de las características de los individuos. Esta técnica se utiliza para convertirlos en la forma tradicional atributo-valor, para poder ser tratados por la mayoría de técnicas de aprendizaje automático. Ejemplos típicos de datos que se presentan en forma de distancias o disimilitudes entre instancias son las distancias entre ciudades (y queremos obtener un espacio de varias dimensiones donde “ver” estas ciudades, que aunque no sea el espacio original, nos permita hablar de unos “atributos” de las ciudades), los precios de vuelos entre ciudades y los resultados de competiciones entre varios equipos.

Una primera idea es hacer de cada una de las distancias un atributo (la distancia al sí mismo es 0 lógicamente). Esta solución, sin embargo entraña, generalmente, mucha redundancia y genera tantos atributos como elementos, lo que la hace prácticamente desecharable. El escalado multidimensional proporciona soluciones más interesantes. Existen dos métodos diferentes: el escalado clásico que asume que las disimilitudes son las distancias euclídeas del espacio que se desea conseguir, y el escalado ordinal, donde disimilitudes mayores se convertirán en distancias mayores, pero no necesariamente en la misma proporción. No vamos a detallar ambos métodos (se puede consultar [Krzanowski 1988; Peña 2002b; Johnson 1999]), pero sí que vamos a ver un ejemplo. Supóngase que tenemos la media del tiempo (en microsegundos) que tarda la conexión entre varias sedes de una multinacional, como se representa en la siguiente tabla:

SEDE	Valencia	Vigo	Detroit	Dagenham	Munich
Valencia					
Vigo	259				
Detroit	323	527			
Dagenham	290	508	374		
Munich	95	203	201	257	

Nuestro objetivo es implantar un nuevo sistema y queremos determinar en qué sede situar el servidor. Una técnica de escalado multidimensional nos permitiría convertir⁹ la información anterior en dos dimensiones/componentes para cada sede y representarlas en un gráfico, observando qué sede puede ser más central. Dicho gráfico se muestra en la Figura 4.16:

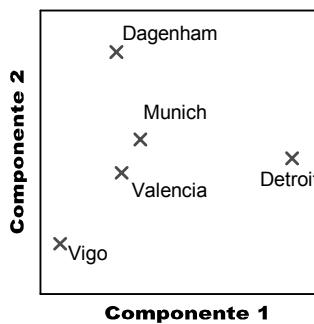


Figura 4.16. Representación gráfica de unos datos tras escalado multidimensional.

No sólo es útil para representarlos gráficamente. Estos atributos se pueden añadir a otros atributos de las sedes para tareas predictivas o descriptivas.

Siguiendo con otro tipo de transformaciones, se puede utilizar el agrupamiento para la preparación de datos. Muchas veces un proceso de agrupamiento puede ayudar a determinar diferentes estratos o segmentos en unos datos que se pueden tratar de manera diferente (con técnicas diferentes) o de los cuales se pueden seleccionar sólo unos pocos grupos. El *mixture decomposition* es un tipo especial de técnicas de agrupamiento que suponen que los datos provienen de diferentes distribuciones (posiblemente entremezcladas) e intentan separarlas o descomponerlas para ver esas distribuciones, grupos o fuentes

⁹ La conversión realizada por el escalado multidimensional a veces no tiene solución o tiene infinitas soluciones y se debe elegir una tal que cumpla las distancias originales.

originales. De modo similar, muchas veces una tarea de agrupamiento puede preceder una tarea de clasificación.

Finalmente, existen transformaciones especiales para variables temporales o en forma de series. Existen suavizados y diferenciados especiales que se pueden aplicar a series en la representación en el dominio de la amplitud. La transformada de Fourier, las transformadas de *wavelets* discreta u otras transformaciones que cambien los datos del dominio de la amplitud (una serie) al dominio de las frecuencias (un espectro) pueden permitir en muchos casos extraer patrones más fáciles e incluso más comprensibles. De este tipo de datos se tratará en el Capítulo 20.

Capítulo 5

EXPLORACIÓN Y SELECCIÓN

Una vez los datos están recopilados, integrados y limpios, todavía no estamos listos (en muchos casos) para realizar una tarea de minería de datos. Es necesario, además, realizar un reconocimiento o análisis exploratorio de los datos con el objetivo de conocerlos mejor de cara a la tarea de minería de datos. Incluso esta fase es imprescindible cuando se realiza minería de datos “abierta”, ya que tenemos todo el volumen de datos pero hemos de determinar los datos a seleccionar y las tareas a realizar sobre esos datos.

Este capítulo cubre un conjunto de técnicas diversas: algunas técnicas simples del análisis exploratorio de datos, técnicas de visualización previa, de agrupamiento exploratorio, técnicas de selección, ya sea horizontalmente, eliminando filas (muestreo), o verticalmente, eliminando atributos, interfaces gráficas y técnicas de consulta y agregación (ya sean más tradicionales al estilo de SQL, herramientas OLAP o incluso lenguajes de consulta para minería de datos).

La salida o resultado de las técnicas presentadas en este capítulo ya es una “vista minable con tarea asignada”, o dicho de otra manera, una vista minable tipada (entradas, salidas) con instrucciones sobre qué datos trabajar, qué tarea realizar y de qué manera obtener el conocimiento.

5.1 Introducción. El contexto de la vista minable

Imagínese que le cae del cielo una base o almacén de datos con una nota: “extraiga usted conocimiento de aquí”. Aparte de la sorpresa natural de ver llover bases de datos, que achacará posiblemente al cambio climático, usted se preguntará, entre otras cosas, lo siguiente:

- ¿Qué parte de los datos es pertinente analizar?
- ¿Qué tipo de conocimiento se desea extraer y cómo se debe presentar?

- ¿Qué conocimiento puede ser válido, novedoso e interesante?
- ¿Qué conocimiento previo me hace falta para realizar esta tarea?

Lógicamente, usted no será capaz de extraer conocimiento si no se le responde a dichas preguntas. Del mismo modo, una herramienta de minería de datos, no puede digerir un conjunto de datos y producir algo razonable, si no se le *orienta*. La razón fundamental del porqué esto es así radica no sólo en la incapacidad actual de las herramientas de realizar algunas tareas de una manera completamente automática, sino, fundamentalmente, en que la extracción de conocimiento viene a cubrir unas necesidades y expectativas, que deben indicarse, en cierto modo, de forma interactiva. Usted puede realizar la compra en un supermercado, por Internet, o se la puede hacer su mayordomo, pero, en ningún caso, podrá realizar una compra si no indica lo que quiere.

Por tanto, es necesario expresar y proporcionar las respuestas a las cuatro preguntas anteriores, ya sea mediante lenguajes de minería de datos, ya sea interactivamente en herramientas especializadas o seleccionando aquellas herramientas necesarias. Resulta, además, que incluso conociendo los datos y el dominio del que provienen, responder a algunas de ellas no es sencillo. Es necesario, en muchos casos, *explorar* los datos, el contexto y los usuarios de la información.

Las cuatro preguntas anteriores son, en realidad, una manera de clasificar el conjunto de preguntas que se podrían realizar, ya que, en el fondo, son preguntas que están interrelacionadas. Por ejemplo, si no se sabe el conocimiento que puede ser útil no se puede decidir qué parte de los datos lo pueden proporcionar. Por el contrario, si no se selecciona y se estudia un subconjunto de datos no se puede saber qué validez pueden tener los modelos extraídos y si finalmente van a ser útiles. Otro ejemplo similar es determinar el método de minería de datos; observando los datos puedo ver qué método puede ser más aconsejable. Sólo al determinar el método se puede saber si hay ciertos atributos que hará falta cambiar o eliminar. De modos diversos se interrelacionan estas preguntas acerca del qué, del dónde y del cómo.

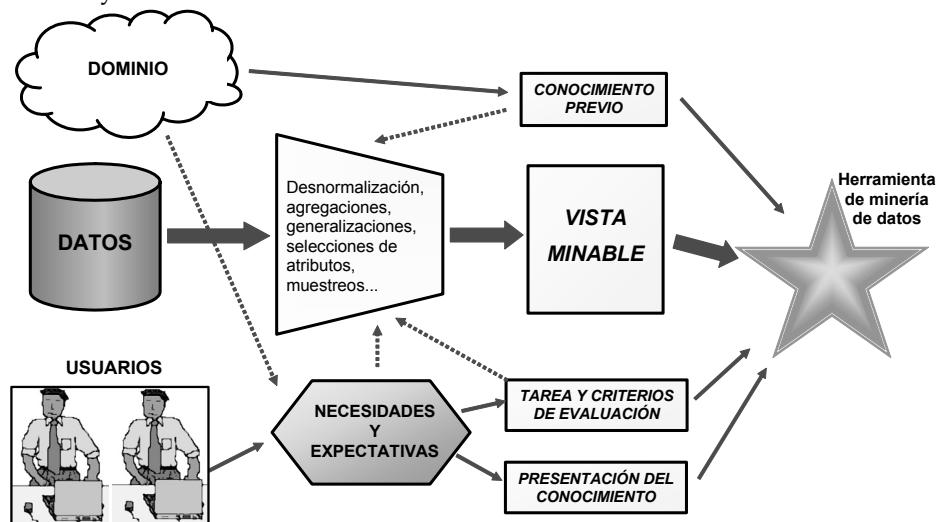


Figura 5.1. De los datos, dominio y usuarios a la vista minable y elementos asociados.

La Figura 5.1 intenta esquematizar el proceso que lleva de los datos, del conocimiento del dominio y de los usuarios a los cuatro aspectos anteriores que son necesarios para llevar a cabo la fase propia de minería de datos.

Como vemos, no es sólo necesario obtener la vista minable (una tabla con los atributos relevantes) sino que debe ir acompañada de la tarea a realizar sobre ella y cómo evaluarla, así como la forma de presentar el resultado final y, en su caso, el conocimiento previo necesario. Demos nombre y extendamos las cuatro preguntas anteriores

- **Vista minable:** ¿qué parte de los datos es pertinente analizar? Una vista minable [Ng et al. 1998] consiste en una vista en el sentido más clásico de base de datos: una tabla. La mayoría de métodos de minería de datos, como veremos, son sólo capaces de tratar una tabla en cada tarea. Por tanto, la vista minable ha de recoger toda (y sólo) la información necesaria para realizar la tarea de minería de datos.
- **Tarea, método y presentación:** ¿qué tipo de conocimiento se desea extraer y cómo se debe presentar? Se trata de decidir qué tarea (clasificación, regresión, agrupamiento, reglas de asociación, etc.), cuáles son las entradas y las salidas (en las tareas predictivas), con qué método, entre los existentes para cada tarea (árboles de decisión, redes neuronales, regresión logística, etc.) y de qué manera se van a presentar o se van a *navegar* los resultados (gráficamente, como un árbol, como un conjunto de reglas, etc.).
- **Criterios de calidad:** ¿qué conocimiento puede ser válido, novedoso e interesante? En muchos casos hay que establecer unos criterios de comprensibilidad de los modelos (número de reglas máximo), criterios de fiabilidad (basados en medidas como la confianza para las reglas de asociación, la precisión para la clasificación, el error cuadrático medio para la regresión, etc.), criterios de utilidad (basados en medidas de cuándo son aplicables, como el soporte, qué beneficios se obtiene, a partir de matrices de costes, etc.), y criterios de novedad o interés (basados en medidas más o menos subjetivas).
- **Conocimiento previo:** ¿qué conocimiento previo me hace falta para realizar esta tarea? Tanto a la hora de construir la vista minable final o para ayudar al propio algoritmo de minería de datos puede ser necesario establecer e incluso expresar de una manera formal cierto conocimiento previo. Por ejemplo, las jerarquías de conceptos o de dimensiones OLAP permiten trabajar con los datos y generar atributos, existen funciones que pueden utilizarse en medidas de similitud o a la hora de expresar los modelos, se pueden añadir otras tablas como conocimiento previo o incluso se pueden añadir otros modelos anteriores como apoyo o sobre los cuales revisar o construir uno nuevo.

Por ejemplo, supongamos que hemos recolectado la información sobre diagnósticos y recetas de atención primaria de toda una zona sanitaria. Nuestro objetivo es extraer conocimiento de estos datos. En primer lugar, antes incluso que mirar los datos, establecemos una serie de entrevistas con los jefes de servicio de atención primaria de la zona. Entre las cosas que salen a la luz en las entrevistas es su preocupación porque una multitud de nuevos medicamentos han aparecido recientemente para una serie de dolencias crónicas, y la mayoría de médicos prescriben de una manera aleatoria de entre los medicamentos generalmente efectivos, o como mucho, siguiendo patrones globales de

éxito de cada medicamento (prueba el “a” antes que el “b”, etc.). Esto tiene como consecuencia que, en muchos casos, a los pocos días el paciente vuelve a la consulta, y el médico le receta otro medicamento, hasta que dan con el medicamento realmente efectivo y que no muestre contraindicaciones no previstas. Entre las necesidades que aparecen en las reuniones, por tanto, se encuentra la de realizar modelos que determinen, según el paciente, qué medicamento prescribir primero, con el objetivo de resolver cuanto antes el problema sanitario del paciente, evitar nuevas visitas de los pacientes (reducción de visitas) y reducción de costes farmacéuticos.

A partir de este ejemplo, podemos establecer los componentes de la Figura 5.1. Como se pueden estudiar varias patologías, si nos centramos en una sola, tendremos que la vista minable va a formarse a partir de los diagnósticos de dicha patología y los medicamentos prescritos. El medicamento satisfactorio es el último prescrito, ya que, se supone, que si no hay más registros del mismo paciente y patología, el último medicamento fue bien. Por tanto habrá que realizar un tipo de consulta que nos seleccione el último medicamento prescrito a los pacientes de una patología (excluyendo los de menos de un mes, para tener más perspectiva). Los factores que vamos a incluir de los antecedentes son todos aquellos existentes del historial del paciente: parámetros generales: edad, tensión..., análisis de sangre, etc.

La tarea a realizar es una tarea de clasificación, ya sea completa o parcial (por ejemplo se podría realizar un subconjunto de reglas de asociación que ayudarán en los casos más claros). Debido a la característica de los usuarios (médicos) y a la exigencia de comprensibilidad de los modelos (para su validación facultativa), se decide que los patrones extraídos estarán expresados en forma de árboles de decisión, ya que los médicos están acostumbrados a seguir este tipo de árboles a la hora de hacer diagnósticos o prescribir medicamentos.

Los criterios de calidad se establecen a partir de la situación anterior. En primer lugar, el porcentaje de éxito acumulado es el criterio más importante. Es decir, que el número medio de medicamentos recetados (o intentos) sea menor. Esto está muy relacionado con la precisión del modelo, pero, como veremos en el Capítulo 17, existen medidas que ayudan a evaluar este tipo de problemas. También se incluirán matrices de costes, con el objetivo de incluir también los costes de las visitas y de los medicamentos. Adicionalmente se buscan modelos con pocas reglas y que se puedan aplicar con parámetros sencillos de los pacientes, sin necesidad de realizar pruebas caras o dolorosas, o pruebas que no se puedan realizar a todos los pacientes, para determinar el mejor medicamento.

Finalmente, existe una gran cantidad de conocimiento previo, extraído fundamentalmente del dominio y de las entrevistas con los especialistas. Por ejemplo, del dominio se puede extraer que la zona sanitaria habitual del paciente es fundamental a la hora de hacer la vista minable, ya que los pacientes que vienen de otras zonas pueden realizar la segunda visita en su zona de origen (por estar de vacaciones o ir de urgencias) y por tanto el criterio de considerar el “no retorno” como éxito puede ser un error. El conocimiento previo nos puede ayudar también en la transformación y selección de atributos relevantes. Los árboles de decisión no aceptan conocimiento previo, pero otros métodos si que podrían beneficiarse.

En realidad no debe cundir el desánimo ante la Figura 5.1, el ejemplo y la cantidad de aspectos a esclarecer. Lo que se intenta ilustrar es que obtener la vista minable, la tarea, el método, el conocimiento previo necesario, etc., es un proceso iterativo, que irá siendo más

sencillo a medida que se conocen los datos, el contexto, los usuarios y lógicamente, las técnicas de exploración y de minería de datos. Como del último aspecto ya quiera dar cabida cuenta este libro, vamos a comentar qué se puede hacer para conocer mejor los datos, el contexto y los usuarios.

Englobemos el reconocimiento en dos aspectos principales:

- **Reconocimiento del dominio y de los usuarios:** debemos reconocer el conocimiento que podría ser útil, además de intentar obtener las reglas ya existentes, ya sea para utilizarlas como conocimiento previo como para reemplazarlas por reglas y modelos mejores obtenidas por técnicas de minería de datos. Es importante determinar las decisiones que se toman frecuentemente y a partir de qué modelos se toman, si éstos tienen una base sólida o son simples reglas de negocio en la cabeza de uno o más directivos. Es importante determinar quién usará el conocimiento obtenido y qué tipo de presentación puede ser más aconsejable.
- **Reconocimiento y exploración de los datos:** de los datos seguimos transformando y seleccionando con el objetivo de obtener una “vista minable”, lista ya para ser tratada por las herramientas de minería de datos. A diferencia de las herramientas del capítulo anterior, las herramientas de exploración y selección requieren saber las expectativas y necesidades del dominio o, de una forma más concreta, la tarea y el conocimiento previo pueden influir más en estas transformaciones y selecciones.

Pasemos a extender los dos aspectos anteriores.

5.1.1 Reconocimiento del dominio y de los usuarios

Como hemos visto en el ejemplo anterior, para conocer qué se puede hacer con unos ciertos datos es necesario conocer el dominio y los usuarios. Si usted es el gerente o un directivo de una empresa o departamento que conoce bien, probablemente no necesite realizar este reconocimiento. Pero si usted es (o va a ser) un profesional de las tecnologías de la información, un estadístico o un profesional de una asesoría que va a dedicarse a minería de datos de varios clientes, usted será ajeno al dominio. Una de las primeras tareas a realizar será, por tanto, conocer y reconocer el dominio y los usuarios.

El procedimiento más similar a este reconocimiento para la minería de datos es el establecimiento de requerimientos realizado por un analista de *software*. A diferencia del caso del *software*, en las entrevistas (o cuestionarios) no buscaremos aquí casos de uso y escenarios de las operaciones mecánicas clásicas de los sistemas *software*, sino que buscaremos los **casos de usos y escenarios de las tomas de decisión**.

Para ello, realizaremos preguntas del estilo: ¿qué aspectos son cruciales en su negocio? ¿Qué reglas o modelos de decisión están utilizando? ¿Se pueden mejorar dichas reglas? ¿Qué base tienen dichas reglas? ¿Existen decisiones que se toman de una manera arbitraria o basándose en reflexiones personales no explícitas? ¿Existe documentación sobre decisiones anteriores? ¿Quiénes toman las decisiones? ¿Qué decisiones son críticas? ¿Los modelos deben ser comprendidos y validados por expertos? ¿Qué otros requerimientos exigiríamos a los patrones extraídos? ¿Qué conocimiento previo suele utilizar para apoyarse en sus decisiones? ¿Utiliza otras fuentes de datos externas para fundamentarse en sus decisiones? Y un largo etcétera de preguntas de este estilo. Algunas de estas cuestiones también son útiles y se pueden realizar a la hora de construir un almacén de datos o en el

momento de integración. Como veremos en el Capítulo 22 este reconocimiento se puede establecer como una fase previa a la minería de datos, en la que se establecen los *requerimientos y objetivos de negocio*.

Con una entrevista o cuestionario de este estilo se dará cuenta de que muchas decisiones se realizan con reglas informales, subjetivas y, en muchos casos, excesivamente simplistas o generalistas. Por ejemplo, la mayoría de aseguradoras de vehículos utilizan reglas generales del estilo “recargo por menos de dos años el carné o por menos de veinticinco años”, sin entrar en otros aspectos que, personalizando, podrían dar mejores resultados (por ejemplo por ser mujer, estudiante, no fumador, etc.). Muchas campañas de publicidad se enfocan a grupos de población (“jóvenes”, “amas de casa”, “niños”, etc.) cuya solidez puede dejar mucho que desear.

El resultado de este “reconocimiento” puede resumirse en una documentación u organizarse de una manera esquemática, estableciendo prioridades de análisis, destacando aquellas reglas de decisión importantes, que pueden mejorarse de manera significativa y para las cuales parece que disponemos de datos.

En general, se van descubriendo mayores posibilidades a medida que se va conociendo el dominio. Como hemos dicho al principio de este capítulo, sin este reconocimiento es imposible esclarecer las tareas, los métodos, los criterios de calidad, explorar los datos y el conocimiento previo.

5.1.2 Reconocimiento y exploración de los datos

Además del reconocimiento del dominio, debemos reconocer los datos. Para ello, lógicamente debemos conocer lo que significan y esto sólo es posible si quien lo realiza conoce el dominio o los datos (ya sea porque son sus propios datos y dominio o porque ha hecho el reconocimiento del dominio). El reconocimiento de los datos por tanto viene guiado por el interés y las necesidades establecidas en el reconocimiento de dominio. Sin éste, no se puede saber qué datos son relevantes ni qué tareas pueden ser útiles.

El reconocimiento de datos se suele conocer con distintos nombres en inglés (*data survey, exploratory data analysis, data fishing...*). De modo similar, en castellano, también se pueden utilizar términos diversos: exploración, prospección...

No obstante, hay que distinguir que el término “análisis exploratorio de datos” (*exploratory data analysis, EDA*), definido como “una serie de técnicas para investigar los datos para ver tendencias, patrones, errores y características” [Tukey 1977] tiene un enfoque diferente o más restrictivo al que vamos a ver aquí para la minería de datos. Gran parte de lo que se realiza en EDA existe en herramientas que no son de minería de datos, especialmente los *Executive Information Systems* (EIS) y, lógicamente, en herramientas estadísticas generales. En realidad, de nuevo no se puede marcar una línea de separación entre EDA y minería de datos, aunque se podría decir que EDA tiene un carácter más “explicativo”, de caracterización de los datos y no suele incluir modelos complejos ni predictivos.

Muchos de los gráficos que hemos comentado en el capítulo anterior se utilizan en este tipo de análisis. Pero, además, las herramientas informatizadas de EDA permiten interactuar con los gráficos. Por ejemplo, seleccionar un grupo que se ve en un gráfico de dispersión, hacer rotar un gráfico tridimensional (en su proyección en dos dimensiones)...

El objetivo de la exploración para la minería de datos es obtener una vista minable, con una tarea asignada. Para ello, se pueden utilizar distintas técnicas para obtener o refinar dicha vista: visualización, descripción, generalización, agregación y selección. En los puntos siguientes veremos estas técnicas. Todas ellas requieren, como hemos dicho anteriormente, conocer el dominio y el significado de los datos.

5.2 Exploración mediante visualización

En el capítulo anterior vimos algunos tipos de tablas, como la tabla de resumen de características, y algunas gráficas, como los histogramas y las gráficas de distribución. Estas gráficas, en general se centran en uno o dos atributos, a lo sumo, y el objetivo principal era, como vimos, la limpieza de datos. En este apartado veremos algunas gráficas más con un objetivo diferente, intentar sugerir tareas de minería de datos o patrones que puedan extraerse. Las gráficas que vamos a ver en este apartado se pueden caracterizar por dos aspectos: o bien son interactivas y permiten una exploración activa, o bien son multidimensionales, con lo que permiten observar muchos atributos a la vez.

Recientemente, ha aparecido el término “minería de datos visual” (*visual data mining*) [Wong 1999] con el significado de una minería de datos que se realiza manejando e interactuando con gráficos (otra interpretación es la del uso de interfaces visuales para la minería de datos, de lo que hablaremos al final del capítulo). En nuestra opinión el concepto de “minería de datos visual” es interesante como híbrido entre la minería de datos y la visualización de datos más tradicional [Cleveland 1993], pero, *en general*, no se puede hacer minería de datos sólo con gráficas. Precisamente lo que caracteriza la minería de datos de técnicas anteriores o de la perspectiva más clásica del análisis de datos es que los modelos son extraídos por algoritmos y, por tanto, no son *vistos* o descubiertos visualmente por el usuario (y posteriormente simplemente validados estadísticamente).

No creemos por tanto que sea conveniente dedicar un capítulo a las herramientas visuales o a las gráficas (aunque existan libros dedicados, como por ejemplo [Fayyad et al. 2002]). Son herramientas que son útiles en distintas fases y de distintas maneras y que se verán a lo largo de todo el libro (como empezamos en el capítulo anterior).

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

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

Estos dos objetivos marcan dos momentos diferentes del uso de la visualización de los datos (no excluyentes):

- **Visualización previa** (ésta es la que normalmente recibe el nombre de minería de datos visual): se utiliza para entender mejor los datos y sugerir posibles patrones o qué tipo de herramienta de KDD utilizar. La visualización previa se utiliza frecuentemente por *picapedreros*, para ver tendencias y resúmenes de los datos, y por *exploradores*, para ver ‘filones’ que investigar.
- **Visualización posterior** al proceso de minería de datos: se utiliza para mostrar los patrones y entenderlos mejor. La visualización posterior se utiliza frecuentemente

para validar y mostrar a los expertos los resultados de la extracción de conocimiento. De ésta trataremos en el Capítulo 19.

El primer tipo de visualización previa ya lo vimos en el Capítulo 4 y se trata de la visión multidimensional de las herramientas OLAP. Aunque realmente no se muestran gráficas, los datos sí que se muestran de manera visual (al menos parcialmente) y se puede interactuar con ellos, navegando por las dimensiones.

Existen otros tipos de visualizaciones más gráficas donde podemos apoyarnos para la preparación de datos. En general, las herramientas de minería de datos o estadísticas no nos van a sugerir qué gráfica utilizar, con lo que en general se requerirá de cierta experiencia y conocimiento de la herramienta para seleccionar qué gráfico nos interesa utilizar entre las decenas de gráficas que proporcionan los sistemas actuales.

Por ejemplo, incluso las hojas de cálculo permiten representar gráficos tridimensionales como los de la parte izquierda de la Figura 5.2. Lo que no podemos hacer en muchos casos es interactuar con la gráfica y obtener datos derivados a partir de ella. Por ejemplo, algunas herramientas estadísticas o de minería de datos, nos permiten desplazar un plano de corte sobre un eje e ir viendo los cortes instantáneamente, como se ve en la parte derecha de la Figura 5.2. En este caso podría ser útil para discretizar un atributo (entre un valor menor o mayor que el plano) para facilitar el agrupamiento. Por ejemplo, en la parte derecha de la Figura 5.2 se ven dos grupos de una manera mucho más clara que en la parte izquierda. Si esos dos grupos son los que realmente nos interesan podemos pasar el corte (convertido en un nuevo problema bidimensional, utilizando el atributo discretizado como filtro) a un método de agrupamiento (en este caso bastaría con un método lineal simple).

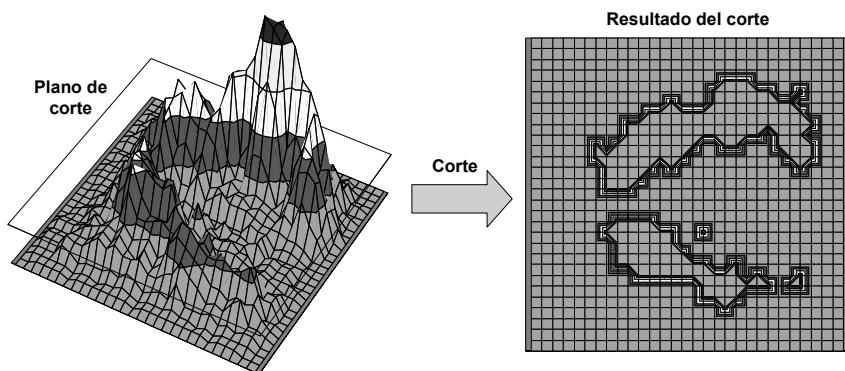


Figura 5.2. Interactuando con un gráfico para obtener una vista más simplificada.

Diferentes sistemas de estadística o minería de datos, tales como Statistica o *Statistica Data Miner* de StatSoft, SAS System o SAS Enterprise Miner, IBM Intelligent Miner, disponen de tipos de gráficas adecuadas para diferentes propósitos, que se pueden navegar, rotar, modificar o combinar.

5.2.1 Visualización multidimensional

La representación gráfica debe limitarse a las pantallas o al papel, que son bidimensionales. La gráfica de la izquierda de la Figura 5.2 no es más que una proyección de tres dimensiones en dos dimensiones, que se ayuda de la tonalidad o del color (éste sí que se puede considerar una tercera dimensión real) para que sea más inteligible. Sin embargo, en

muchas situaciones tenemos más de tres dimensiones, lo que plantea grandes problemas de cara a la visualización.

La técnica de visualización de datos multidimensionales más conocida es la visualización de coordenadas paralelas [Inselberg & Dimsdale 1990]. Se mapea el espacio k -dimensional en dos dimensiones mediante el uso de k ejes de ordenadas (escalados linealmente) por uno de abscisas. Cada punto en el espacio k -dimensional se hace corresponder con una línea poligonal (polígono abierto), donde cada vértice de la línea poligonal interseca los k ejes en el valor para la dimensión. La Figura 5.3 muestra un espacio 6-dimensional representado de esta forma.

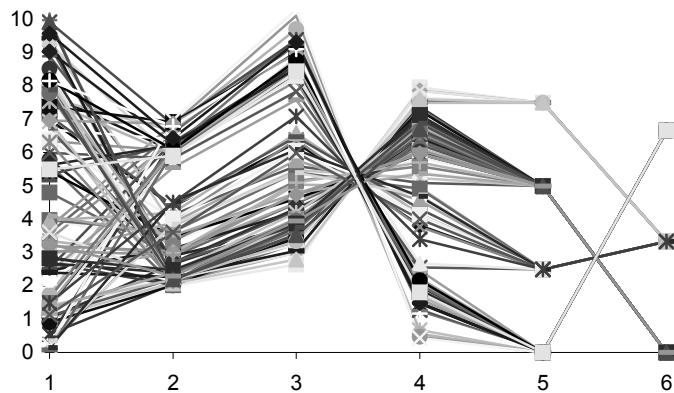


Figura 5.3. Gráfica de seis coordenadas paralelas con muchos ejemplos.

Aunque un gran número de ejemplos convierte la gráfica en una maraña, aun así, podemos detectar patrones. Por ejemplo, se puede observar que los atributos 1 y 2 no están correlacionados, mientras que el 2 y 3 están positivamente correlacionados, el 3 y el 4 están inversamente correlacionados, así como existe una relación entre la magnitud de 4 y las categorías del atributo nominal 5. Los atributos nominales 5 y 6 tienen una correspondencia clara entre categorías (en realidad el atributo 6 es dependiente del 5). Un gráfico similar es el gráfico de reconocimiento (*survey plot*), donde se muestran todos los ejemplos ordenados y la amplitud de cada línea muestra el valor para cada atributo (véase por ejemplo [Fayyad et al. 2002]). En general es más apropiado para valores numéricos.

En general, si los datos no correlacionan (que suele ocurrir para la mayoría de atributos) se ve una maraña de líneas. Otra cosa que suele suceder es que el orden de las dimensiones (atributos) es muy significativo para saber si hay relaciones. Por ejemplo, en la gráfica anterior no vemos si el atributo 6 y el 1 tienen relación o no.

Incluso en el caso de que existan relaciones, un número excesivo de ejemplos puede hacer que los puntos se “apiñen” o se tapen. Por ejemplo, en la parte derecha de la Figura 5.3 los valores nominales se montan unos sobre otros y se ven sólo cuatro líneas (aunque, en realidad, hay decenas de ejemplos). Este problema se conoce como *overplotting*. Esto se puede solucionar parcialmente con colores o haciendo un muestreo (mostrando sólo un subconjunto aleatorio de los ejemplos). En los casos que existan muchos ejemplos con los mismos valores, podemos incluir un grado de aleatoriedad en las magnitudes (esto se conoce como *jitter*), para que unos puntos no aparezcan exactamente encima de otros. Este truco permite ver la cardinalidad de ejemplos en cada sitio.

El gráfico anterior se puede utilizar también cuando hay pocos ejemplos. Por ejemplo, en la Figura 5.4, se muestran las características de doce pacientes de enfermedades cardiovasculares: el nivel de tabaquismo, colesterol, tensión, obesidad, alcoholismo, precedentes, estrés y riesgo estimado de enfermedades coronarias, normalizados de 0 a 10.

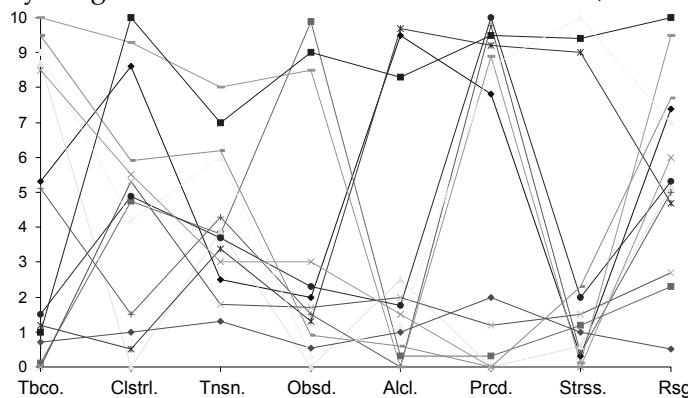


Figura 5.4. Gráfica de ocho coordenadas paralelas con pocos ejemplos.

En este caso se puede realizar un seguimiento de cada ejemplo.

Una variante del caso anterior cuando hay pocos ejemplos es la representación radial o circular, como se muestra en la Figura 5.5:

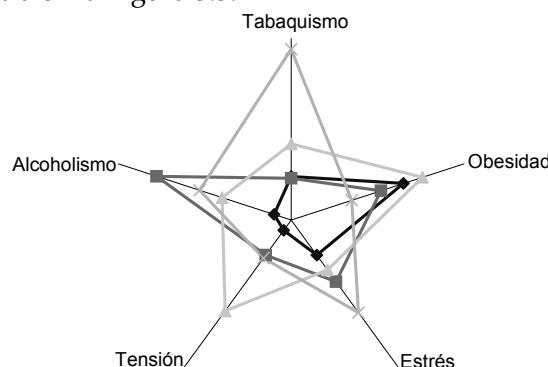


Figura 5.5. Gráfica radial de cinco dimensiones y cuatro ejemplos.

En realidad es similar al de coordenadas paralelas, con lo que no aporta mucho si se usa de esta manera. Frecuentemente, en vez de mostrar todos los valores en la radial, mostramos uno a uno, con lo que tenemos diferentes figuras para comparar los ejemplos. Éste es un uso más ilustrativo, como se ve en la Figura 5.6.

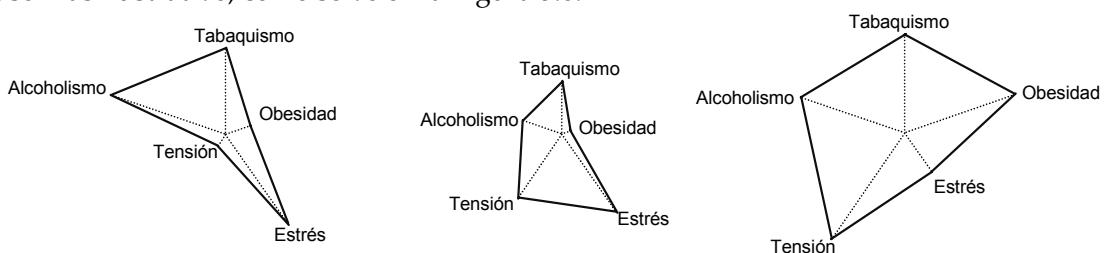


Figura 5.6. Tres pacientes diferentes vistos mediante gráfica radial.

Esta gráfica permite diferenciar individuos, o si se realizan medias, permite comparar grupos. Este tipo de representaciones con distintas formas que se da a diferentes objetos puede llevarse más allá mediante el uso de representaciones icónicas, en las cuales se utilizan figuras fáciles de reconocer por seres humanos (animales, caras, casas, etc.) y cada dimensión representa una característica de la figura. Por ejemplo, en el caso de los animales, un atributo puede representar el tamaño de la cabeza, otro el de las piernas, el del rabo, etc. Según sus partidarios, con este tipo de representaciones icónicas uno puede, de un solo vistazo, darse cuenta del tipo de individuo o grupo con el que estamos tratando. Lógicamente, cada representación icónica requiere una cierta familiarización previa para ser efectiva.

Otra de las ventajas de las representaciones icónicas frente a las radiales es que se pueden combinar más convenientemente valores discretos y continuos. Por ejemplo tener o no tener bigote puede utilizarse para un atributo binario, podemos representar un atributo con cuatro valores con cuatro tipos de nariz, etc.

5.3 Sumarización, descripción, generalización y pivoteamiento

La construcción de la vista minable es un proceso iterativo que pasa por conocer y visualizar los datos, combinados de diferentes maneras. Para esta combinación podemos utilizar operadores de consultas de bases de datos y operadores OLAP. Los datos con los que se trabaja en minería de datos son, muy frecuentemente, datos históricos que, por tanto, pueden agregarse a diferentes niveles de detalle temporal. Si, además, la estructura de los datos es multidimensional (por ejemplo un *datamart*) o existen campos de agregación, podemos obtener diferentes vistas concatenando (juntando o enlazando) diferentes tablas y agregando al nivel que deseemos.

Una pregunta que aparece generalmente en el entorno de la minería de datos es la siguiente: "si ya he decidido qué tablas y atributos son relevantes, ¿por qué debo construir una única tabla derivada, denominada vista minable? ¿No es suficiente con marcar dichos atributos y dejar a la herramienta de minería de datos que trabaje sobre la base de datos?". Existen dos razones fundamentales para contestar a esta pregunta. La primera es que dadas varias tablas, incluso aunque tenga claves ajena definidas, existen muchas maneras de concatenarlas, es decir, de combinar la información que contienen. Por tanto, es más difícil definir tareas concretas si no se clarifica exactamente la información sobre la que se van a definir. La segunda razón es quizás más importante: la mayoría de métodos de minería de datos sólo tratan con una única tabla. Si bien es cierto que veremos herramientas de la programación lógica inductiva y la minería de datos relacional en el Capítulo 12 que sí empiezan a ser capaces de trabajar con varias tablas, la mayoría de técnicas (en el caso de este libro, el resto de técnicas) sólo son capaces de trabajar con representaciones del estilo atributo-valor, es decir, una tabla.

Por tanto, debemos definir una consulta o vista minable. Para ello, las operaciones necesarias son aquellas de un lenguaje relacional (como por ejemplo el SQL), concatenacio-

nes (*joins*¹⁰), selecciones, proyecciones, agrupamientos/agregaciones, etc. La Figura 5.7 muestra precisamente la construcción de una vista minable a partir de un conjunto de tablas. Aunque las tablas tienen una estructura multidimensional y podamos apoyarnos en herramientas OLAP, en realidad, las operaciones necesarias son las típicas de una consulta SQL: concatenación, selección, proyección y agrupamiento.

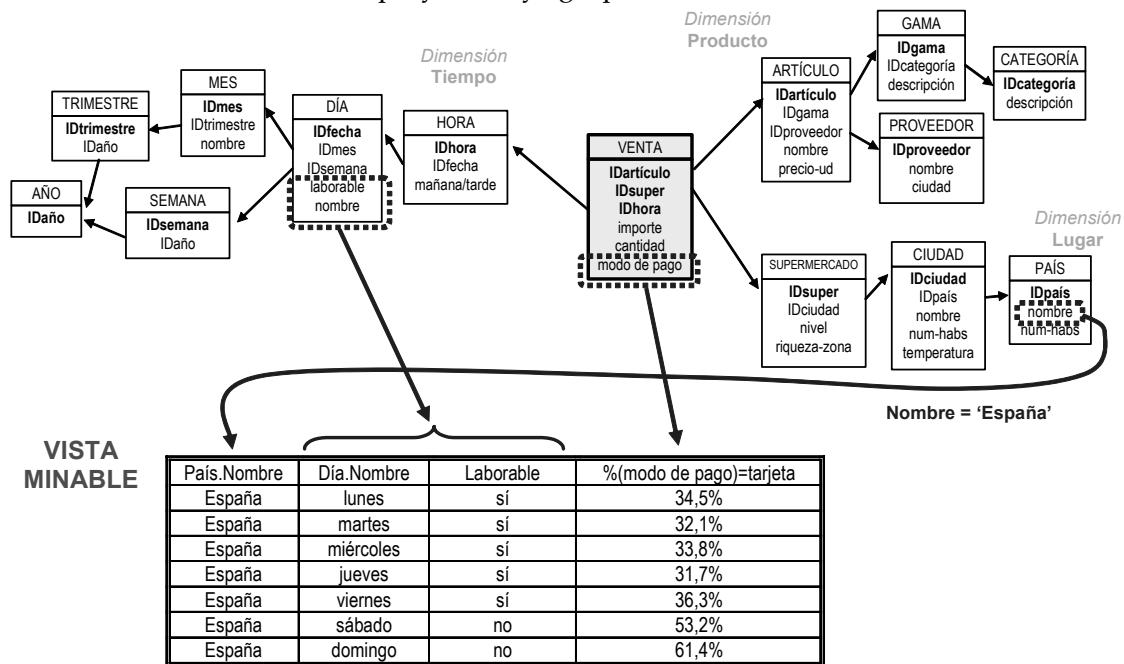


Figura 5.7. Selección de tablas, atributos, condiciones y niveles de agregación para obtener una vista minable.

Es quizá la concatenación de tablas aquella que permite juntar en una tabla la información proveniente de varias. Este proceso generalmente obtiene vistas desnormalizadas, en las que, por ejemplo, la tabla ciudad y país se funden en una sola, donde aparece el nombre de la ciudad y el nombre del país. Este tipo de desnormalizaciones contienen redundancia y, por tanto, patrones. Por tanto, hay que ser consciente de ellos, porque si incluyéramos todos los atributos para reglas de asociación, por ejemplo, tendremos patrones *redescubiertos* del estilo de dependencias funcionales como “ciudad → país” o, en el ejemplo anterior “día.nombre → laborable”.

5.3.1 Sumarización

La **sumarización** o **agregación** muestra los datos de una manera más resumida, permitiendo, precisamente, calcular valores agregados, que no son los datos directos registrados, sino datos derivados de ellos. Se puede considerar, en cierto modo, una generalización de los datos y, por tanto, suele facilitar el aprendizaje. Pero no es sólo una cuestión de eficiencia, sino, muchas veces, una necesidad. En la mayoría de eventos físicos,

¹⁰ El término inglés *join*, que significa conectar dos tablas por el valor de uno o más atributos de ambas tablas, ha sufrido múltiples y variadas traducciones en castellano: “junción”, juntura, junta, “yunción”, reunión, combinación.

cuando más se detallan los datos menos patrones suelen encontrarse. Esto es patente, por ejemplo, en los fenómenos meteorológicos: no podemos establecer un patrón de si hará sol el día 31 de diciembre de 2015 en Valparaíso (Chile), pero sí que podemos afirmar que en el mes de diciembre de 2015 la temperatura media será más alta que la temperatura media de julio. Tampoco podemos predecir cuántos pacientes ingresarán en urgencias el sábado que viene por la noche de 3:00h a 4:00h, pero podemos asegurar que los sábados por la noche hay muchas más urgencias que el resto de noches de la semana. Para establecer estos patrones, debemos agregar los datos.

Además, de la agregación aparecen nuevos atributos, que pueden ser mucho más significativos que los atributos más detallados. Por ejemplo, podemos tener los atributos típicos de los clientes (edad, estado civil, dirección...). En otras tablas podemos tener información sobre los productos que han comprado anteriormente y cuándo. Nos puede interesar generar nuevos atributos como por ejemplo “gasto medio por mes”, “número de productos comprados por año”, etc. Estos atributos son, en realidad, nuevos atributos por agregación y summarización (*feature aggregation*).

La summarización se puede utilizar no sólo para construir la vista minable directamente, sino para realizar un análisis exploratorio, similar, en cierto modo, a las gráficas del punto anterior. Una aplicación muy práctica de la summarización es la **comparación o discriminación de clases**. Consiste en sumarizar para dos o más clases, es decir agrupar, y ver las características para las submuestras formadas. Las clases pueden ser las que finalmente van a servir para una tarea de clasificación o puede ser cualquier atributo nominal (o numérico separado por intervalos) elegido para realizar el “contraste”. Por ejemplo, siguiendo con el tema de los pacientes de enfermedades cardiovasculares, podríamos querer ver cuál es la distribución por sexos de algunos de los otros atributos:

		Edad		Colesterol		Obesidad		Fumador	
Sexo	Card.	Media	Desv.	Media	Desv.	Media	Desv.	Sí	No
H	2341 (80,6%)	47,6	12,3	190,1	51,1	1,8	0,6	1803 (77%)	538 (23%)
M	563 (19,4%)	58,1	9,7	230	58,7	2,3	0,4	242 (43%)	321 (57%)
Todos	2904	49,6	11,7	197,8	54,2	1,9	0,5	2045 (70,4%)	859 (29,6%)

Esto sirve en muchos casos para comparar cardinalidades (en este caso hay muchos más hombres que mujeres con este tipo de enfermedades), para comparar las distribuciones de otros atributos numéricos (por ejemplo la obesidad parece más determinante en las mujeres que en los hombres) o las frecuencias de valores de atributos (el tabaquismo es mucho más alto en los hombres que en las mujeres), etc. Puede ser muy útil de cara a ver relevancias y dependencias de atributos. A veces se realiza un contraste de una clase con respecto a las demás, o respecto a todas (la distribución global).

Este tipo de operaciones se realizan más eficientemente con técnicas OLAP y de consultas que permitan calcular medias, cuentas, etc., eficientemente. No obstante, es de resaltar que este tipo de operaciones se pueden realizar con SQL básico (véase por ejemplo [Trueblood & Lovett 2001]).

Otra forma de realizar una summarización es determinar los “tipos” de individuos más frecuentes de cada clase.

Edad : 45-60, Colesterol> 150, Fumador=sí. Abarca el 50,2% de los hombres.

Edad : 60-inf. Abarca el 22,3% de los hombres

Edad : 30-45, Colesterol> 200. Abarca el 13,9% de los hombres.

Edad : 45-60, Colesterol> 150, Fumador=no. Abarca el 6,6% de los hombres.

Resto: 7% de los hombres.

Y lo mismo para la otra clase, para ver los casos típicos de cada clase.

Para hacer los resúmenes anteriores, son necesarias herramientas ya más específicas de summarización, para establecer estos grupos frecuentes o también para establecer grupos cuyas distribuciones se separan de la general (subgrupos diferenciados). De hecho, muchos autores consideran la summarización como una tarea propia de minería de datos. Nosotros la veremos desde el punto de vista de la tarea de agrupamiento y la de reglas de asociación.

5.3.2 Generalización y descripción

La generalización se puede realizar fundamentalmente mediante agregación por dimensiones, como se ha visto en el apartado anterior (mediante herramientas OLAP u otros medios más tradicionales) o se puede realizar mediante lo que a veces se llama “*inducción orientada al atributo*”. Este tipo de generalización se denomina a veces “descripción de conceptos” (*concept description*) e incluye: generalización multinivel, summarización, caracterización y comparación (como la vista en el punto anterior). Es un tipo de técnicas de consulta explorativas que pueden considerarse como un paso previo a la minería de datos (ésta es la perspectiva tomada en este libro) o se puede considerar como un tipo muy básico de minería de datos descriptiva.

Este tipo de generalización multinivel o descripción de conceptos se basa fundamentalmente en la definición de niveles conceptuales o jerarquías. Éstas se pueden definir para los valores, más que para los atributos, y de manera más flexible que las jerarquías de dimensiones de los *datamarts*, generalizando a través de los atributos descriptivos de las dimensiones. Veamos un ejemplo: podemos tener atributos para la fecha, el mes, el año, etc., dentro de una jerarquía de una dimensión. Moverse en estos atributos se realiza mediante operadores *drill* y *roll* como vimos en el Capítulo 3. En cambio, podemos hacer una generalización conceptual cuando distinguimos no entre la fecha A y la fecha B, o el mes A y el mes B, o la semana A o la semana B, sino cuando distinguimos entre “martes” y “jueves” o distinguimos entre “días festivos” y “días laborables”, utilizando para ello jerarquías de valor de la propia fecha. Como consecuencia de esto se puede eliminar la fecha (18/11/2007, por ejemplo), ya que es irrelevante.

Este tipo de jerarquías de valor se pueden definir, además de las jerarquías de la dimensión. Esto permite que algunas operaciones se puedan automatizar, en especial para los atributos nominales:

- Si un atributo tiene muchos valores y se puede generalizar, considerar las posibles generalizaciones. Por ejemplo, si tenemos las fechas (muy numerosas lógicamente), pero podemos generalizar en “días festivos”, “días laborables”, o en meses del año, etc., se debe considerar de estas generalizaciones cuál se ajusta más a la tarea a realizar.
- Si un atributo tiene muchos valores y no se puede generalizar, considerar borrar el atributo. Por ejemplo, los números de teléfono, si no tienen generalización, tal vez sea conveniente borrarlos.

- Si un atributo tiene pocos valores, dejar el atributo intacto. Por ejemplo, el estado civil de un cliente puede no requerir generalización.
- Si después de las generalizaciones existen atributos con muchos valores que ya no se pueden generalizar, considerar borrar el atributo.
- Para los atributos numéricos se puede utilizar generalización por intervalos, en varias escalas significativas.

El problema de la técnica anterior es, lógicamente, definir las jerarquías, aspecto que ya tratamos en el Capítulo 3, para las jerarquías de dimensiones y para los atributos de valor. La definición de estas jerarquías requiere, en muchos casos, obtener datos externos. Por ejemplo, buscar el calendario de festividades de los años de análisis (para saber si las fechas eran festivas o no), obtener datos del nivel de ventas del mismo ramo para saber si era año de recesión o año de bonanza, etc. Este proceso de creación de jerarquías ayudado por información externa se llama enriquecimiento (*enrichment* o *enhancement*).

5.3.3 Pivotamiento

Una operación muy usual a la hora de preparar la vista minable se conoce como pivotamiento y, como vimos en el Capítulo 3, forma parte de los operadores OLAP. La operación de pivotamiento cambia filas por columnas y, por tanto, realiza un cambio verdaderamente radical para una representación basada en pares “atributo-valor”.

El ejemplo más clásico de pivotamiento es de la cesta de la compra. Supongamos que unos grandes almacenes guardan una gran tabla de cestas de la compra, donde cada atributo indica si el producto se ha comprado o no. Existen unos 10.000 productos en los grandes almacenes y millones de cestas semanales. El objetivo del análisis es ver qué productos se compran conjuntamente (regla de asociación).

Lógicamente, los datos no caben en memoria, con lo que hay que ir trabajando en disco. Para tener algo de fiabilidad en las reglas hay que mirar al menos la raíz cuadrada de todas las cestas, eso obliga a seleccionar unas 1.000 filas (aleatoriamente) de la tabla para cada dos atributos que queramos evaluar.

Si este tipo de análisis se van a realizar frecuentemente, puede merecer la pena cambiar filas por columnas, como se muestra en la Figura 5.8.

Figura 5.8. Pivotamiento: cambio de filas por columnas.

#Cesta	Prod1	Prod2	Prod3	...	Prod10.000
1	Sí	No	No	...	No
2	No	No	No	...	Sí
3	Sí	Sí	No	...	No
4	Sí	No	No	...	No
5	No	Sí	Sí	...	Sí
...
10.000.000	No	No	Sí	...	Sí

Pivot

#Prod	Cesta1	Cesta2	Cesta3	...	Cesta10.000.000
1	Sí	No	Sí	...	No
2	No	No	Sí	...	No
3	No	No	No	...	Sí
4	No	Sí	No	...	Sí
5	Sí	Sí	No	...	No
...
10.000	No	Sí	No	...	Sí

Figura 5.8. Pivotamiento: cambio de filas por columnas.

Ahora, para observar si dos productos están asociados es sólo necesario tomar dos filas de la tabla y realizar, por ejemplo, un “o exclusivo” (XOR) entre las dos filas, para ver si están asociadas o no.

Las herramientas de reglas de asociación (como veremos en el Capítulo 9) hoy en día son muy eficientes y en muchos casos no necesitan este pivotamiento (o incluso puede ser

que si se realiza el pivotamiento, la tarea no se pueda definir), pero la idea sigue siendo válida para otras tareas o para sistemas específicos.

5.4 Selección de datos

La selección de datos es algo más que decidir qué tablas (o archivos, en su caso) se van a necesitar para la minería de datos y de qué manera concatenarlas. Esto podría estar ya decidido, pero todavía no sabemos qué atributos/variables necesitamos y cuántas instancias (ejemplos) van a ser necesarias. Dicho de otra manera, puede ser que no todas las columnas, ni todas las filas sean necesarias. El problema existente es precisamente que si seleccionamos como “vista minable” todo aquello que pueda ser relevante podemos acabar con una vista minable de cientos de columnas/atributos y millones de filas/registros. El tamaño de una tabla como ésta desborda la capacidad de muchas de las técnicas de minería de datos que veremos en la Parte III de este libro. Hemos de ser capaces de ver si podemos obtener unos primeros modelos (o incluso mejores modelos) con un subconjunto de las instancias y de las variables.

La selección de datos no tiene únicamente como objetivo la reducción del tamaño para obtener una minería de datos más rápida sino que, en muchos casos, puede permitir mejorar el resultado (tanto en precisión o en coste, por ejemplo utilizando muestreo estratificado o en comprensibilidad, por ejemplo utilizando reducción de dimensionalidad).

El proceso de selección de datos muchas veces se engloba dentro de un concepto más amplio, denominado reducción de datos (*data reduction*), aunque este término también puede incluir la agregación (por ejemplo si pasamos de instancias de cada día a instancias agregadas mensualmente), la generalización (por ejemplo si reemplazamos el atributo ciudad por región, siguiendo por ejemplo la jerarquía de alguna dimensión), o incluso la compresión de datos (por ejemplo eliminando datos redundantes).

En general, cuando tratamos de datos del estilo atributo-valor (es decir, una tabla), hay dos tipos de selección aplicables: selección horizontal (muestreo), donde se eliminan algunas filas (individuos) y selección vertical (reducción de dimensionalidad), donde se eliminan características de todos los individuos. Pasemos a ver estos dos tipos de selección.

5.4.1 Técnicas de muestreo

La manera más directa de reducir el tamaño de una población o conjuntos de individuos es realizar el muestreo. La gran mayoría de medidas y técnicas estadísticas y de sus aplicaciones, se basan en el concepto de muestra, es decir, no se trabaja sobre toda la población sino sobre un subconjunto de la misma. Un ejemplo diario es la realización de encuestas; con unos cientos o miles de llamadas se puede extraer a una población total de millones de personas.

En el caso de la minería de datos nos podemos plantear dos situaciones, dependiendo de la disponibilidad de la población:

- Se dispone de la población: en este caso se ha de determinar qué cantidad de datos son necesarios y cómo hacer la muestra. En muchos casos, por ejemplo, una muestra aleatoria no es lo más aconsejable. Por ejemplo, en el caso de una encuesta, se intenta escoger al menos un mínimo de individuos de cada tipología (edades, sexos,

nivel económico, rural/urbano, etc.). No obstante, ésta es la situación ideal, ya que se dispone (con mayor o menor facilidad) de la población total.

- Los datos son ya una muestra de realidad, por ejemplo, los datos recogidos en una base de datos y sólo representan una parte de la realidad. Por ejemplo, las reclamaciones realizadas a través de la web quedan registradas, mientras que el resto de reclamaciones no se registran. Es importante tener claro que el hecho de que esté en una base de datos no quiere decir que sea una muestra de realidad. Generalmente, la información de pedidos, clientes, proveedores, etc., de una empresa representan la “población total” de pedidos, clientes, proveedores, etc. Ésta es la situación comentada anteriormente.

Ya sea en uno u otro caso existen otras razones para querer realizar un muestreo sobre los datos disponibles (ya sean, a su vez, un muestreo o el total de la población). Entre las razones están, principalmente, la reducción del tamaño (con el objetivo de facilitar y agilizar los algoritmos de minería de datos), pero existen otras muchas (balanceado de clases, cobertura equilibrada del espacio, facilitar la visualización...).

Dependiendo de estas razones existen varios tipos de muestreo: aleatorio, estratificado, por grupos o exhaustivo. Veamos con detalle estos tipos de muestreo.

- **Muestreo Aleatorio Simple.** La premisa de este muestreo es que cualquier instancia tiene la misma probabilidad de ser extraída en la muestra. Puede ser con o sin reemplazamiento. La manera más sencilla de realizar un muestreo sin reemplazamiento de este estilo es asignar un número aleatorio a cada instancia y después ordenarlos por este valor. Se seleccionan los n primeros y tenemos una muestra de n instancias sin reemplazamiento. Para realizar un muestreo con reemplazamiento, una manera sencilla es, dado un total de m instancias, generar aleatoriamente un valor i entre 1 y m . Con este valor i se elige la i -ésima instancia del conjunto. Esta operación se repite hasta tener n valores. Lógicamente, existen maneras más efectivas de realizar ambos tipos de muestreo cuando no se desea *ordenar* una tabla grande ni se desea realizar n barridos de la misma tabla. Por ejemplo, se puede realizar un muestreo “sistemático” o “intercalado”, en el que se calcula un valor $k=m/n$ donde m es el tamaño total de los datos y n es el tamaño de la muestra deseada. Una vez obtenido este valor, se genera un valor aleatorio i entre 1 y k . Se escoge la instancia i y después se van eligiendo la $i+k$, $i+2k$, ..., hasta $i+(n-1)k$, haciendo un total de n instancias muestreadas. No obstante, este muestreo puede, en algunos casos, crear sesgos, aunque, en otros, es una solución bastante práctica. En ningún caso, sin embargo, se pueden realizar muestreos del estilo “los n primeros de la tabla”; aunque muy eficientes, generalmente pueden suponer un sesgo muy importante, ya que el orden en el que las instancias están almacenadas en la tabla puede depender de muchos factores: fechas, índices, métodos de optimización propios del sistema de gestión de bases de datos, etc.
- **Muestreo Aleatorio Estratificado.** El objetivo de este muestreo es obtener una muestra balanceada con suficientes elementos de todos los estratos, o grupos. Lógicamente, para poder hacerlo es necesario conocer los estratos o grupos de interés. Esto generalmente ocurre con problemas de clasificación, donde estos estratos son precisamente las clases existentes, y puede ser que se tengan pocas instancias de unas clases o muchas instancias de otras. No obstante, se puede plantear realizar un

agrupamiento previo al muestreo para descubrir los estratos. Sea como sea, el muestreo aleatorio estratificado se puede realizar de muchas maneras, dependiendo del número de elementos que se quiera para cada estrato. Por ejemplo, si existen suficientes ejemplos de todos los estratos y queremos n elementos de cada estrato, realizaremos un muestreo aleatorio simple sin reemplazamiento de cada estrato hasta obtener los n elementos de ese estrato. Finalmente, tendremos $n \cdot e$ donde e es el número de estratos. Sin embargo, muchas veces no tenemos suficientes ejemplos de todos los estratos. En esta situación lo mejor es realizar un muestreo aleatorio simple con reemplazamiento de aquellos estratos minoritarios. Como resultado, para algunos estratos tendremos más instancias de las que había originalmente en el estrato (sobremuestreo) y para otros estratos tendremos menos instancias de las que había originalmente en el estrato (submuestro). Las ventajas del muestreo estratificado no es sólo que se cubren suficientes estancias de todos los estratos, sino que además es estadísticamente aconsejable, especialmente cuando se realiza submuestreo, ya que los elementos del mismo estrato suelen ser similares y, por tanto, de alguna manera, se mantiene más diversidad en la muestra. Dicho de otra manera, es preferible pocos elementos, pero de todos los estratos, a tener muchos, pero de sólo algunos estratos. De ahí la manera en la que se suelen realizar los sondeos de opinión. En el Capítulo 17 veremos de nuevo las aplicaciones de la estratificación, especialmente cuando se tienen en cuenta costes de clasificación.

- **Muestreo de Grupos.** En cierto modo inverso al anterior, el muestreo de grupos consiste en elegir sólo elementos de unos grupos. El objetivo de este muestreo es generalmente descartar ciertos grupos que, por diversas razones, pueden impedir la obtención de buenos modelos.
- **Muestreo Exhaustivo.** Se trata de una exageración del muestreo estratificado, con motivaciones similares. Se procede de la siguiente manera: para los atributos numéricos (normalizados) se genera al azar un valor en el intervalo posible; para los atributos nominales se genera al azar un valor entre los posibles. Con esto obtenemos una instancia ficticia. El *quid* del procedimiento es que ahora buscamos la instancia real más similar a la ficticia. Se repite este proceso hasta tener n instancias. Nótese que el objetivo de este método es cubrir completamente el espacio de instancias y evitar poner muchos ejemplos de las zonas muy densas. Lógicamente, puede haber zonas donde no haya instancias, pero buscaremos de los alrededores la más cercana. Existen variantes de este tipo de muestreo, como por ejemplo el muestreo por cuadrícula, en el que las instancias ficticias no se calculan aleatoriamente sino como una cuadrícula o *grid* sobre el espacio de instancias. Otra variante más sofisticada es ir modificando la probabilidad de extraer una instancia a medida que se van extrayendo instancias, de forma que las instancias más próximas a las instancias ya extraídas tengan menos probabilidad que las demás.

En la Figura 5.9 se muestran las diferencias, realmente apreciables, que se pueden producir por utilizar diferentes tipos de muestreo (aleatorio simple, aleatorio estratificado, por grupos, exhaustivo), todos ellos con un tamaño de 20 instancias muestreadas. También se muestra punteada una posible clasificación realizada por una técnica simple (vecino más próximo), que ilustra, todavía más claramente, que los patrones detectadas por el modelo pueden variar drásticamente dependiendo de la forma en la que se haya hecho el muestreo.

Se puede observar que, en el ejemplo, el muestreo por grupos elimina la clase triángulo, que no parece respetar ningún patrón y sólo hace que empeoren los modelos. El muestreo exhaustivo parece también mostrar buenos resultados, debido especialmente a que hay zonas densas y muy desérticas en el espacio, y este muestreo corrige este tipo de situaciones. Lógicamente, también se podría utilizar una combinación de ambos tipos de muestreo.

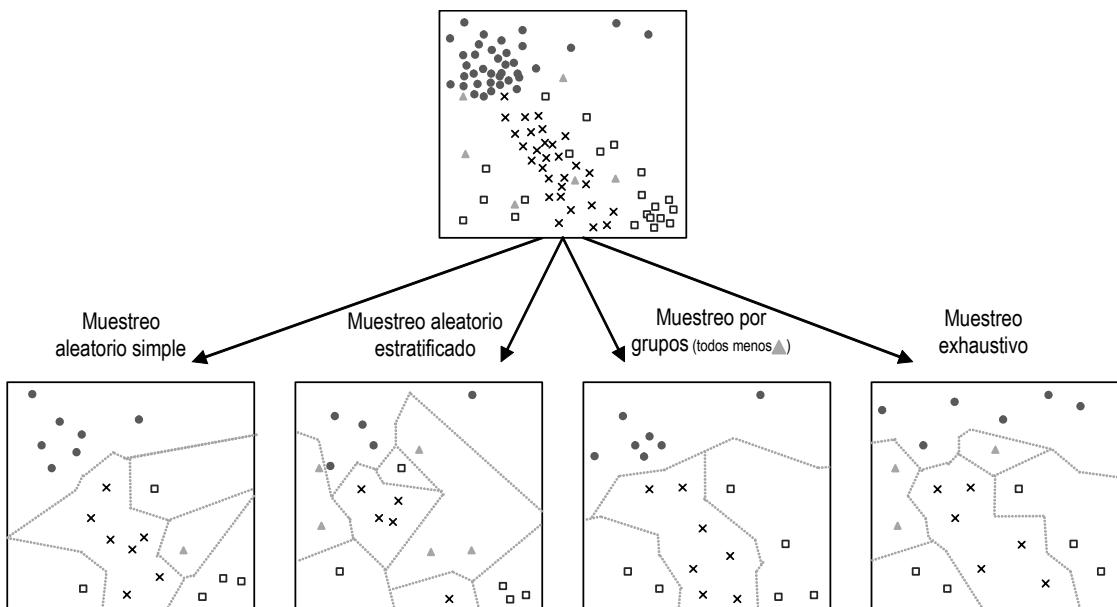


Figura 5.9. Representación de diferentes tipos de muestreo.

A veces se pueden realizar varios muestreos en cascada, el primero, por ejemplo aleatorio simple, para poder analizar mejor los datos y determinar grupos y, el segundo, estratificado, dependiendo del resultado anterior.

En algunos de los casos anteriores hemos hablado de que se puede realizar sobremuestreo, es decir “duplicación” de algunos ejemplos. Este concepto, en el que una misma instancia puede repetirse, admite muchísimas aplicaciones, además de nivelar las distribuciones de clases (balanceo), y muchísimas variantes. Por ejemplo, se pueden realizar copias que no sean exactamente iguales, como puede ser introduciendo un poco de ruido (errores aleatorios), para facilitar la generalización, o, directamente, generar ejemplos aleatorios utilizando la distribución de los atributos observados en los datos reales. En el caso de problemas de clasificación, se puede optar por generarlos etiquetados o no etiquetados. Este tipo de técnicas se utilizan justamente en el caso contrario del que hemos comentado antes, cuando hay *pocos* datos. Otro uso cada día más popular del muestreo es el realizar muestras dispares para obtener modelos diferentes y después combinarlos. El objetivo es bien diferente, obtener muestras que varíen significativamente entre sí. Por tanto, estas técnicas (como el *bootstrapping* usado para la técnica *bagging*) se verán en el Capítulo 18.

Finalmente, vistos los tipos de muestreo existentes, nos queda la pregunta: ¿cuántos datos son necesarios? En el caso de validación de hipótesis (cuando el muestreo se utiliza para obtener el conjunto de prueba) existen técnicas estadísticas para determinar cuántos

datos son necesarios para tener un nivel de confianza preestablecido (algunas, basadas en la distribución binomial o en la aproximación por la normal, se verán en el Capítulo 17). No obstante, si el objetivo es determinar cuántos datos son necesarios para el entrenamiento de un modelo, la respuesta es más difícil. Depende, en general, del número de "grados de libertad"¹¹ que, a su vez, depende del número de atributos (y de los valores posibles de cada uno). Además, depende del método de aprendizaje y de su expresividad (por ejemplo una regresión lineal requiere muchos menos ejemplos que una red neuronal). En general, la tendencia en estos casos es utilizar un muestreo progresivo o incremental, en el que se va haciendo la muestra cada vez más grande (y diferente si es posible) hasta que se vea que los resultados no varían significativamente entre un modelo y otro.

Estimación de frecuencias con muestras pequeñas y de alta dimensionalidad

Un problema general que se produce especialmente si tenemos muchas dimensiones y pocos ejemplos es que ciertos valores agregados, como las frecuencias de aparición, se estiman mal. Por ejemplo, si tenemos los datos de ventas de productos de una empresa y queremos saber la proporción de sexos de los compradores por cada producto, podríamos tener algún producto con pocas ventas y obtener estimaciones dudosas. Digamos que en el último año se han realizado 100.000 ventas, de las cuales tenemos que 30.000 fueron realizadas por hombres y 70.000 por mujeres. Supongamos que nos fijamos en un nuevo producto que se acaba de lanzar y vemos que las ventas producidas, de momento, son sólo tres unidades, todas de compradores varones. Si estimamos la proporción de sexos de este producto tendremos que el 100 por cien son varones, que es cierto hasta ahora, pero cuya extrapolación será muy mala para las futuras ventas. El problema es que la muestra es muy pequeña. ¿Existe alguna manera de atenuar el problema?

En realidad este tipo de situaciones son muy frecuentes y, más aún, si se realizan muestreos o se analizan datos cruzando unas dimensiones con otras. Para evitar este problema, existen estimadores de frecuencias basados en suavizados (*smoothing*). El más conocido es el **estimador** basado en la ley de la sucesión **de Laplace**, que se define de la siguiente manera:

$$f_{\text{Laplace}}(s) = \frac{n(s_f) + 1}{n(s) + alt}$$

La estimación de la frecuencia vendrá dada por los casos favorables + 1 (es decir $n(s_f) + 1$) dividido por los casos totales más el número de posibilidades o alternativas (es decir $n(s) + alt$). Si aplicamos el suavizado de Laplace al caso anterior, tenemos que la frecuencia estimada será de:

$$f_{\text{Laplace}}(\text{varón}) = \frac{3+1}{3+2} = \frac{4}{5}, \quad f_{\text{Laplace}}(\text{mujer}) = \frac{0+1}{3+2} = \frac{1}{5}$$

que es diferente del 100 por cien y del cero por ciento que teníamos sin corrección.

Cuando el valor de la muestra es grande, la corrección de Laplace no afecta prácticamente, como es de esperar, ya que si tuviéramos muchos casos, en la frecuencia sin

¹¹ Los grados de libertad representan, aproximadamente, cuantas cosas pueden hacer cambiar el estado o salida de un sistema. Cuantas más haya, es más probable que algún patrón *falso*, por mera casualidad, pueda conectar la salida con algunas de las entradas.

corrección se puede suponer una buena estimación. Cuando el valor es pequeño, como en este caso, se corrige la estimación acercándose ligeramente a una distribución a priori, supuesta ésta uniforme (50 por ciento para hombres y para mujeres).

La corrección de Laplace asume una distribución a priori uniforme, y no es muy ajustable en el caso de que queramos suavizar más o menos. El ***m*-estimado** es una generalización de la corrección de Laplace que permite mayor flexibilidad. Supongamos que tenemos una frecuencia a priori para el evento $f_{PRIORI}(s)$. El *m*-estimado se define de la siguiente manera:

$$f(s) = \frac{n(s_f) + m \cdot f_{PRIORI}(s)}{n(s) + m}$$

La estimación de la frecuencia vendrá ahora dada por los casos favorables más una constante m multiplicada por la frecuencia a priori del evento dividido por los casos totales $+ m$. Si aplicamos el *m*-estimado al caso anterior, en primer lugar debemos obtener la $f_{PRIORI}(s)$ que, según todos los productos, es de $f_{PRIORI}(\text{varón}) = 30$ por ciento y $f_{PRIORI}(\text{mujer}) = 70$ por ciento. Por tanto, usando $m=2$, tenemos que la frecuencia estimada será de:

$$f_{m\text{-estimado}}(\text{varón}) = \frac{3 + 2 \frac{1}{3}}{3 + 2} = 0,733, \quad f_{m\text{-estimado}}(\text{mujer}) = \frac{0 + 2 \frac{2}{3}}{3 + 2} = 0,267$$

Como vemos, los resultados pueden variar según la frecuencia o distribución a priori y el valor de m . A mayor valor de m tenemos un suavizado mayor. En general se suele escoger un valor de m entre 2 y 10.

La relación entre el *m*-estimado y el suavizado de Laplace es clara. El suavizado de Laplace es un caso particular del *m*-estimado cuando m es igual al número de alternativas o posibilidades y la distribución original es uniforme.

Los suavizados son una de las primeras técnicas de estimación, ya que permiten extrapolar frecuencias futuras y, en cierto modo, predecir frecuencias de sucesos. Estos suavizados permiten corregir patrones que podrían verse por tener unas muestras muy pequeñas. El uso de estos estimadores suavizados es fundamental en métodos bayesianos, como veremos en el Capítulo 10, ya que se realizan productos de muchas probabilidades y los suavizados anteriores evitan que ninguna de las probabilidades sea cero, ya que siempre hay un valor mayor que cero en el numerador.

5.4.2 Selección de características relevantes. Reducción de dimensionalidad

La selección de características, variables o atributos (*feature selection*, en inglés) tiene cuatro objetivos principales. En primer lugar, permite reducir el tamaño de los datos, al eliminar características o atributos de todos los ejemplos que puedan ser irrelevantes o redundantes. En segundo lugar, una buena selección de características puede mejorar la calidad del modelo, al permitir al método de minería de datos centrarse en las características relevantes. En tercer lugar, una buena selección de características permite expresar el modelo resultante en función de menos variables; esto es especialmente importante cuando se desean modelos comprensibles (árboles de decisión, regresión lineal, etc.). En cuarto lugar, se puede requerir una reducción de dimensionalidad a dos o tres características

exclusivamente, con el propósito de representar los datos visualmente. Existen otras razones para eliminar atributos, por ejemplo, cuando existen muchos datos erróneos o faltantes en un atributo, y es preferible deshacerse de él. También es necesario para realizar análisis de componentes principales (visto en el capítulo anterior), porque si no eliminamos atributos identificadores estos pesarán más que ningún otro.

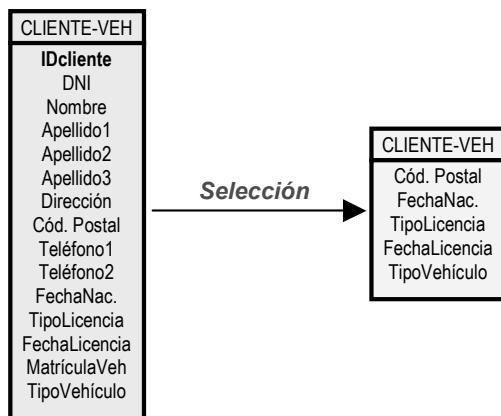


Figura 5.10. Selección de características (atributos).

En principio, si disponemos de un problema con muchos atributos puede parecer que siempre será mejor que cuando tenemos pocos atributos. Por ejemplo, cuanta más información tengamos de cada cliente puede parecer que podremos hacer mejores decisiones sobre las ofertas o productos que se le puedan dirigir. El problema es que la gran mayoría de métodos de minería de datos pueden *perderse* entre tantas características en un espacio que al tener alta dimensionalidad resulta estar más deserto (especialmente si las hay irrelevantes, redundantes o con valores erróneos) y obtener modelos que se ajustan a particularidades de los datos de entrenamiento y no de los datos en general. Esto ocurre especialmente cuando existen muchas dimensiones pero no tenemos un número suficiente de ejemplos para *reducir* los "grados de libertad".

Algunos atributos son fáciles de eliminar, por ejemplo si un atributo es constante, es decir, tiene el mismo valor para todas las instancias es claramente eliminable. En general, sin embargo, no es tan sencillo. Empecemos por los casos relativamente simples.

Considérese, por ejemplo, el caso de una compañía de seguros que intenta clasificar entre clientes aptos para un tipo de seguro de vehículo y clientes no aptos. Si en los datos de entrenamiento tenemos el identificador personal como atributo (por ejemplo, la clave primaria de la tabla, el número del documento de identidad o de la licencia de conducción, el pasaporte, etc.), algunos métodos de minería de datos pueden obtener reglas del estilo: *si el identificador = "12345678" entonces "apto"*. Lógicamente, esta regla sirve para "cubrir" los datos de entrenamiento, pero es completamente inútil para nuevos datos, ya que no habrá otro cliente futuro (aparte de él mismo), sobre el cual queramos utilizar el modelo, que tenga el mismo identificador. Dicho de otra manera, el método de aprendizaje ha sido *engaño* y el modelo se sobreajusta o se especializa para los datos de entrenamiento, pero no tiene ningún poder predictivo para otros datos. Debemos, por tanto, eliminar este identificador antes del entrenamiento para permitir que el método de minería de datos

obtenga otras reglas que se basen en la edad, en el sexo, en el número de años de licencia de conducir, etc.

Existen dos reglas generales para eliminar características, en especial los atributos nominales, que resultarán familiares a aquellos que conozcan la tecnología y terminología de bases de datos:

- **Eliminación de (partes de) claves candidatas:** la regla general es eliminar cualquier atributo que pueda ser clave primaria de la tabla (o que sea clave candidata o incluso parte de clave candidata, parcial o totalmente). Por ejemplo, hay que eliminar números de documentos de identificación, códigos internos, nombre y apellidos, direcciones (exceptuando, quizás, los códigos postales), teléfonos, e incluso fechas (si éstas no son relevantes). Para aquellos que no son familiares con la tecnología de bases de datos, una manera sencilla de saber si un atributo nominal es demasiado específico y debe ser eliminado es ver si tiene casi tantos valores como ejemplos. Si no se elimina este tipo de atributos puede ser especialmente problemático para tareas de clasificación o de regresión.
- **Eliminación de atributos dependientes:** en la teoría de la normalización de bases de datos, cuando existen dependencias funcionales entre atributos se intenta normalizar en varias tablas. Por ejemplo, cuando se tiene el código postal, la ciudad, la región y el país de un individuo, sabemos que con el código postal tenemos la ciudad y la región, con la región tenemos el país, con lo que podemos establecer una serie de dependencias funcionales que se deben normalizar en cuantas tablas sean necesarias. Dado que los datos con los que trabajamos para minería de datos pueden provenir de una “vista minable” que ha desnormalizado los datos, muchas veces debemos decidir si realmente necesitamos todos los atributos, ya que muchos de ellos son redundantes. En este caso, una buena opción puede ser mantener sólo el código postal o, si el nivel de detalle es excesivo, puede interesar mantener sólo la ciudad o la región. No eliminar los atributos dependientes es especialmente nefasto a la hora de establecer reglas de asociación o para las tareas de agrupamiento.

Aunque determinar los atributos anteriores (claves candidatas y atributos dependientes) no es sencillo en muchos casos, es mucho más fácil que determinar otras características que tampoco son relevantes o son redundantes. Por ejemplo, en el caso de la compañía de seguros podemos pensar que la procedencia del cliente puede ser irrelevante y luego no serlo, o puede serlo para algunas tareas de minería de datos pero no para otras. En general, detectados los atributos irrelevantes y redundantes, debemos utilizar técnicas más sofisticadas si queremos seguir reduciendo la dimensionalidad, en particular para los atributos numéricos. Existen dos tipos generales de métodos para seleccionar características.

- **Métodos de filtro** o métodos previos: se filtran los atributos irrelevantes antes de cualquier proceso de minería de datos y, en cierto modo, independiente de él. Las técnicas, como veremos, son fundamentalmente estadísticas (medidas de información, distancia, dependencia o inconsistencia). El criterio para establecer el subconjunto de características “óptimo” se basa en medidas de calidad previa que se calculan a partir de los datos mismos.

- **Métodos basados en modelo** o métodos de envolvente (*wrapper*): la bondad de la selección de atributos se evalúa respecto a la calidad de un modelo de minería de datos o estadístico extraído a partir de los datos (utilizando, lógicamente, algún buen método de validación). Lógicamente, este tipo de técnicas requieren mucho más tiempo que las otras, ya que para evaluar hay que entrenar un modelo. Nótese que el método de minería de datos utilizado para hacer la selección de atributos no tiene por qué ser el método de minería de datos que se utilizará finalmente. Por ejemplo, se puede utilizar una red neuronal para determinar los atributos más significativos y, una vez determinados, utilizar un árbol de decisión para obtener un modelo comprensible utilizando sólo las características seleccionadas para generar las reglas del modelo. También se puede utilizar directamente un árbol de decisión y examinar qué atributos no se utilizan o se utilizan en menos reglas (y menos importantes). También se puede analizar sólo la partición superior y eliminar los atributos cuyas particiones produzcan menos ganancia de información o la medida que se utilice generalmente como criterio de partición en el árbol de decisión (se verá en el Capítulo 11).

Sean de filtro o basados en modelo, ambos métodos pueden ser iterativos, es decir, se van eliminando atributos y se va observando el resultado (sobre la medida de calidad previa o la medida de calidad del modelo). Se van recuperando o eliminando más atributos de una manera iterativa, hasta que se obtiene una combinación que maximiza la calidad. Existen muchas maneras de realizar este procedimiento, por ejemplo, empezando con un atributo y elegir el que dé mayor calidad de la selección con atributos, después añadir el atributo que dé mayor calidad de selección con dos atributos, y así hasta que no se mejore la calidad o se llegue al número deseado de atributos (estrategia *forward*). La manera inversa (comenzar con todos e ir eliminando) o maneras mixtas también se pueden utilizar. Otra manera, más simple pero más costosa, consiste en realizar selecciones aleatorias de atributos, hasta que se encuentra una selección satisfactoria (estrategia *backward*). El objetivo en los dos casos es el mismo, obtener un subconjunto de atributos representativo (que contenga la mayor parte de la información que existía en el conjunto inicial) y que no exista la menor correlación entre los atributos seleccionados.

Veamos brevemente algunos métodos clásicos que se utilizan para establecer la independencia o importancia de las variables originales. En algunos casos hablaremos de que las técnicas se aplican para valores nominales y en otros para valores numéricos, aunque, en realidad, la mayoría de técnicas para valores nominales también se pueden utilizar para valores numéricos utilizando discretización y viceversa, usando numerización (como vimos en el capítulo anterior). Cuando existen variables numéricas y nominales, lo más normal es convertirlas todas a numéricas (después de un primer filtro de las variables nominales que, como hemos visto, es más sencillo).

La primera técnica que vamos a ver consiste en realizar una matriz de correlaciones, conocida también como **análisis correlacional**. Considérese, por ejemplo, el caso en el que tenemos donantes de sangre en un hospital con los siguientes atributos numéricos: edad, tensión (sanguínea), (índice de) obesidad, colesterol, (nivel de) tabaquismo, (nivel de) alcoholismo, pulsaciones (por minuto en reposo) y hierro (en sangre).

En la Tabla 5.1 se muestran las correlaciones entre los atributos para una población de 800 individuos (la matriz es simétrica, lógicamente, ya que la correlación también lo es):

Esta información de pacientes sanos puede utilizarse para ver qué atributos están más estrechamente asociados y qué atributos parecen ser más independientes. En este caso, podemos observar que los dos atributos más correlacionados son obesidad y tabaquismo, tensión y tabaquismo, así como colesterol y obesidad. En algún caso se podría pensar en utilizar aquellos que sean más fáciles de obtener con fiabilidad (por ejemplo, tensión y obesidad) y eliminar el tabaquismo para obtener otros modelos.

ATRIBUTO	Edad	Tensión	Obes.	Colest.	Tabaq.	Alcohol.	Puls.	Hierro
Edad		0,63	0,34	0,42	-0,02	0,15	0,12	-0,33
Tensión	0,63		0,22	0,56	0,72	0,43	0,27	-0,08
Obesidad	0,34	0,22		0,67	0,72	0,32	0,32	0,21
Colesterol	0,42	0,56	0,67		0,52	0,27	0,40	0,45
Tabaquismo	-0,02	0,72	0,72	0,52		0,58	0,39	-0,12
Alcoholismo	0,15	0,43	0,32	0,27	0,58		0,23	-0,22
Pulsaciones	0,12	0,27	0,32	0,40	0,39	0,23		-0,15
Hierro	-0,33	-0,08	0,21	0,45	-0,12	-0,22	-0,15	

Tabla 5.1. Matriz de correlaciones.

También se observan atributos aparentemente independientes, como el tabaquismo y la edad. Esto en ningún caso sugiere que si se desea predecir el tabaquismo, la edad se ha de eliminar (la relación puede existir a distintos segmentos de edad o puede tener forma de campana). La información anterior es suficientemente ilustrativa para ayudar en la selección, aunque siempre hay que realizarlo de manera cautelosa. En realidad, un estudio correlacional permite, además de ayudar en la selección, comprender los datos y, por tanto, forma parte de la exploración de datos que estamos tratando en este capítulo.

El siguiente paso en sofisticación en el caso de atributos numéricos es pasar a una técnica *basada en modelo* (y no *de filtro*, como la anterior), en concreto un **análisis por modelo lineal**. Si el objetivo fuera predecir el azúcar en sangre a partir de los atributos anteriores, podríamos realizar una regresión lineal, es decir una función del estilo:

$$y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n$$

donde las x_i son los atributos originales y los a_i son los coeficientes estimados (veremos en detalle la regresión lineal y no lineal en los capítulos 7 y 8, de hecho el método es igualmente válido para regresiones no lineales). Por ejemplo, podríamos tener los siguientes coeficientes (suponiendo que hemos normalizado los valores anteriormente en una escala de 0 a 1):

Atributo	Edad	Tensión	Obes.	Colest.	Tabaq.	Alcohol.	Puls.	Hierro
Azúcar	2,23	-1,63	3,23	0,42	-0,12	2,23	0,00	-3,01

Podemos ver que el atributo que más influye en el modelo es la obesidad y el que menos (en realidad, nada) el número de pulsaciones. Al ser un modelo conjunto, las variables dependientes se compensan unas con otras y aunque el número de pulsaciones pudiera correlacionar positivamente con el azúcar, el modelo lineal destaca que la influencia se puede extraer de otras variables. Esto, en realidad, es sólo el principio de múltiples técnicas del análisis multivariante (*multivariate analysis*). Por ejemplo, si quisieramos saber si el colesterol, el tabaquismo y las pulsaciones no influyen en el azúcar y, son, por tanto, descartables, deberíamos usar, por ejemplo, el Análisis de la Varianza (conocido

popularmente como ANOVA y que puede encontrarse en muchísimos libros de texto [Glantz & Slinker 2000; Peña 2002b]).

Una forma diferente de ver el problema, en especial cuando los atributos son nominales, es realizar un análisis de frecuencias, es decir, ver para cada combinación de valores de atributos cuántos casos hay¹². Por ejemplo, supongamos una empresa de vigilancia que tiene ciertos sensores instalados en un comercio, registrando como atributos: "luz", con valores {oscuridad, penumbra, claridad}, "horario", con valores {abierto, cerrado}, "presencia zona 1", con valores {sí, no}, "presencia zona 2", con valores {sí, no} y "ruido", con valores {nada, poco, bastante, mucho}. Los valores de los atributos se registran cada diez segundos, y dan como resultado 8.640 ejemplos diarios, muchos de los cuales, lógicamente, estarán repetidos. Podemos representar los 60.480 ejemplos de toda una semana en una tabla de frecuencias multidimensional:

		Pr. 1	sí						no									
			sí			no			sí			no						
		Ruido	n	p	b	m	n	p	b	m	n	p	b	m	n	p	b	m
Horario	abierto	oscuridad	3	22	173	84	12	77	41	33	5	16	70	30	20	2	35	13
		penumbra	34	522	1253	227	41	222	533	227	32	532	330	134	4	229	251	53
		claridad	137	1623	2357	5262	137	123	325	1062	30	634	631	732	25	103	225	362
	cerrado	oscuridad	31	7	2	0	131	87	32	5	2031	537	178	36	23892	1284	332	95
		penumbra	63	22	7	1	223	5	0	0	823	325	77	22	3035	1852	210	43
		claridad	171	124	163	122	136	124	67	12	271	341	478	327	771	924	1267	766

En la tabla anterior, se pueden observar algunas combinaciones mucho más frecuentes que otras y qué combinaciones son más frecuentes y, sobre todo, cuáles son más improbables (quizá son éstas las que se desea estudiar para detectar problemas de seguridad). También podemos agregar teniendo en cuenta sólo un par de atributos (esto sería similar a utilizar los operadores OLAP *slice* y *roll-across*, vistos en el capítulo anterior):

		Luz		
		oscuridad	penumbra	claridad
Pres. 1	sí	740	3380	11945
	no	28576	7952	7887

Cuando nos centramos en sólo dos variables, esta tabla de frecuencias se denomina matriz de incidencias o tablas de contingencia bidimensionales (*two-way contingency tables*). Dicha tabla también se puede representar gráficamente con un histograma, como los vistos en el capítulo anterior. Los análisis que parten de este tipo de matriz se denominan **análisis de correspondencias** (*correspondence analysis*), que nuevamente se engloba, junto otras técnicas, dentro de lo que se denomina análisis multivariante (*multivariate analysis*), por tener una base algebraica, como algunas técnicas vistas en el capítulo anterior (análisis de componentes principales). El análisis de correspondencias se basa en realizar tests ji-

¹² Si tenemos en cuenta todos los atributos, en general el número de instancias por combinación es 0 o 1, suponiendo que no hay ejemplos repetidos. La cosa cambia cuando consideramos un subconjunto de atributos o es un dominio donde se realiza medidas muy frecuentes y pueden aparecer datos repetidos.

cuadrado¹³ (χ^2) para saber si el efecto de los valores de una variable es independiente de los valores de la otra. En general, se puede transformar la tabla anterior en una matriz de distancias ji-cuadrado, que permite, de un solo vistazo, ver qué valores influyen en otros y si esto es significativo. En el Capítulo 9 de reglas de asociación veremos un ejemplo sencillo del test ji-cuadrado para determinar reglas de dependencias. Más información sobre el análisis de correspondencias se puede obtener en [Greenacre 1984; Berthold & Hand 2003] o en libros de análisis multivariante, como [Peña 2002b; Johnson 1999; Krzanowski 1988]. Existen métodos basados también en el ji-cuadrado, como el método CHAID (*chi-square automatic interaction detection*), que, aunque, en realidad, es un sistema de construcción de árboles de decisión para clasificación y regresión, se utiliza, fundamentalmente, para detectar la interacción entre variables [Kass 1975].

En cierto modo este tipo de “correspondencias” se parecen a las reglas de asociación, a las que se dedica el Capítulo 9. No obstante, no hay que confundirlas. Por ejemplo, las asociaciones de valor no sirven, en general, para determinar qué variables son redundantes. En realidad, además de las medidas de soporte y precisión, podríamos ver si las asociaciones son significativas utilizando un análisis de correspondencias.

Siguiendo en sofisticación, los modelos lineales generalizados, en particular los modelos *log-linear*, también pueden utilizarse para analizar atributos nominales y, en particular, tablas de frecuencia. En concreto, a un modelo lineal, se asume que, además de una salida logarítmica, tenemos una distribución de Poisson sobre los elementos para cada combinación de elementos. Más formalmente,

$$y \approx \text{Poisson}(\mu) \quad \text{donde } \log(\mu) = a_0 + a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

La idea es que el modelo determina la cardinalidad o la frecuencia de cada casilla de la tabla de frecuencias multidimensional. Lo interesante no es predecir estos valores (que los sabemos) sino estimar la influencia de cada valor y factor en el modelo, con el objetivo de ver los atributos y pares de atributos que son más relevantes. Para ello se utiliza una medida denominada “desviación” (*deviance*). Los atributos o interacciones de atributos con mayor desviación se preservan. El resto se descartan. Una idea similar es el uso de otros modelos lineales generalizados, como por ejemplo un modelo logístico. Para más información sobre el uso de modelos lineales generalizados para la selección de atributos, se puede consultar [McCullagh & Nelder 1989]. Además, en la Sección 7.7 trataremos los modelos lineales generalizados.

Existen muchísimos otros métodos para realizar selección de atributos, y algunos de ellos los comentaremos brevemente junto con algunas técnicas de la Parte III (como por ejemplo ciertas técnicas evolutivas que se comentarán en el Capítulo 15 (Sección 15.3.3). Muestra de esta variedad es el ejemplo que veremos a continuación en el que una herramienta de minería de datos puede tener decenas de métodos de selección. Un buen texto de referencia sobre selección de atributos es [Liu & Motoda 1998b]. Respecto a la automatización de este análisis exploratorio de datos, se puede consultar [Becher et al. 2000].

¹³ Ji-cuadrado es el nombre correcto de esta distribución, por la pronunciación de la letra griega χ . No obstante, cada vez se utiliza más frecuentemente el término “Chi-cuadrado” para referirse a esta distribución, probablemente por influencia del inglés.

Para terminar, hay que resaltar que la característica principal y la gran ventaja de los métodos de reducción de dimensionalidad mediante selección de características (a diferencia de los basados en transformación o sustitución de unos por otros, como el análisis de componentes principales) es que el conjunto final de características es un subconjunto de los atributos originales y, por tanto, los modelos extraídos con posterioridad se definirán en función de (parte de) los atributos originales del problema y, por tanto, no perderán comprensibilidad.

5.4.3 Ejemplo de selección de atributos

Vamos a realizar un ejemplo de selección de atributos utilizando la herramienta WEKA (véase el Apéndice A). Vamos a utilizar para ello el conjunto de datos "labor" (véase el Apéndice B), que recoge información (16 atributos: ocho nominales y ocho numéricos) sobre acuerdos sindicales, indicando si éstos fueron buenos o malos (la clase). El conjunto sólo recoge 57 instancias, por lo que, para poder extraer un modelo fiable, sería interesante realizar una selección de atributos previa para aumentar la proporción entre número de ejemplos y número de atributos. En este ejemplo vamos a analizar la relevancia respecto a la *clase*, es decir vamos a intentar identificar un subconjunto de atributos que estén fuertemente relacionados con la clase (y por tanto ayuden a su predicción) y, además, estén poco relacionados entre sí.

El sistema WEKA incorpora una gran cantidad de métodos para estudiar la relevancia de atributos y realizar una selección automática de los mismos. Para utilizar estos métodos, una vez cargados los datos, es necesario elegir uno de estos métodos de selección (en el entorno "Explorer", se encuentran en la pestaña "Select Attributes"). Existen una serie de evaluadores, entre los que, además de los componentes principales, hay dos tipos: "SubSetEval" (evaluadores de subconjuntos o selectores) y "AttributeEval" (prorrataedores de atributos). Los primeros necesitan elegir un método o estrategia de búsqueda de los subconjuntos ("Search Method"), como los que hemos visto antes (*forward* o *backward*), además de una estrategia exhaustiva (prueba todos los subconjuntos posibles) y otras. Los segundos sólo pueden combinarse con un "Ranker", ya que no seleccionan atributos sino que los ordenan por relevancia.

Veamos cómo se comportan distintos métodos de selección con este conjunto de datos. La siguiente tabla muestra una serie de métodos y los atributos seleccionados por ellos:

Método de selección	Filtro/Modelo	Opciones	Estrategia de búsqueda	Atributos seleccionados
CfsSubSetEval	Filtro	Valores por defecto	"ExhaustiveSearch"	2,3,11,13
WrapperSubSetEval	Modelo	Modelo j4.8.j4.8, valores por defecto	"ExhaustiveSearch"	Muy lento
			"ForwardSelection"	2,11
			"BestFirst"	2,11
			"GeneticSearch"	3,6,7,13
			"RankSearch"	2,3,11
		Modelo LogisticRegression, valores por defecto	"ForwardSelection"	2,6,10,13
			"GeneticSearch"	3,6,8,10,13,14
			"RankSearch"	2,3,11,13
ConsistentSubSetEval	Filtro	Valores por defecto	"ExhaustiveSearch"	2,3,10,16

Tabla 5.2. Selecciones realizadas con WEKA con métodos "Search".

Como se puede observar, se han utilizado métodos de filtro y métodos que utilizan modelo (envolventes, *wrappers*). Entre estos métodos de modelo, hemos elegido como modelo el J4.8 (la versión del WEKA del clasificador por árboles de decisión C4.5, véase el Capítulo 11) y la clasificación por regresión logística (véase el Capítulo 7).

Para analizar mejor las selecciones anteriores vamos a utilizar ahora prorrataedores de atributos (para todos ellos hay que elegir “Ranker” como estrategia de búsqueda). En la Tabla 5.3 se muestra el orden de los 16 atributos según varios métodos de identificación de la relevancia de atributos disponibles en WEKA.

A la vista de las dos tablas tenemos bastante información para tomar una decisión a la hora de seleccionar los atributos más relevantes. Los atributos 2 y 3, que representan los incrementos salariales en los dos primeros años de contrato, son los dos atributos más relevantes para llegar a un buen acuerdo, ya que aparecen en casi todas las selecciones y aparecen en primer lugar en la mayoría de los *rankings*.

Método de evaluación	Filtro / Modelo	Estrategia de búsqueda	Orden de los atributos
RelieFAttributeEval	Filtro	“Ranker”	14,2,3,12,16,4,7,13,11,9,8,6,10,5,15,1
InfoGainAttributeEval	Filtro	“Ranker”	2,3,11,14,16,12,13,9,7,5,15,4,10,8,1,6
GainRatioAttributeEval	Filtro	“Ranker”	2,3,11,13,14,16,9,15,12,7,5,4,10,8,1,6
SymUncertAttributeEval	Filtro	“Ranker”	2,3,11,14,16,13,12,9,15,7,5,4,10,8,1,6
OneRAttributeEval	Filtro	“Ranker”	13,11,16,7,10,2,3,5,14,9,15,6,12,4,1,8
ChiSquaredAttributeEval	Filtro	“Ranker”	2,3,11,14,16,12,13,9,7,5,15,4,10,8,1,6
WrapperSubSetEval	Modelo j4.8.j4.8	“ForwardSelection” con generateRanking=“true”	2,11,1,10,8,4,5,3,15,12,6,9,13,7,16,14
WrapperSubSetEval	Modelo Logistic	“ForwardSelection” con generateRanking=“true”	2,13,10,6,15,5,4,3,7,9,14,16,1,8,11,12

Tabla 5.3. Selecciones realizadas con WEKA con métodos “Ranker”.

Siguiendo con los atributos, el atributo 11 suele también aparecer seleccionado o bien colocado por los distintos métodos anteriores. No es difícil ver, tampoco, que cuando el atributo 11 (“vacaciones”) aparece mal colocado, el atributo 10 (“facilidades de formación”) aparece muchas veces en su lugar, sugiriendo que ambos están relacionados. Para verlo podríamos estudiar ambos atributos separadamente. Para ello, como el 10 es nominal y el 11 es numérico, deberíamos o bien convertir el 10 a numéricos (y estudiar la correlación) o bien convertir el 11 a nominal (y estudiar las frecuencias conjuntas). Teniendo en cuenta estas consideraciones, podríamos seguir mirando para determinar qué subconjunto nos interesa más. En cualquier caso siempre es preferible elegir una de las selecciones ya hechas que realizar esta selección manualmente, por ejemplo la realizada por el método CfsSubSetEval (2,3,11,13), que incluye los atributos 2, 3 y 11. Una selección hecha por algún método asegura poca relación entre los atributos elegidos.

5.5 Lenguajes, primitivas e interfaces de minería de datos

A lo largo de los dos últimos capítulos y, en especial, en este último, hemos visto que podemos realizar una serie de exploraciones, transformaciones, vínculos, modificaciones, selecciones, etc., con el objetivo de obtener una vista minable, acompañada de una tarea a realizar y un método para llevarla a cabo, una evaluación de calidad, un conocimiento previo y una manera de presentar los patrones obtenidos, como vimos en la Figura 5.1.

Todo ello es necesario para realizar el proceso de minería de datos. La pregunta que surge es la siguiente: ¿cómo especificamos este conjunto de elementos? Para ello, podemos utilizar tres tipos de medios para especificar los componentes: lenguajes de consulta, conjunto de primitivas o interfaces integrados.

- Lenguajes de consulta: llamados también lenguajes de consulta inductivos o de minería de datos, permite enfocar el proceso de minería de datos de una manera similar al proceso de consulta de una base de datos. Ya sea en una o más instrucciones del lenguaje, estas consultas obtienen “modelos” o conjuntos de reglas a partir de los datos, siguiendo las especificaciones establecidas en las consultas.
- Conjuntos de primitivas o interfaces *middleware*: en vez de proporcionar un lenguaje, se proporciona una serie de primitivas que, junto a un lenguaje de programación (C++, Java, Python, etc.), permiten especificar los componentes o realizar todos los pasos previos a la minería de datos y la minería de datos propiamente dicha. Generalmente este conjunto de primitivas se organizan en interfaces de programación de aplicaciones (API, *Application Programming Interface*). Otro tipo de interfaces son aquellos que permiten interrelacionar servidores OLAP con aplicaciones que realicen análisis a través de un modelo cliente/servidor.
- Interfaces o entornos integrados visuales: basados en la idea de flujo de datos/información/conocimiento, presentan una serie de nodos que tienen una serie de entradas y una serie de salidas, lo que permite interconectarlos. Esto permite ver todo el proceso como un flujo que se origina en la información y que termina en los patrones o en su evaluación. Las herramientas gráficas, de selección, de transformación, de modelado, de evaluación, etc., son “nodos” del sistema.

Aunque no existe una distinción estricta entre ellos (de hecho existen interfaces integradas que tienen lenguajes de macros que están basados en primitivas) vamos a describir las tres aproximaciones separadamente.

La gran ventaja de las dos primeras aproximaciones es que permiten definir estándares (sobre los cuales se puedan realizar interfaces). Como veremos a continuación, es sorprendente que exista más consenso en las interfaces y entornos visuales (el parecido entre ellos es muy alta) que entre los lenguajes y primitivas (cuya disparidad es alta, como veremos).

5.5.1 Lenguajes de consulta de minería de datos

Cuando hablamos de vista minable, nos puede parecer que podríamos utilizar un lenguaje de consulta como el SQL y alguna herramienta “reports” para realizar estas vistas minables. Presentadas las transformaciones y la exploración requerida, podríamos considerar utilizar herramientas OLAP, ya que permiten transformar y agregar los datos de una manera más flexible, cómoda y, sobre todo, eficiente. Si bien todo esto es cierto, estos lenguajes o herramientas no son suficientes para especificar todos los elementos vistos en la Figura 5.1.

Con el objetivo de especificar no sólo la vista minable, sino además todos los elementos asociados, han aparecido una serie de lenguajes para la minería de datos. Algunos de estos lenguajes ven el descubrimiento de conocimiento en bases de datos como un *proceso de consulta a una base de datos* [Imielinski & Manilla 1996]. Dependiendo del lenguaje se especifican unas cosas u otras, aunque todos los lenguajes, al menos, permiten especificar

la vista minable, el tipo de patrón o reglas a extraer y algunos criterios de evaluación a cumplir. Aunque la mayoría de lenguajes de consulta de minería de datos tienen sus orígenes a mitad de los años 90, diez años después, a la hora de escribir este texto, la situación es similar a la situación de múltiples propuestas que aconteció en los 60 y 70 hasta que el SQL se instauró poco a poco como el lenguaje de consultas estándar para bases de datos relacionales (y hoy en día objeto-relacionales).

Por tanto, veamos simplemente unas pinceladas de estos lenguajes en forma de ejemplos comentados. El lector puede seguir las fuentes y los enlaces para seguir el estado actual de estos lenguajes y de su estandarización.

Comencemos por la propuesta **M-SQL**, que se originó a mediados de los 90 [Imielinski et al. 1996]. Este lenguaje se centraba inicialmente en pocos tipos de patrones, básicamente reglas de asociación. En la Figura 5.11 se muestra que la vista minable se delega a unos paréntesis, donde se pone el nombre de una tabla o una vista (en el ejemplo con nombre "T"). Esto significa que la vista minable se construye con SQL estándar. Lo que sí se permite especificar en la consulta son restricciones sobre la forma del modelo (en el ejemplo que en el consecuente de las reglas de asociación debe aparecer el atributo "Age"). También se pueden especificar restricciones sobre la calidad o la evaluación del mismo. En este caso, que el soporte sea mayor que 1.000 (el número de instancias en las que la regla a extraer se aplica satisfactoriamente) y que la confianza sea mayor que 0,65 (que de las que se pueda aplicar, el 65 por ciento de las veces se aplique con éxito).



Figura 5.11. Ejemplo de consulta de minería de datos en el lenguaje M-SQL.

Se pueden indicar otras restricciones sobre el antecedente (*R.Body*).

Otra propuesta se denomina **DMQ o DMQL** (*Data Mining Query Language*), originalmente presentado en [Ng et al. 1998] y extendido en [Han & Kamber 2001]. Éste es un lenguaje mucho más completo y permite expresar la vista minable (especificando tablas e incluso base de datos de origen), el tipo de conocimiento a extraer (asociaciones, clasificación, etc.), las medidas de evaluación o de interés para determinar los patrones válidos y la manera de presentar el resultado. Entre los aspectos más destacables del lenguaje podemos citar el hecho de que se puede definir conocimiento previo en forma de jerarquías, para los procesos de generalización.



Figura 5.12. Ejemplo de consulta de minería de datos en el lenguaje DMQL.

En la Figura 5.12 se muestra una consulta para clasificar el riesgo de crédito a partir de dos atributos (ingresos y ocupación) de los clientes que hayan hecho compras y que tengan

entre 30 y 40 años. Para más información de esta propuesta, se puede consultar [Han & Kamber 2001].

Una de las propuestas que se ha hecho más popular (como no podía ser de otra manera, debido al proponente) es el “**OLE DB for Data Mining**” de Microsoft, que es, en realidad, una extensión del protocolo de acceso a bases de datos OLE DB. En realidad es una extensión del SQL de este protocolo para trabajar con modelos de minería de datos (DMM, *Data Mining Model*). El proceso se estructura en tres fases: crear un modelo vacío, entrenar el modelo y realizar predicciones.

La manera de crear un modelo vacío se ilustra en la Figura 5.13.

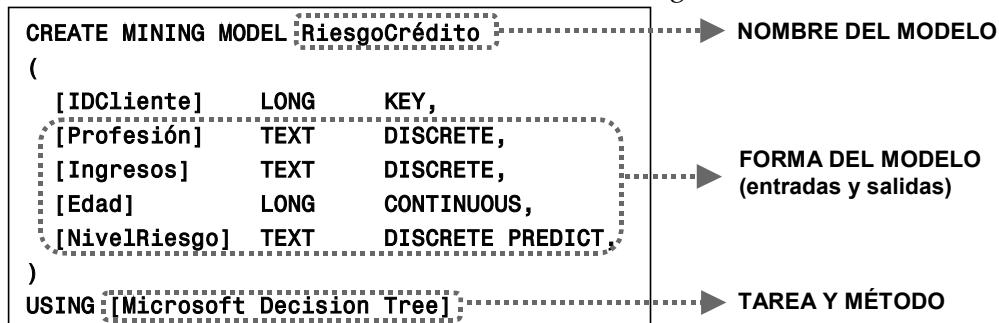


Figura 5.13. Creación de un modelo vacío en “OLE DB for Data Mining”.

En realidad, esto simplemente le da un nombre al DMM y cuáles van a ser sus entradas y sus salidas (la forma que va a tener). A continuación, lógicamente, hay que entrenar el modelo. Para ello, y de una manera bastante peculiar y original al mismo tiempo, se utiliza una variación de la sentencia `INSERT INTO` del SQL, como se ve en la Figura 5.14.

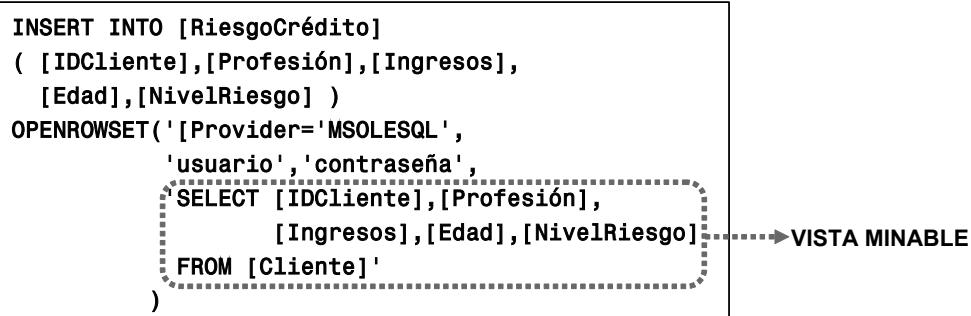


Figura 5.14. Entrenamiento de un modelo en “OLE DB for Data Mining”.

A diferencia del `INSERT` del SQL (que inserta datos en una tabla normal) lo que realmente esta instrucción hace es analizar los casos que le introduzcamos (la vista minable) y construir el contenido del DMM.

Finalmente, para usar el modelo, éste se aplica a nuevos datos. La manera de hacerlo es similar a la concatenación (junción) de dos tablas relacionales, considerando el modelo como una tabla y los datos a predecir como otra tabla. El resultado es una nueva tabla con los datos que queramos (todos o sólo las predicciones). Esta última parte se ilustra en la Figura 5.15.

```

SELECT [ClienteID], R.NivelRiesgo,
       PredictProbability(R.NivelRiesgo)
  FROM RiesgoCrédito R [PREDICTION JOIN ClienteNuevo C]
    ON R.Profesión=C.Profesión
   AND R.Ingresos=C.Ingresos
   AND R.Edad=C.Edad

```

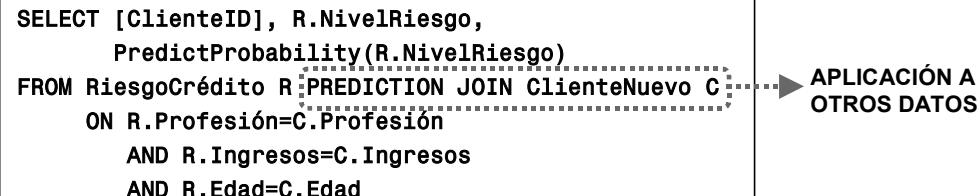


Figura 5.15. Uso de un modelo en "OLE DB for Data Mining".

Esta parte permite, por ejemplo, si en el SELECT añadimos C.NivelRiesgo, obtener una tabla con dos columnas, una para el valor predicho y otra para el valor real, con lo que podríamos utilizarla para validación.

Quizá la propuesta de Microsoft es una de las más difundidas, aunque existen algunas propuestas más recientes de estandarización.

Otra propuesta interesante es el estándar a incluir en el SQL sobre minería de datos, en concreto (**SQL/MM Part 6: Data Mining**). Esta propuesta, en estandarización por el ISO/IEC, define mediante tipos definidos por el usuario (UDTs, *User Defined Types*), varios modelos de minería de datos, parámetros para los algoritmos, modos de aplicar el modelo a una fila de una tabla, etc.

Veámoslo con un ejemplo; creamos en primer lugar la vista minable:

```

VistaMinable= NEW DM_MiningData("Cliente")
  References: table Cliente
  Field 1: name "Profesión", alias "A1", type: CAT
  Field 2: name "Ingresos", alias "A2", type: CAT
  Field 3: name "Edad", alias "A3", type: NUM

```

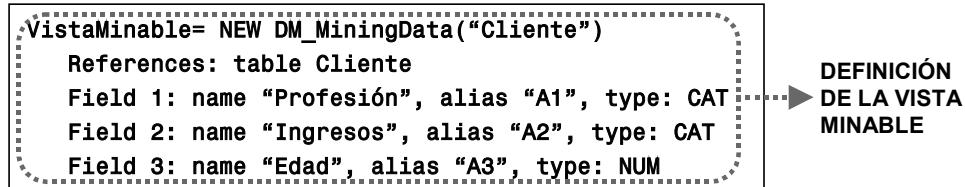


Figura 5.16. Definición de la vista minable con "SQL/MM Part 6: Data Mining".

La definición de "alias" viene motivada porque hay que realizar un *mapeo* entre esta vista minable y la tabla de destino (mediante DM_MiningMapping(...)), y los nombres deben coincidir. A continuación hemos de elegir entre cuatro tareas básicas: reglas de asociación (Rule), agrupamiento (Clus), clasificación (Clas) y regresión (Reg). Hemos de determinar también los parámetros de cada método. Por ejemplo, supongamos que tenemos definido un mapeo con nombre "Mapeo"; para definir y entrenar un modelo de reglas de asociación se haría de la siguiente manera:

```

Parametros= (((NEW DM_RuleSettings()
  .DM_ruleUseMapping(Mapeo))
  .DM_ruleSetGroup("A2"))
  ; DM_ruleSetMinSupport(0.05))

Tarea= (DM_defRuleTask (VistaMinable, Parametros))

Modelo= Tarea.DM_buildRuleModel()

```

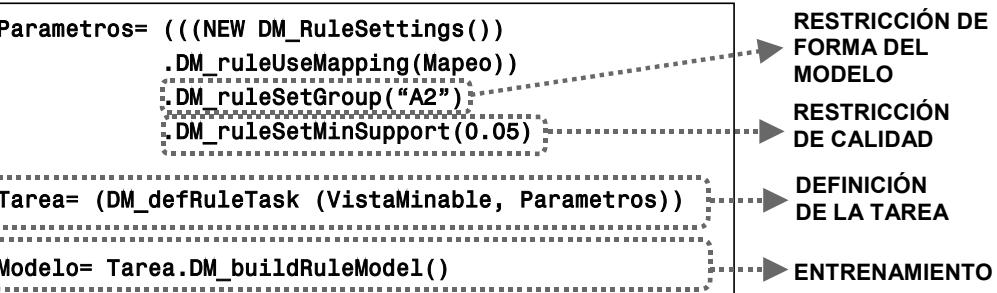


Figura 5.17. Definición de tarea y entrenamiento del modelo con "SQL/MM Part 6: Data Mining".

A partir de aquí, el modelo se puede exportar en formato textual (en el estándar PMML que veremos en el Capítulo 19) o aplicarlo a filas de otra tabla, mediante el mapeo

correspondiente y los objetos DM_ClasResult, DM_ClusResult o DM_RegResult. Los modelos “Clas” o “Reg” pueden exportar los resultados de una validación con un conjunto de datos, mediante el objeto DM_ClasTestResult o DM_RegTestResult.

Para acabar, hay que destacar que este estándar sólo es una especificación y no proporciona una implementación de estos métodos.

Los lenguajes anteriores se basan en decidir el tipo de patrón a extraer y, en cierto modo, se ajustan a ciertos patrones de consulta. Es decir, si un lenguaje no tiene definido el agrupamiento, no hay manera en el lenguaje de definir tal tarea. Existen otras propuestas más flexibles (también más complejas de entender y usar) en las que uno puede formalizar las tareas a realizar. Por ejemplo, la propuesta LDL+ [Shen et al. 1996] se basa en un lenguaje lógico de orden superior que permite expresar gran variedad de restricciones de reglas utilizando esquemas de orden superior. Los esquemas se llaman metaconsultas (*metaqueries*). Una propuesta todavía más abierta, aunque en cierto modo más difícil de manejar, es la propuesta RDM (*Relational Data Mining*) de Luc De Raedt [De Raedt 1998a]. Esta propuesta usa Prolog para definir patrones de muy diverso tipo.

Pese a la gran cantidad de lenguajes y propuestas que hemos visto, es importante enfatizar que la definición de lenguajes de consulta más flexibles (llamados lenguajes de consulta inductivos) que cubran y automaticen las tareas y fases de la extracción de conocimiento de bases de datos está todavía abierta.

5.5.2 Conjuntos de primitivas de minería de datos

Los lenguajes de consulta de minería de datos pueden utilizarse interactivamente o pueden utilizarse dentro de algún lenguaje de programación. Éste es el caso de, por ejemplo, el “OLE DB for Data Mining” o el SQL/MM. Si lo que deseamos es realizar minería de datos a través de una aplicación, puede ser preferible disponer de un conjunto de primitivas que se puedan utilizar, a forma de API, en nuestros programas. Ésta suele ser la elección más aconsejable y eficiente para programadores. Otras veces se necesitan interfaces entre aplicaciones y servidores, constituyendo estándares, en cierto modo, de lo que se ha venido llamando *middleware*.

Cuando hablamos de conjuntos de primitivas nos referimos a propuestas con un cierto nivel de abstracción y de portabilidad. Por ejemplo, existen librerías, como las MLC++, XELOPES o WEKA, que van creciendo con el tiempo, cuya especificación es prácticamente la de la implementación en C++ o Java, respectivamente. De este tipo de librerías hablaremos en el Apéndice A. En este punto vamos a tratar de propuestas de especificaciones de primitivas de minería de datos, que puedan ser utilizadas para realizar el proceso de extracción de conocimiento.

Para aclarar el concepto, vamos a empezar con el ejemplo más paradigmático. A partir de la versión 9i del sistema de gestión de bases de datos Oracle, se incluyó una API en Java para poder realizar minería de datos en las aplicaciones realizadas en Oracle, que permite a estas aplicaciones realizar ciertas tareas básicas de minería de datos con Oracle. Sin desvincularse completamente de Oracle, esta API sufrió posteriormente un proceso de estandarización a través de una *Java Specification Request 73* (JSR-73) con el nombre de *Java Data Mining* (JDM) que permite y promueve que otros fabricantes incluyan la misma API. Por eso se le conoce también como JDM API. JDM no es una propuesta cerrada, ya que tiene

una serie de características principales y una serie de paquetes opcionales que pueden ser incluidos o no por cada implementación. Para permitir flexibilidad y evitar problemas debido a este hecho, se puede saber en tiempo de ejecución si una determinada implementación tiene una cierta característica. Para ello existe el método “*supportsCapability*” que permite saber, en tiempo de ejecución, qué características están implementadas. En la primera versión del estándar se incluyeron las tareas de clasificación, regresión, asociación, agrupamiento, así como el análisis de importancia de atributos. También permite, además de construir los modelos, aplicarlos, validarlos, importarlos y exportarlos. En las siguientes versiones se consideran aspectos como la minería de datos no estructurados (textos e imágenes), *ensembles*, extracción de características y algunas otras.

Brevemente, comentemos otras propuestas. “*XML for Analysis*” (<http://xmla.org>) es un conjunto de interfaces de mensajes XML que utilizan el estándar SOAP (*Simple Object Access Protocol*) para definir el acceso e interacción con los datos entre una aplicación cliente y un proveedor de datos analítico (una herramienta OLAP o un servidor de minería de datos) que estén separados, por ejemplo, por Internet.

Una propuesta más centrada, de momento, en los almacenes de datos, es el *OMG Common Warehouse Metadata* (CWM) del *OMG* (*Object Management Group*, <http://www.omg.org>). Este estándar está definido en UML (*Unified Modeling Language*) y puede considerarse como una extensión del UML para modelar almacenes de datos. No obstante, aunque en principio puede parecer que es sólo un estándar de modelado en UML, su objetivo principal es permitir el intercambio de metadatos entre almacenes de datos y repositorios de metadatos o conocimiento previo con herramientas clientes, ya sean herramientas OLAP o herramientas de minería de datos. Para ello incorpora un estándar de intercambio, definido, cómo no, en XML, así como una interfaz de especificación para el acceso de distintos entornos. La importancia de cara a la minería de datos viene por la capacidad de utilizar este estándar para la definición de la vista minable.

Existen lenguajes para representación de modelos, como el PMML, que se tratará en el Capítulo 19, con bastante interrelación con los lenguajes e interfaces *OMG CWM*, *SQL/DM Part 6 DM*, *OLE DB for DM* y *JSR-073*. Existen otros estándares de minería de datos para el proceso global de minería de datos (como el *CRISP-DM*), que se tratarán en el Capítulo 22.

Aunque la “*XELOPES Data Mining Library*” se describirá en el Apéndice A es quizás otra muestra de la integración a la que están llegando muchos de estos estándares. Por ejemplo, esta librería incluye soporte para CWM, PMML, OLE DB for DM, JDM API, MLC++ y WEKA.

5.5.3 Interfaces visuales de minería de datos

Uno de los aspectos que ha hecho popularizar la minería de datos en la última década es la aparición de unas interfaces visuales que facilitan en gran medida la realización de todo el proceso de extracción de conocimiento. Recordemos que parte de los usuarios potenciales de la minería de datos no son informáticos ni profesionales de las tecnologías de la información, sino que pueden ser directivos o analistas. En vez de tener que aprender lenguajes al estilo del SQL que, si lo conocen, probablemente les costó aprender, o, todavía peor, conjuntos de primitivas, existen actualmente herramientas visuales que permiten realizar el proceso de extracción de conocimiento de una manera visual, indicando con el

ratón qué transformaciones, procesos y procedimientos aplicar, de una manera relativamente sencilla.

Los ejemplos más paradigmáticos de esta manera de trabajar quizá sean las interfaces del SPSS Clementine y del SAS Enterprise Miner, ambas muy similares. Por ejemplo, en la Figura 5.18 se muestra un ejemplo de un proceso de extracción de conocimiento con SPSS Clementine. Como se puede ver en la parte superior izquierda, la información parte de un nodo denominado “titanic.dat” y se le van aplicando nodos, transformándose, analizando, seleccionando, partiendo, visualizando en distintas ramas, que se pueden seguir por las flechas que conectan los nodos. Esto hace ver el proceso de minería de datos como un flujo de trabajo (“workflow”), donde cada nodo transforma información en otra información.

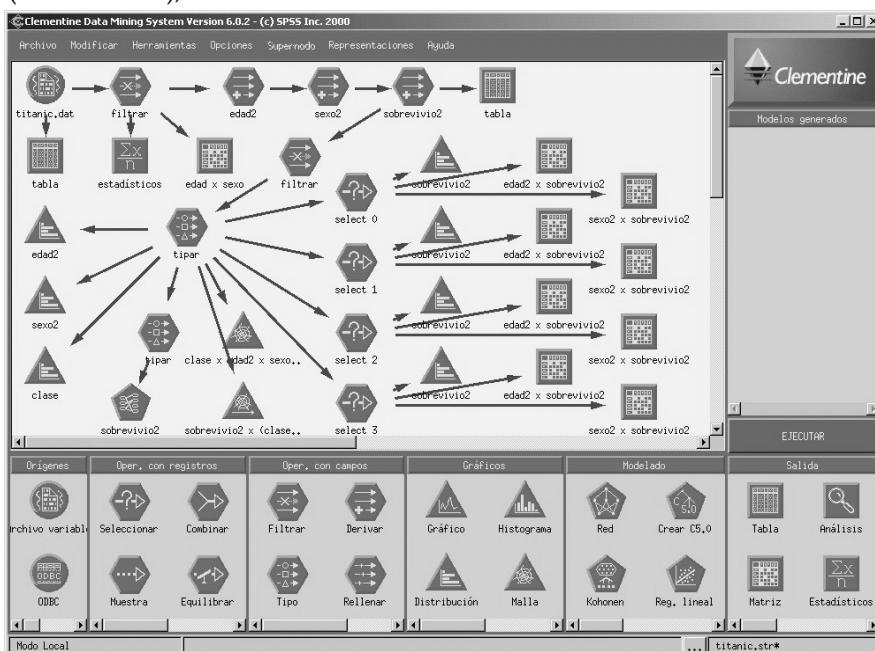


Figura 5.18. Ejemplo de interfaz visual del paquete de minería de datos SPSS Clementine.

Como se ve en la parte inferior de la Figura 5.18 existen varios tipos de nodos. En el caso del Clementine, tenemos nodos “orígenes” para poder leer o incorporar los datos, tenemos nodos “oper. con registros” y “oper. con campos” para realizar operaciones de selección, proyección, transformación, muestreos, concatenación, etc., sobre las tablas, nodos “gráficos” para representar los datos con distintas gráficas, nodos de “modelado” para aplicar distintos métodos de minería de datos y nodos de “salida”, para evaluar, visualizar o exportar los modelos extraídos. En el Apéndice A hablaremos con más detalle de los sistemas y comentaremos otros aspectos de los mismos, como el nivel de acoplamiento con la base de datos o almacén de datos.

Existen otro tipo de funciones que se pueden hacer de una manera visual, incluidas en otros sistemas, como la definición de jerarquías y el establecimiento de conocimiento previo, el diseño de experimentos y de validación, operadores OLAP, etc. A medida que avanzan las versiones de los sistemas, van incorporando cada vez más herramientas (traducidas en nodos u opciones de los mismos).

Las interfaces visuales tienen sus desventajas. La primera es que el usuario se acostumbra a utilizar una herramienta y acaba dependiendo de ella. Además, se dificulta en gran medida el poder portar a otras herramientas el flujo o trabajo de minería de datos realizado. En segundo lugar, estos entornos visuales están diseñados para que las operaciones y el flujo se vayan construyendo a mano. Afortunadamente, el segundo problema se resuelve en gran medida si la herramienta proporciona un lenguaje de macros o de *scripts*. Además, si este lenguaje sigue algún tipo de estándar, se resolvería también el primer problema. Esta parece ser la tendencia de cara al futuro, tener lenguajes y primitivas estándar y, por encima de ellas, interfaces visuales sobre las herramientas.

PARTE III

TÉCNICAS DE

MINERÍA DE DATOS

Esta parte es la más extensa y ligeramente más técnica del libro, que trata de describir el funcionamiento y la conveniencia de las técnicas de la fase central del proceso: la fase de minería de datos. Los once capítulos de esta parte cubren técnicas incorporadas de muy distintos ámbitos, desde la estadística hasta los algoritmos evolutivos. Debido a esta variedad se ha intentado que cada capítulo sea lo más autocontenido posible.

CAPÍTULOS

- 6. El problema de la extracción de patrones**
- 7. Modelización estadística paramétrica**
- 8. Modelización estadística no paramétrica**
- 9. Reglas de asociación y dependencia**
- 10. Métodos bayesianos**
- 11. Árboles de decisión y sistemas de reglas**
- 12. Métodos relacionales y estructurales**
- 13. Redes neuronales artificiales**
- 14. Máquinas de vectores soporte**
- 15. Extracción de conocimiento con algoritmos evolutivos y reglas difusas**
- 16. Métodos basados en casos y en vecindad**

Capítulo 6

EL PROBLEMA DE LA

EXTRACCIÓN DE PATRONES

La extracción de conocimiento a partir de datos tiene como objetivo descubrir patrones que, entre otras cosas, deben ser válidos, novedosos, interesantes y, en última instancia, comprensibles. Los seres humanos tenemos una capacidad innata de ver patrones a nuestro alrededor, incluso donde no los hay (borreguitos en las nubes, cuadrigas en las estrellas, pautas en la ruleta, recompensas divinas tras las plegarias, mariposas en manchas de tinta, etc.). A veces nos parece tan sencillo que no caemos en la cuenta de la complejidad intrínseca que existe en procesos tan rutinarios como estimar a qué hora hay que ponerse el despertador para llegar a tiempo al trabajo. Las técnicas de minería de datos (en especial las heredadas del aprendizaje automático y del reconocimiento de formas) han querido emular, en un primer momento, e ir más allá, más tarde y en ciertos aspectos, a estas capacidades de aprendizaje.

En los capítulos siguientes vamos a ver muchas técnicas de minería de datos de muy diferente tipo. Antes de pasar a ver las particularidades de cada uno de ellos, es conveniente ver, de una manera más genérica, qué tienen todas estas técnicas en común, a qué problema se enfrentan, de qué depende la dificultad de cada método y de qué maneras se puede expresar el resultado de estas técnicas. En particular, nos centraremos en estudiar dos aspectos fundamentales: la expresividad y la comprensibilidad de los modelos. Éstas, junto a otras características (robustez, facilidad de uso, precisión, eficiencia, etc.), nos van a permitir realizar una comparativa de las técnicas más comunes que se verán en los capítulos siguientes.

6.1 Introducción

Se dice comúnmente que el proceso de minería de datos convierte datos en conocimiento, tal cual un alquimista pudiera convertir espigas de trigo en lingotes de oro. Por si esto no

fuera poco, en algunos casos se llega a decir que el objetivo es extraer “verdad a partir de basura” [Thornton 2000]. Si nos referimos al contexto de trabajo de capítulos anteriores, en los que estuvimos preparando unos datos para, por fin, aplicar una técnica de minería de datos, las perspectivas anteriores se podrían resumir como se ilustra en la Figura 6.1.

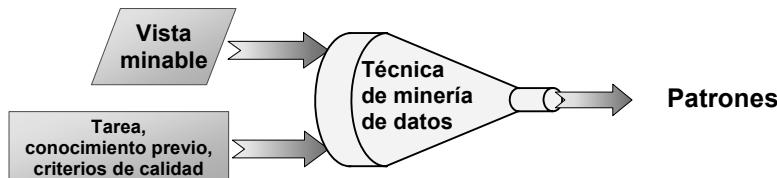


Figura 6.1. Proceso ideal de minería de datos.

En la figura las técnicas de minería de datos aparecen como una especie de colador pasapurés que, al introducirle los datos (en forma de vista minable junto a ciertos elementos asociados) produce, sin grumos ni atasco alguno, una serie de patrones lustrosos y relucientes. Aunque la idea es simplista, resulta útil, y, de hecho, gráficamente, es la utilizada en las herramientas visuales que comentamos en el capítulo anterior.

Ciertamente las cosas no son tan simples como en la figura. Hemos de pensar que el colador pasapurés debe de ser algo mágico o, al menos, muy sofisticado en su interior. Como la magia y la prestidigitación requieren muchas más horas de aprendizaje que la minería de datos, hemos de afrontar la realidad: en general, los procesos que extraen patrones a partir de datos son computacionalmente costosos y, lo que es más importante, son más costosos cuanto más expresivos, novedosos, comprensibles e interesantes queramos que sean los patrones extraídos.

De hecho, cuando algún algoritmo obtiene malos resultados no se debe a que la mayoría de investigadores de numerosas universidades y centros de investigación que han trabajado durante las últimas décadas en realizarlo y perfeccionarlo sean unos ineptos. Es mucho más probable que se trate de que, o bien no existe patrón en los datos, o bien no estemos utilizando la herramienta adecuada para encontrarlos, o bien el patrón sea realmente difícil de encontrar.

Existen tareas, presentaciones de tareas, instancias de tareas, que son más sencillas que otras. Por ejemplo, la extracción de reglas de asociación es un problema más sencillo, computacionalmente hablando, que la clasificación. Esto quiere decir que, para los mismos datos, generalmente deberemos esperar menos tiempo a que un algoritmo de extracción de reglas de asociación acabe que un algoritmo de clasificación. Del mismo modo, dos problemas de clasificación, con el mismo número de ejemplos, tipos de atributos y números de clases pueden diferir en dificultad. Dependerá de lo intrincados que estén los patrones en los datos o si realmente existen patrones plausibles en ellos.

Además de los datos y de la tarea, existen otros aspectos que influyen en el aprendizaje, que suelen denominarse conjuntamente *bias*. Quizás el *bias* que más influye en esta complejidad sea la manera de expresar o definir los patrones (*bias* del lenguaje). Por ejemplo, no es lo mismo una regresión lineal que una regresión realizada por una red neuronal multicapa. Ambos métodos permiten realizar la misma tarea, pero la expresividad y la mayor capacidad de la segunda se paga, de alguna manera, con un mayor tiempo de espera para obtener el modelo. El conocimiento previo es otro tipo de *bias*, que puede ayudar a refinar el espacio de búsqueda (*bias* de búsqueda).

6.2 Tareas y métodos

Una de las primeras cosas que debemos clarificar definitivamente antes de continuar es diferenciar una tarea de un método, así como destacar las tareas y métodos más relevantes. Una (un tipo de) tarea de minería de datos es un (tipo de) problema de minería de datos. Por ejemplo, “clasificar las piezas del proveedor Minatronix en óptimas, defectuosas reparables y defectuosas irreparables” es una tarea. Concretamente, el tipo de la tarea es *clasificación*. Esta tarea, por ejemplo, se podría resolver mediante árboles de decisión o redes neuronales, entre otros métodos. Éstos son métodos o técnicas que permiten resolver tareas. Es muy importante distinguir el problema de los métodos para solucionarlo. Pasemos a ver, en primer lugar, las tareas y, posteriormente, los métodos más importantes.

6.2.1 Tareas

En el Capítulo 2 vimos los tipos de tareas más importantes, clasificación, regresión, agrupamiento, reglas de asociación, etc., acompañadas de ejemplos. En esta sección vamos a ver una relación más completa y una descripción más precisa de cada una de las tareas.

En primer lugar, para definir las tareas debemos definir el conjunto de ejemplos con los que se van a tratar. Definamos E como el conjunto de todos los posibles elementos de entrada. Las instancias posibles dentro de E generalmente se representan como un conjunto de valores para una serie de atributos (sean nominales o numéricos). Es decir $E = A_1 \times A_2 \times \dots \times A_n$ y un ejemplo e es una tupla $\langle a_1, a_2, \dots, a_n \rangle$ tal que $a_i \in A_i$. Como veremos en próximos apartados, esta presentación *tabular* es la más habitual (pares atributos-valor), pero no la única.

Veamos a continuación las tareas más importantes en minería de datos.

- **Predictivas:** se trata de problemas y tareas en los que hay que predecir un o más valores para uno o más ejemplos. Los ejemplos en la evidencia van acompañados de una salida (clase, categoría o valor numérico) o un orden entre ellos. Dependiendo de cómo sea la correspondencia entre los ejemplos y los valores de salida y la presentación de los ejemplos podemos definir varias tareas predictivas:
 - **Clasificación (o discriminación¹⁴):** los ejemplos se presentan como un conjunto de pares de elementos de dos conjuntos, $\delta = \{ \langle e, s \rangle : e \in E, s \in S \}$, donde S es el conjunto de valores de salida. Los ejemplos e , al ir acompañados de un valor de S , se denominan comúnmente ejemplos etiquetados $\langle e, s \rangle$ y, en consecuencia, δ se denomina conjunto de datos etiquetado. El objetivo es aprender una función $\lambda: E \rightarrow S$, denominada clasificador, que represente la correspondencia existente en los ejemplos, es decir, para cada valor de E tenemos un único valor para S . Además, S es nominal, es decir, puede tomar un conjunto de valores c_1, c_2, \dots, c_m , denominados clases (cuando el número de clases es dos, tenemos lo que se llama clasificación binaria). La función aprendida será capaz de determinar la clase para cada nuevo ejemplo sin etiquetar, es decir dará un valor de S para cada valor de e . Ejemplos: clasificar un mensaje de correo electrónico como *spam* o no, clasificar entre varios medicamentos cuál es el mejor para una determinada pa-

¹⁴ El término discriminación se utiliza, fundamentalmente, en estadística.

tología... La tarea de clasificación es una de las más frecuentes en la minería de datos.

- **Clasificación suave:** la presentación del problema es la misma que la de la clasificación, pares de elementos de dos conjuntos, $\delta = \{ \langle e, s \rangle : e \in E, s \in S \}$. Además de la función $\lambda: E \rightarrow S$ se aprende otra función $\theta: E \rightarrow \mathcal{R}$ que significa el grado de certeza de la predicción hecha por la función λ . Lógicamente, siempre es preferible tener un clasificador suave que acompañe a las predicciones de una medida de certeza o de fiabilidad de dichas predicciones (aunque sea una mera estimación). Este tipo de extensión permite realizar otras aplicaciones, como son los *rankings* de predicciones o la selección de los n mejores ejemplos. Ejemplos: clasificar un mensaje de correo electrónico como *spam* o no, proporcionando, además, la certeza de la clasificación, clasificar entre varios medicamentos cuál es el mejor para una determinada patología, proporcionando, además, la certeza de la clasificación...
- **Estimación de probabilidad de clasificación:** se trata, en realidad, de una generalización de la clasificación suave. La presentación del problema es la misma que la de la clasificación normal y suave, pares de elementos de dos conjuntos, $\delta = \{ \langle e, s \rangle : e \in E, s \in S \}$. La función a aprender, sin embargo, es distinta de la clasificación y de la función suave. Se trata de aprender exclusivamente m funciones $\theta_i: E \rightarrow \mathcal{R}$, donde m es el número de clases. Es decir, cada función a aprender retorna para cada ejemplo e un valor real p_i . Cada uno de estos valores p_i se denomina probabilidad de la clase i y significa el grado de certeza de que un ejemplo sea de la clase i . Idealmente, si además se cumple que $\forall p_i: 0 \leq p_i \leq 1$, y además, $\sum p_i = 1$, estas p_i representan la *probabilidad* de que un ejemplo sea de la clase i . El conjunto de funciones aprendidas se denomina *estimador de probabilidad*. Nótese que si tenemos un estimador de probabilidades, para predecir la clase sólo tenemos que hacer *argmax* p_i . A veces, especialmente para dos clases, sólo se proporciona una de las probabilidades (ya que la otra se obtiene como $1 - p_1$). En este caso un estimador de probabilidades y un clasificador suave vienen a ser lo mismo, ya que con la probabilidad de una de las clases, podemos saber la clase predicha y su certeza. Ejemplo: dado el problema de clasificar entre varios medicamentos cuál es el mejor para una determinada patología, proporcionar la probabilidad de que sea cada uno de los medicamentos.
- **Categorización:** no se trata de aprender una función, sino una correspondencia. Es decir, cada ejemplo de $\delta = \{ \langle e, s \rangle : e \in E, s \in S \}$, así como la correspondencia a aprender $\lambda: E \rightarrow S$, pueden asignar *varias* categorías a un mismo e , a diferencia de la clasificación, que sólo asigna una y sólo una. Dicho de otra manera, un ejemplo podría tener varias categorías asociadas. Ejemplos: dado un conjunto de documentos, asignar categorías de los temas que trata cada documento, dados un conjunto de perfiles de clientes, determinar los productos que puedan comprar... La categorización se puede presentar también en forma de categorización suave (cada categoría asignada va acompañada de su certeza) o en forma de un estimador de probabilidades (se estima una probabilidad para todas las categorías), en este caso la suma de probabilidades puede ser mayor que 1. En este caso, nótese que, aunque en δ podemos tener varias etiquetas para el mis-

mo ejemplo, la función aprendida por un estimador de probabilidades de clasificación y un estimador de probabilidades de categorización viene a ser prácticamente lo mismo. En realidad usarlo como clasificador consiste en seleccionar la clase de mayor probabilidad y realizar un categorizador consiste simplemente en seleccionar las k mejores categorías o las categorías que superen un cierto valor.

- **Preferencias o priorización:** el aprendizaje de preferencias consiste en determinar a partir de dos o más ejemplos, un orden de preferencia. La definición formal es un poco más compleja. Cada ejemplo es en realidad una secuencia: $\langle e_1, e_2, \dots, e_k \rangle$, $e_i \in E$, $k \geq 2$, donde el orden de la secuencia representa la predicción. Un conjunto de datos para este problema es, por tanto, un conjunto de secuencias δ : $\{ \langle e_1, e_2, \dots, e_k \rangle : e_i \in E \}$. Otra manera alternativa de presentar los datos es mediante un orden parcial, que se puede considerar un caso particular de la anterior, donde las secuencias sólo tienen dos elementos ($k=2$). Ejemplos: dados una serie de candidatos para un trabajo, dar un orden priorizado para cubrir el puesto (el modelo de preferencia se habrá estimado a partir de selecciones anteriores (priorizaciones) o comparaciones de grupos de candidatos anteriores, etc). Una simplificación del problema sería obtener sólo preferencias entre dos elementos. El modelo aprendido sólo sería capaz de decir cuál prefiere entre dos objetos (decir si e_i es mejor que e_j), pero no ordenar más de dos objetos. En otros casos, lo que se aprende es un valor real de la magnitud del ejemplo, y el resultado se parece mucho a la regresión (aunque el valor de salida es ficticio). Quizá lo más característico de esta tarea es la presentación de los datos, ya que, por ejemplo, con un clasificador suave o un estimador de probabilidades también se pueden hacer priorizaciones, aunque aquí lo que se prioriza es la clase no el ejemplo completo.
- **Regresión:** es quizá la tarea más sencilla de definir. El conjunto de evidencias son correspondencias entre dos conjuntos $\delta: E \rightarrow S$, donde S es el conjunto de valores de salida. Al igual que con la clasificación, los ejemplos, al ir acompañados de un valor de S , se denominan comúnmente ejemplos etiquetados y δ es un conjunto de datos etiquetado. El objetivo es aprender una función $\lambda: E \rightarrow S$ que represente la correspondencia existente en los ejemplos, es decir, para cada valor de E tenemos un único valor para S . La diferencia respecto a la clasificación es que S es numérico, es decir puede ser un valor entero o real. Ejemplos: estimar las ventas del año 2007, predecir el número de unidades defectuosas de una partida de productos, predecir la presión de una válvula a partir de las entradas... La regresión se conoce con otros nombres: interpolación (generalmente cuando el valor predicho está en medio de otros) o estimación (cuando se trata de algo futuro). Es importante destacar de nuevo las conexiones entre distintas tareas; por ejemplo, un modelo de regresión se puede convertir en un clasificador binario suave si se establece un umbral u para la salida y de la función λ , es decir, si $y < u$ tendremos una clase y si $y \geq u$ tendremos la otra clase. Este es el caso del método denominado “regresión logística” que en realidad se utiliza para clasificación.

- **Descriptivas:** los ejemplos se presentan como un conjunto $\delta = \{ e : e \in E \}$, sin etiquetar ni ordenar de ninguna manera. El objetivo, por tanto, no es predecir nuevos datos sino describir los existentes. Lógicamente, esto se puede hacer de muchas maneras y la variedad de tareas se dispara. Algunas de las técnicas vistas en los capítulos 4 y 5 se pueden considerar tareas descriptivas (detección de valores anómalos, tablas de frecuencias, análisis de componentes principales). No obstante, vamos a ver las tareas descriptivas más delimitadas:
 - **Agrupamiento (clustering):** el objetivo de esta tarea es obtener grupos o conjuntos entre los elementos de δ , de tal manera que los elementos asignados al mismo grupo sean *similares*. Lo importante del agrupamiento respecto a la clasificación es que son precisamente los grupos y la pertenencia a los grupos lo que se quiere determinar y, a priori, no se sabe ni cómo son los grupos ni cuántos hay. En algunos casos se puede proporcionar el número de grupos que se desea obtener. Otras veces, este número se determina por el algoritmo de agrupamiento, según las características de los datos. La función a obtener es idéntica a la de la clasificación, $\lambda: E \rightarrow S$, con la diferencia de que los valores de S y sus miembros se *crean* o *inventan*, durante el proceso de aprendizaje. En el Capítulo 2 ya se ilustró la utilidad principal del agrupamiento: averiguar que las instancias se agrupan en varios segmentos es útil porque podemos determinar el comportamiento de una nueva instancia viendo a qué grupo de instancias pertenece. Es decir, si tenemos que una nueva instancia o ejemplo es etiquetado en el grupo 3 significa que, posiblemente, en muchos aspectos, se comporte como el resto de elementos del grupo 3. Generalmente, posteriormente al agrupamiento, se toman decisiones diferentes para cada grupo. Por ejemplo, podemos agrupar los clientes en segmentos diferenciados, estudiar qué grupos se comportan mejor ante determinados productos, y después orientar ciertos productos a ciertos grupos. El agrupamiento se conoce muy frecuentemente por su término en inglés: *clustering*, o por traducciones alternativas: segmentación, aglomeración, racimamiento... También se pueden definir variantes para realizar un agrupamiento suave u obtener estimadores de probabilidad de agrupamiento, que proporcionan más flexibilidad y posibilidades a la hora de interpretar y trabajar con los grupos formados, o permiten construir taxonomías o agrupamientos jerárquicos. La tarea general de agrupamiento puede utilizar con objetivos ligeramente distintos. Por ejemplo, si reducimos un conjunto de datos de miles de ejemplos a media docena de grupos, y analizamos los grupos formados, podemos entender mejor los datos originales y, en cierto modo, estos grupos sirven como *resumen* de los datos originales. De hecho, muchos autores consideran el agrupamiento con este objetivo como una tarea nueva, llamada **sumarización**. De modo similar si buscamos subgrupos que se separen del resto de la población, tenemos lo que a veces se conoce como "**descubrimiento de subgrupos**" (*subgroup discovery*), que también se suele considerar una tarea en sí misma.
 - **Correlaciones y factorizaciones:** tratados someramente en el Capítulo 5, con el objetivo de ver la relevancia de atributos, detectar atributos redundantes o dependencias entre atributos, o seleccionar un subconjunto, pueden considerarse, también, una tarea de minería de datos. Los estudios correlacionales y factoria-

les se centran exclusivamente en los atributos numéricos (ya sean inicialmente numéricos o después de una numerización, como se vio en el Capítulo 4). El objetivo es ver, dados los ejemplos del conjunto $E = A_1 \times A_2 \times \dots \times A_n$, si dos o más atributos numéricos A_i y A_j están correlacionados linealmente o relacionados de algún otro modo. Este tipo de relaciones son bidireccionales o no orientadas. Para ver si existe algún tipo de orientación (causa / efecto) se pueden utilizar modelos de regresión¹⁵ restringidos a sólo esos dos atributos.

- **Reglas de asociación:** ésta es una de las tareas reinas de la minería de datos y que ha evolucionado conjuntamente, desde mediados de los 90, con la propia minería de datos. El objetivo, en cierto modo, es similar a los estudios correlacionales y factoriales, pero para los atributos nominales, muy frecuentes en las bases de datos. Este tipo de estudios reciben, además del nombre de análisis de asociaciones, el nombre de análisis de vínculos (*link analysis*), aunque este término también se utiliza en el agrupamiento jerárquico. Dados los ejemplos del conjunto $E = A_1 \times A_2 \times \dots \times A_n$, una regla de asociación se define generalmente de la siguiente forma: "si $A_i = a \wedge A_j = b \wedge \dots \wedge A_k = h$ entonces $A_r = u \wedge A_s = v \wedge \dots \wedge A_z = w$ ", donde todos los atributos son nominales y las igualdades se definen utilizando algún valor de los posibles para cada atributo. La regla anterior está orientada, es decir es una regla de asociación direccional. Por este motivo, las reglas de asociación orientadas también se llaman dependencias de valor. También existen reglas de asociación bidireccionales, en las que en vez de una implicación tenemos una coimplicación. Por ejemplo, si tenemos una regla del estilo "si compra_aguacates = sí \wedge compra_cebollines = sí entonces compra_limones = sí", ésta sería una regla de asociación direccional u orientada. Esta regla tiene una orientación y no puede ser entendida en el orden inverso (si se compran limones no podemos concluir nada). En cambio, si tenemos una regla del estilo "compra_hamburguesa = sí sucede conjuntamente con compra_ketchup = sí". Esta regla quiere decir que si se compran hamburguesas se suele comprar *ketchup* y si se compra *ketchup* se suelen comprar hamburguesas. En realidad se buscan generalmente conjuntos de reglas de asociación, es decir más de una regla de asociación. A veces, conseguimos un conjunto tan bueno de reglas de asociación que si nos centramos en un solo atributo en la parte derecha de reglas orientadas podemos llegar a tener casi un clasificador (parcial, sólo algunos casos y sólo de una clase). Existen muchas variantes de las reglas de asociación: reglas de asociación negativas (se pueden incluir desigualdades, lo que tiene interés cuando los atributos pueden tener más de dos valores posibles), reglas de asociación secuenciales (cuando las asociaciones no ocurren en el mismo momento sino en sucesivos registros en un intervalo de tiempo), multínivel (se consideran las categorías de los productos, por ejemplo), etc. Todas ellas pueden ser orientadas o no.
- **Dependencias funcionales:** el descubrimiento de dependencias funcionales generalmente se suele incluir dentro de la variedad de tareas con el nombre de

¹⁵ Estaríamos utilizando una tarea predictiva no para predecir un valor a partir del otro, sino para ver la dependencia de los dos valores (analizando, por ejemplo el coeficiente de la regresión lineal obtenida).

“reglas de asociación”. En realidad las dependencias funcionales consideran todos los posibles valores (a diferencia de las asociaciones o dependencias de valor). Se definen de la siguiente manera: “dados los valores de A_i, A_j, \dots, A_k puedo determinar el valor de A_r ”. Es decir, el valor de A_r depende o es función de los valores de ciertos atributos A_i, A_j, \dots, A_k . Por ejemplo, una dependencia funcional podría ser “dada la edad (discretizado en seis intervalos), el nivel de ingresos (discretizado en cinco intervalos), el código postal, si está casado o no, puedo determinar con bastante fiabilidad si el cliente tiene vehículo”. Las dependencias funcionales, en particular cuando hay un atributo a cada lado, pueden ser orientadas y no orientadas, al igual que las reglas de asociación.

- **Detección de valores e instancias anómalas:** la detección de *valores anómalos* o *atípicos* (*outlier detection*) se vio en el Capítulo 4, con el objetivo de realizar limpieza de datos. No obstante, la detección de valores anómalos puede ser muy útil para detectar precisamente comportamientos anómalos, que pueden sugerir fraudes, fallos, intrusos o comportamientos diferenciados. La definición de instancia anómala es más general en el sentido que no sólo considera un único atributo, sino que los considera todos. La tarea se define con el objetivo de encontrar aquellas instancias que no son similares a ninguna (o muy pocas) de las otras instancias. La manera de abordar el problema es generalmente la de agrupar los ejemplos y ver aquellas instancias que se quedan “desplazadas” de los grupos mayoritarios. Para ello son especialmente útiles los agrupadores suaves o los estimadores de probabilidad de agrupamiento, ya que si un ejemplo tiene baja probabilidad de agrupamiento con todos los grupos se puede considerar un caso “aislado” y, por tanto, anómalo. También se utilizan otros métodos no necesariamente basados en la tarea de agrupamiento, como la medición de distancias (aquellas instancias cuyo vecino más próximo esté muy lejos puede considerarse, en cierto modo, una instancia anómala). Un ejemplo de tarea de detección de instancias anómalas sería: “encontrar, de las compras realizadas con tarjeta, aquellas que sean anómalas”.

Como hemos visto, algunas tareas están relacionadas. Esto, en cierto modo, ha hecho que la terminología de algunas de ellas sea bastante diversa y, a veces, sea difícil aclararse con los nombres que utilizan algunas herramientas o textos de referencia. Por ejemplo, se suele utilizar el término “aprendizaje supervisado” para los métodos predictivos y el término “aprendizaje no supervisado” para los métodos no predictivos. En realidad, los términos originales eran más precisos; el aprendizaje supervisado generalmente se utiliza para nombrar, de entre todos los métodos predictivos, la clasificación y, rara vez, la regresión. En cambio, el agrupamiento se considera el problema no supervisado por excelencia, mientras el resto de métodos descriptivos no lo son en realidad. La importancia de estos nombres actualmente es puramente terminológica, porque muchos libros de texto, programas, artículos, etc., todavía pueden utilizar esta notación.

Un caso especial de relación entre tareas ocurre cuando tenemos un problema de clasificación de más de dos clases (multiclasificación) y queremos resolverlo mediante clasificadores que sólo consigan discriminar entre dos clases (clasificación binaria), ya sea porque el algoritmo que vamos a utilizar sólo permite clasificación binaria, porque queremos utilizar el análisis ROC o por otras razones. El proceso de adaptar un problema

de más de dos clases a un problema de dos clases se denomina **binarización**. Existen varios métodos para realizarla:

- Uno frente al resto (*One-vs-all*): se construye un clasificador binario utilizando todos los ejemplos de una clase y agrupando en una misma clase el resto de ejemplos. Esto se realiza para todas las clases. Si tenemos n clases, el resultado serán n clasificadores binarios que posteriormente habremos de combinar. Ésta es la misma aproximación que se utiliza para convertir problemas de categorización en problemas de clasificación, con la diferencia de que, en categorización, luego no hay que “fusionar” los resultados, puesto que podemos quedarnos con varias categorías y no con una sola, como en el caso de la clasificación.
- Todos los pares (*All-pairs*): se construye un clasificador binario utilizando todos los ejemplos de dos clases e ignorando el resto. Esto se realiza para todos los pares de clases. Si tenemos n clases, el resultado serán $n(n-1)/2$ clasificadores binarios que posteriormente habremos de combinar.
- Todas las mitades (*All-halves*): se construye un clasificador binario utilizando los ejemplos de la mitad de las clases por un lado y el resto por el otro. Si tenemos n clases y este número es par, tendremos $n/2$ clases por un lado y $n/2$ clases por el otro. Esto se realiza para todas las particiones posibles del mismo número de clases. También se puede realizar para todas las posibles particiones, tengan o no el mismo número de clases, lo que dispara el número de variantes. La combinación en este caso es más costosa y más complicada porque la clasificación final se hace como una ponderación, ya que ninguno de los clasificadores binarios representa a clases simples.

Para la combinación de los clasificadores binarios en el clasificador final se pueden utilizar muchas técnicas, que se basan mayoritariamente en utilizar la confianza de la predicción de cada clasificador parcial, como por ejemplo la técnica ECOC (*Error Correcting Output Code*) [Allwein et al. 2000]. Por tanto, vemos que para convertir un problema de multiclasicación en clasificación binaria, los clasificadores binarios han de ser suaves.

Otra distinción importante a realizar es entre modelos (o patrones) *totales* y *parciales*. El concepto es el mismo que para las funciones; una función total define imagen para todos los valores del dominio, mientras que una función parcial sólo define imagen para algunos de los valores del dominio, siendo indefinido para el resto.

Por ejemplo, cuando hemos tratado de clasificación, hemos considerado que los clasificadores aprendidos eran funciones totales. ¿Qué ocurre, sin embargo, si el clasificador es parcial, es decir, se abstiene para algunos ejemplos? En realidad, siempre es preferible tener un clasificador total, pero en otros casos nos puede ser útil también tener un conjunto menor de reglas que sean capaces de predecir la clase en el 80 por ciento (alcance o *recall* del modelo) de los casos, que tener muchísimas más reglas para poder predecir la clase en el 100 por 100 de los casos. Además, la precisión en el caso del 80 por ciento puede ser mayor que con el 100 por 100. Por ejemplo, una aproximación que puede ser muy útil es obtener un conjunto de reglas de asociación (con la restricción de que a la parte derecha tengan el atributo de la clase). Juntando unas cuantas reglas de asociación se puede construir un clasificador parcial que, además, puede aportar bastante conocimiento.

6.2.2 Métodos. Correspondencia entre tareas y métodos

Cada una de las tareas anteriores, como cualquier problema, requiere métodos, técnicas o algoritmos para resolverlas. Una de las cosas que sorprenden a los recién llegados a la minería de datos es que, además de que, lógicamente, una tarea puede tener muchos métodos diferentes para resolverla, tenemos que el mismo método (o al menos el mismo tipo de técnica) puede resolver un gran abanico de tareas. Como veremos en la sección siguiente, esto no es casualidad, sino que se debe a que, en el fondo, la mayoría de las tareas que acabamos de ver son caras de la misma moneda, el aprendizaje inductivo.

Veamos brevemente los tipos de técnicas existentes para llevar a cabo las tareas anteriores. La relación que se muestra a continuación sólo pretende dar una reseña de la variedad de técnicas existentes. Estas y otras técnicas se verán extensamente en los capítulos siguientes.

- Técnicas algebraicas y estadísticas: se basan, generalmente, en expresar modelos y patrones mediante fórmulas algebraicas, funciones lineales, funciones no lineales, distribuciones o valores agregados estadísticos tales como medias, varianzas, correlaciones, etc. Frecuentemente, estas técnicas, cuando obtienen un patrón, lo hacen a partir de un modelo ya predeterminado del cual, se estiman unos coeficientes o parámetros, de ahí el nombre de técnicas *paramétricas*. Algunos de los algoritmos más conocidos dentro de este grupo de técnicas son la regresión lineal (global o local), la regresión logarítmica y la regresión logística. Los discriminantes lineales y no lineales, basados en funciones predefinidas, es decir discriminantes paramétricos, entran dentro de esta categoría. No obstante, aunque el término “no paramétrico” se utiliza para englobar gran parte de técnicas provenientes del aprendizaje automático, como son las redes neuronales, también existen muchas técnicas de modelización estadística no paramétrica.
- Técnicas bayesianas. se basan en estimar la probabilidad de pertenencia (a una clase o grupo), mediante la estimación de las probabilidades condicionales inversas o a priori, utilizando para ello el teorema de Bayes. Algunos algoritmos muy populares son el clasificador bayesiano naive, los métodos basados en máxima verosimilitud y el algoritmo EM. Las redes bayesianas generalizan las topologías de las interacciones probabilísticas entre variables y permiten representar gráficamente dichas interacciones.
- Técnicas basadas en conteos de frecuencias y tablas de contingencia: estas técnicas se basan en contar la frecuencia en la que dos o más sucesos se presenten conjuntamente. Cuando el conjunto de sucesos posibles es muy grande, existen algoritmos que van comenzando por pares de sucesos e incrementando los conjuntos sólo en aquellos casos que las frecuencias conjuntas superen un cierto umbral. Ejemplos de estos algoritmos son el algoritmo “Apriori” y similares.
- Técnicas basadas en árboles de decisión y sistemas de aprendizaje de reglas: son técnicas que, además de su representación en forma de reglas, se basan en dos tipos de algoritmos: los algoritmos denominados “divide y vencerás”, como el ID3/C4.5 o el CART, y los algoritmos denominados “separa y vencerás”, como el CN2.
- Técnicas relacionales, declarativas y estructurales: la característica principal de este conjunto de técnicas es que representan los modelos mediante lenguajes declarati-

vos, como los lenguajes lógicos, funcionales o lógico-funcionales. Las técnicas de ILP (programación lógica inductiva) son las más representativas y las que han dado nombre a un conjunto de técnicas denominadas *minería de datos relacional*.

- Técnicas basadas en redes neuronales artificiales: se trata de técnicas que aprenden un modelo mediante el entrenamiento de los pesos que conectan un conjunto de nodos o neuronas. La topología de la red y los pesos de las conexiones determinan el patrón aprendido. Existen innumerables variantes de organización: perceptrón simple, redes multicapa, redes de base radial, redes de Kohonen, etc., con no menos algoritmos diferentes para cada organización; el más conocido es el de retropropagación (*backpropagation*).
- Técnicas basadas en núcleo y máquinas de soporte vectorial: se trata de técnicas que intentan maximizar el margen entre los grupos o las clases formadas. Para ello se basan en unas transformaciones que pueden aumentar la dimensionalidad. Estas transformaciones se llaman núcleos (*kernels*). Existen muchísimas variantes, dependiendo del núcleo utilizado y de la manera de trabajar con el margen.
- Técnicas estocásticas y difusas: bajo este paraguas se incluyen la mayoría de las técnicas que, junto a las redes neuronales, forman lo que se denomina computación flexible (*soft computing*). Son técnicas en las que o bien los componentes aleatorios son fundamentales, como el *simulated annealing*, los métodos evolutivos y genéticos, o bien al utilizar funciones de pertenencia difusas (*fuzzy*).
- Técnicas basadas en casos, en densidad o distancia: son métodos que se basan en distancias al resto de elementos, ya sea directamente, como los vecinos más próximos (los casos más similares), de una manera más sofisticada, mediante la estimación de funciones de densidad. Además de los vecinos más próximos, algunos algoritmos muy conocidos son los jerárquicos, como Two-step o COBWEB, y los no jerárquicos, como K medias.

Además de todo lo anterior existen multitud de híbridos que dificultan más aún realizar una taxonomía razonable. Y todo esto sin tener en cuenta los métodos por combinación, que se tratarán en el Capítulo 18. En definitiva, es muy difícil realizar una taxonomía óptima que aglutine grupos de técnicas de una manera indiscutible y que, de alguna manera, no confunda tareas con métodos. De hecho, la exposición anterior no corresponde exactamente con la distribución de los siguientes capítulos. Por ejemplo, aunque en la organización de los capítulos siguientes se orienta más a separar capítulos según los tipos de técnicas (pudiéndose, en algunos capítulos, ver aplicaciones de las técnicas para diferentes métodos), en algunos casos (como en el Capítulo 9 de reglas de asociación), se ha optado por aglutinar según las tareas.

A título meramente ilustrativo, la siguiente tabla muestra algunas tareas (clasificación, regresión, agrupamiento, reglas de asociación, correlaciones/factorizaciones) y algunas técnicas o algoritmos (que se irán viendo en la Parte III de este libro) que pueden abordarlas:

Nombre	PREDICTIVO		DESCRIPTIVO		
	Clasificación	Regresión	Agrupamiento	Reglas de asociación	Correlaciones / Factorizaciones
Redes neuronales	✓	✓	✓		
Árboles de decisión ID3, C4.5, C5.0	✓				
Árboles de decisión CART	✓	✓			
Otros árboles de decisión	✓	✓	✓	✓	
Redes de Kohonen			✓		
Regresión lineal y logarítmica		✓			✓
Regresión logística	✓			✓	
Kmeans			✓		
Apriori				✓	
Naive Bayes	✓				
Vecinos más próximos	✓	✓	✓		
Análisis factorial y de comp. ppales.					✓
Twostep, Cobweb			✓		
Algoritmos genéticos y evolutivos	✓	✓	✓	✓	✓
Máquinas de vectores soporte	✓	✓	✓		
CN2 rules (cobertura)	✓			✓	
Análisis discriminante multivariante	✓				

Como se puede observar, la correspondencia entre tareas y técnicas es muy variada. Algunas tareas pueden ser resueltas por muy diversas técnicas y algunas técnicas pueden aplicarse para tres o incluso cuatro tareas. Esta variedad es una de las razones por la que es necesario conocer las capacidades de cada técnica, los ámbitos donde suele funcionar mejor, la eficiencia, la robustez, etc., en definitiva, las características *funcionales* de cada técnica respecto a las demás. Más adelante en este capítulo haremos una comparativa general de las técnicas más importantes. Los capítulos 7 a 16 ya analizan en más profundidad cada técnica, y resaltan sus ventajas e inconvenientes.

6.3 Minería de datos y aprendizaje inductivo

Al considerar tantas tareas y métodos parece que estamos tratando de problemas inconexos. Un aspecto chocante es que la misma técnica pueda utilizarse para varias tareas. Esto debe querer decir que hay algunos procesos útiles para una tarea que también lo es, generalmente, para otras, con una ligera adaptación. La pregunta, no obstante, nos asalta: ¿qué tiene que ver un agrupamiento con una regresión? O si hablamos de métodos, ¿qué tiene que ver una red neuronal con un árbol de decisión? ¿Cómo puede ser que todo esto entre dentro de la “extracción de conocimiento”?

La respuesta a estas preguntas consiste en reconocer que todas las tareas (exceptuando quizás las reglas de asociación y las correlaciones) y los métodos se centran alrededor de la idea del *aprendizaje inductivo*. Son, por decirlo de un modo más coloquial, presentaciones diferentes del mismo proceso. Para entender esta respuesta debemos, primero, definir qué es el aprendizaje inductivo.

Definir el aprendizaje no es sencillo, ya que es un término que se utiliza en Psicología, Pedagogía, Zoología, Antropología y, más recientemente, en Informática. Veamos cuatro definiciones diferentes (pero en cierto modo equivalentes) del aprendizaje.

- La visión más genérica define el aprendizaje como la mejora del comportamiento a partir de la experiencia [Mitchell 1997].
- Una visión más externa define el aprendizaje como la capacidad de predecir observaciones futuras con plausibilidad o explicar observaciones pasadas.
- Una visión más estática define el aprendizaje inductivo como la identificación de patrones, de regularidades, existentes en la evidencia.
- Una visión más teórica y matemática define el aprendizaje inductivo como eliminación de redundancia, vista como compresión de información [Solomonoff 1966].

Estas cuatro visiones se conjugan perfectamente de la siguiente manera: el aprendizaje nos permite identificar regularidades en un conjunto de observaciones. Estas regularidades son en realidad redundancias que pueden ser representadas por patrones o modelos que los compriman o que los definan. Estos patrones pueden ser utilizados para predecir observaciones futuras o explicar observaciones pasadas. Esta capacidad de predecir y explicar el entorno es fundamental para mejorar el comportamiento.

Las dos primeras definiciones son de aprendizaje, mientras que las dos últimas son de aprendizaje inductivo. El aprendizaje inductivo es un tipo especial de aprendizaje (el más importante, no obstante) que parte de casos particulares (ejemplos) y obtiene casos generales (reglas o modelos) que generalizan o abstraen la evidencia. Existen, además, otros tipos de aprendizaje, como por ejemplo el aprendizaje abductivo (o explicativo, conocido en inglés por EBL, *explanation-based learning*) [DeJong & Mooney 1986; Minton et al. 1989] y el aprendizaje por analogía, que van de casos particulares a casos particulares. Este tipo de aprendizaje es menos útil para la minería de datos y sólo veremos algunos tipos de técnicas que se podrían considerar dentro de esta categoría: los métodos por vecindad, basados en instancias y los métodos CBR (*case-based reasoning*) en el Capítulo 16.

El aprendizaje puede ser además de varios tipos según ciertas características de la presentación de los datos y de la forma de aprender. Por ejemplo, el aprendizaje puede ser *incremental* o no, dependiendo de si los datos se van presentando poco a poco, o si se tienen todos desde el principio. En el caso de la minería de datos se suele considerar un aprendizaje no incremental, ya que los datos se obtienen de una base de datos o un almacén de datos. No obstante, si se considera que los datos pueden ir cambiando o incrementándose (cada mes, por ejemplo), podría ser interesante utilizar métodos específicos para el aprendizaje incremental, que permite revisar los modelos aprendidos y no tener que realizarlos de nuevo con todos los datos.

Dependiendo de si se puede actuar sobre las observaciones, el aprendizaje puede ser interactivo o no. En el caso de la minería de datos tenemos generalmente datos históricos que no podemos afectar, con lo que no podemos actuar sobre las observaciones. El aprendizaje interactivo es útil cuando se puede interaccionar con un entorno para generar nuevas observaciones y así facilitar la tarea de aprendizaje. Es como una especie de diseño de experimentos. Un tipo especial de aprendizaje interactivo es aquel en el que podemos preguntar a un oráculo sobre cualquier ejemplo. Este tipo de aprendizaje puede empezar a ser relevante para la minería de datos web o minería de datos en entornos *software*, donde podamos considerar a los usuarios como oráculos, a los que podamos preguntar por sus preferencias. También podemos considerar a la base de datos como un oráculo, entonces

tenemos en realidad lo que se conoce como *query learning* (aprendizaje por consultas) [Angluin 1988].

Un tipo especial de aprendizaje interactivo es el aprendizaje por refuerzo [Sutton & Barto 1998], en el que un agente va realizando una serie de tareas, que si son acertadas son recompensadas positivamente (premio) y si son desacertadas se penalizan (castigo). Aunque no vamos a tratar este tipo de aprendizaje en este libro, puede ser de gran utilidad en ámbitos donde se desea personalizar portales web, aplicaciones o asistentes *software* a las preferencias del usuario, actuando éste de manera que recompense o penalice según la aplicación se comporte o no apropiadamente.

Vamos viendo que muchísimas técnicas de aprendizaje automático se utilizan en minería de datos. Otras, en cambio, no son de uso habitual en minería de datos. También hemos visto que algunas técnicas de minería de datos, como por ejemplo, las reglas de asociación, las correlaciones, etc., no son técnicas de aprendizaje automático. No obstante, la motivación y objetivos finales solapan en gran medida.

6.3.1 Los patrones son hipótesis. Evaluación

Sea como sea la presentación del problema, una de las características presente en cualquier tipo de aprendizaje y en cualquier tipo de técnica de minería de datos es su carácter *hipotético*, es decir, lo aprendido puede, en cualquier momento, ser refutado por evidencia futura. En muchos casos, los modelos no aspiran a ser modelos perfectos, sino modelos *aproximados*. En cualquier caso, al estar trabajando con hipótesis, es necesario realizar una evaluación de los patrones obtenidos, con el objetivo de estimar su validez y poder compararlos con otros.

La evaluación de modelos se ha visto brevemente en el Capítulo 2 y se tratará en detalle en el Capítulo 16. No obstante, vamos a conectar la manera de evaluar con las definiciones de aprendizaje anteriores. Según las dos primeras definiciones del aprendizaje, la manera de evaluar un patrón o modelo es ver cómo se comporta con observaciones futuras. Ésta es la base, por ejemplo, de las medidas de evaluación basadas en conjunto de entrenamiento y conjunto de test, o el método de validación cruzada, que trataremos en el Capítulo 17. Separando, por tanto, un subconjunto de datos frescos (independientes) que haga el papel de evidencia futura, podemos estimar diferentes funciones de puntuación (*score functions*) para los modelos. Por ejemplo, para clasificación podemos utilizar el porcentaje de errores (o de aciertos), para la regresión el error cuadrático medio, para las reglas de asociación podemos usar el soporte y la confianza, etc. Según las otras dos definiciones de aprendizaje, un modelo es mejor cuanta más regularidad encuentra, es decir, cuanto más comprima la evidencia. Ésta es la base, de nuevo, de otras medidas de evaluación basadas en la navaja de Occam o en el principio de la longitud de la descripción mínima (MDL, del inglés *Minimum Description Length*) [Rissanen 1978; Li & Vitányi 1997] que también trataremos en el Capítulo 17. Estos criterios prefieren modelos simples sobre modelos complejos, estimándose la complejidad en número de reglas o, de una manera más sofisticada, el número de bits de una codificación del modelo.

Además de la evaluación de los modelos o patrones extraídos por cada método, es interesante evaluar y comparar métodos. Puede parecer evidente que un método será mejor que otro si genera mejores modelos. La curiosidad (aunque no podría ser de otra

manera) es que dado cualquier método de aprendizaje (por muy bueno que sea), siempre existen problemas que son resueltos mejor por otros métodos. Esta característica se denomina lacónicamente “no hay almuerzo gratis” (*no-free-lunch theorem*) [Wolpert & Macready 1997].

No obstante, como veremos en el Capítulo 17, sí que hay métodos que funcionan mejor que otros para ciertas familias de problemas o para los problemas típicos que se tratan en minería de datos. Este tipo de comparaciones experimentales son muy importantes de cara a contrastar la validez de las técnicas.

Por último, un aspecto importante en la evaluación de métodos es su variabilidad. Puede haber métodos de aprendizaje que tengan una precisión media muy alta, pero que tengan muchos altibajos. Esto se puede medir para distintos problemas, entonces se dice que el método tiene un amplio rango o abanico de aplicación. Además, podemos hablar de si el método funciona de manera regular para distintas muestras de datos del mismo problema, es decir, si el método es muy susceptible al orden, cantidad o subconjunto de datos presentado (del mismo problema). En este caso hablamos de estabilidad (o en términos contrarios, de debilidad, inestabilidad o variabilidad) de un algoritmo de aprendizaje. Otro término relacionado es la robustez al ruido, en el que se intenta medir si el método trata bien conjuntos de datos con valores erróneos o faltantes.

6.3.2 Métodos retardados y anticipativos. Comprensibilidad

Un aspecto muy relevante en la extracción de conocimiento es que los modelos extraídos sean comprensibles. La primera condición para que un modelo sea comprensible es que tengamos un modelo. Esto puede parecer una sandez, pero es que ocurre que existen técnicas de minería de datos que resuelven tareas sin construir modelos, al menos explícitamente. Por ejemplo, un clasificador de vecino más próximo es capaz de clasificar cualquier instancia futura mirando, de las instancias anteriores, cuál es la instancia más similar (más próxima) y retornando su clase. Este método no construye ningún modelo que aplique a una nueva instancia, sino que cada vez que se le pregunta actúa y obtiene una respuesta. En realidad, no obtiene una función completamente definida en algún tipo de lenguaje matemático o de reglas que podamos consultar y manejar. En cambio, una red neuronal, después del aprendizaje, se queda entrenada y constituye una función, un modelo, que puede aplicarse a nuevos ejemplos sin necesidad de mirar los anteriores.

Los métodos sin modelo y con modelo reciben generalmente el nombre de métodos retardados o perezosos (*lazy*) y métodos anticipativos o impacientes (*eager*):

- **Métodos retardados o perezosos:** el método actúa para cada pregunta o predicción requerida. No se construye modelo. Optimización local. Los ejemplos deben preservarse porque son necesarios para realizar cada predicción. El tiempo de respuesta empieza a degradarse cuando el número de ejemplos es muy grande porque hay que consultar muchos de ellos. En cambio, la ventaja es que no hay que entrenar el modelo.
- **Métodos anticipativos o impacientes:** el método obtiene un modelo a partir de todos los ejemplos. Los ejemplos, por tanto, pueden ignorarse. Optimización global. Se requiere un tiempo de entrenamiento, que suele ser grande, pero una vez entrenado el modelo, su aplicación suele ser instantánea.

En general, los métodos retardados, para ser eficientes, actúan generalmente teniendo en cuenta sólo los ejemplos cercanos. En este sentido, suelen ser locales. En cambio, los métodos impacientes construyen un modelo global. La gran mayoría de los métodos que veremos en los siguientes capítulos son anticipativos, excepto parte del Capítulo 8 y la mayoría de los del Capítulo 16.

Aclarado este punto, ahora nos podemos preguntar: de los métodos con modelo, ¿son todos comprensibles? La respuesta a esta pregunta es negativa. Basta examinar por un tiempo una red neuronal entrenada e intentar extraer algún conocimiento sobre los patrones encontrados. En cambio, un árbol de decisión con pocas reglas puede ser mucho más sencillo de examinar, de comprender y, por tanto, de validar, modificar y adaptar a las necesidades de aplicación.

En la Figura 6.2 se muestra una clasificación de algunos métodos donde se indica si generan modelo y, de éstos, cuáles generan un modelo comprensible.

	Con modelo	Sin modelo o no inteligible
Útiles para extracción de conocimiento Anticipativo	<ul style="list-style-type: none"> • Regresión lineal • K-medias. • ID3, C4.5, CART. • CN2. • Apriori • ILP, IFLP. • Redes bayesianas • Reglas difusas 	<ul style="list-style-type: none"> • Redes neuronales. • <i>Radial basis functions.</i> • Clasificador bayesiano naive. • Máquinas de vect. soporte • <i>Boosting.</i>
Retardado		<ul style="list-style-type: none"> • k-NN (Vecinos más próximos). • Regresión lineal pond. local • Otros métodos locales • CBR (<i>Case-based reasoning</i>)

Figura 6.2. Técnicas teóricamente útiles para extraer “conocimiento”.

Algunos de las clasificaciones de la figura anterior pueden ser discutibles. Por ejemplo, los métodos basados en núcleos (*ernels*), como las máquinas de vectores soporte, se vuelven incomprensibles si se utilizan núcleos complejos, ya que los patrones extraídos son en función de un espacio transformado por el núcleo y no del espacio original de atributos.

En realidad, si recordamos la definición de la extracción automática de conocimiento a partir de datos, teníamos que los modelos debían ser “en última instancia comprensibles”. A priori, esto descartaría muchos métodos que se usan generalmente en minería de datos. Como comentamos en los capítulos de introducción, no siempre es crítica esta comprensibilidad. En algunas aplicaciones, puede ser suficiente con que el método funcione bien.

No obstante, veremos en el Capítulo 19 cómo extraer reglas comprensibles a partir de modelos no comprensibles, por lo que, en realidad, se abre la puerta a utilizar todas estas técnicas no comprensibles.

6.3.3 La eficiencia del aprendizaje

Por último, un aspecto también fundamental a considerar en el aprendizaje es el esfuerzo computacional que se requiere. Lógicamente, entre dos métodos, preferiremos aquel que

obtenga patrones de una manera más rápida. La eficiencia del aprendizaje, como hemos comentado anteriormente, depende de muchos aspectos. Depende, generalmente, del número de ejemplos, del número de atributos o complejidad de los ejemplos, del espacio de hipótesis que se está considerando, del conocimiento previo existente, del nivel de error permitido, de lo sorprendente o innovador que se busquen los patrones. Existe una rama del aprendizaje automático, denominada Teoría del Aprendizaje Computacional (COLT, *Computational Learning Theory*), que se encarga de analizar esta complejidad para distintas tareas, dependiendo de todos estos factores. Veamos, simplemente, y de una manera informal, algunos de estos factores.

Uno de los factores que más influyen en la eficiencia del aprendizaje es el tamaño de los datos, que suele ser función del número de ejemplos, el número de atributos y la complejidad de los valores existentes (nominales, numéricos, con estructura, etc.). Uno de los aspectos que más influyen es precisamente el número de atributos, en particular si éstos son no significativos. En este caso tenemos un espacio de alta dimensionalidad y los métodos que se basan en distancias se pierden en un espacio tan poco denso. Este problema se denomina, como ya comentamos en la Sección 4.3.1, **la maldición de la dimensionalidad** (*the curse of dimensionality*).

Sea en relación al número de ejemplos o al número de atributos, en muchos casos, el tiempo de aprendizaje depende de una manera no lineal respecto al tamaño de los datos. Esto significa que cuando el volumen de datos se dispara debemos reducir el tamaño (mediante muestreo o mediante selección de atributos) o cambiar de algoritmo para conseguir tener unos tiempos de respuesta aceptables.

Otro aspecto que influye en el aprendizaje es la calidad de datos y la existencia de conocimiento previo, es decir, conceptos de apoyo que faciliten el aprendizaje. En los dos capítulos anteriores hemos visto la importancia de la preparación de datos.

El espacio de hipótesis, es decir la cantidad y expresividad de hipótesis que se están considerando es quizás el factor más determinante. Por ejemplo, no es lo mismo determinar unos coeficientes de una función lineal que determinar cientos de reglas de un árbol de decisión o de pesos en una red neuronal. Cuanto más expresivo sea el lenguaje para representar los patrones, más abierto es el espacio de búsqueda y, por tanto, más tiempo se requiere para explorarlo.

Dentro de la teoría del aprendizaje computacional, se sabe muy bien que el problema del aprendizaje para lenguajes de expresividad universales es intratable. Incluso se sabe que es intratable para lenguajes bastante restringidos. Existe un concepto, el del aprendizaje PAC (*Probabilistic Approximate Correct*) [Valiant 1984], que flexibiliza y parametriza el error permitido, mostrando que para algunos lenguajes se puede aprender de una manera eficiente. Esta teoría muestra que si relajamos el nivel de error de los modelos (permitiendo hasta un umbral de error máximo) y también relajamos la robustez del algoritmo (porcentaje de las veces en las que el algoritmo es capaz de estar por debajo del error establecido) podemos realizar sistemas de aprendizaje eficientes para algunas clases de problemas. No vamos a tratar en detalle la teoría del aprendizaje PAC; aunque es un concepto importante dentro del aprendizaje automático, no es determinante para muchas aplicaciones prácticas.

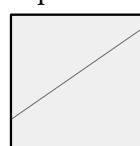
Existen otras medidas de la dificultad de un problema de aprendizaje. Por ejemplo, la dimensión VC (Vapnik – Chervonenkis) [Vapnik 1995] es una medida teórica que se puede estimar para problemas y también para sistemas de aprendizaje que permite, en cierto modo, saber si cierto método de aprendizaje puede ser capaz de capturar un determinado concepto. Cuanta mayor sea la dimensión VC de un problema, presumiblemente será mayor la dificultad de aprenderlo. De hecho, existen resultados que conectan esta dimensión con el aprendizaje PAC.

Finalmente, otro aspecto que afecta a la complejidad del aprendizaje son las aspiraciones de novedad, interés o sorpresa que uno se marque a los patrones extraídos. Aquello que está explícito en los datos (que se ve a simple vista) no suele ser interesante, pero tampoco cuesta. Es precisamente aquello que está implícito en los datos de una manera poco evidente, lo que generalmente sorprende y aporta conocimiento novedoso. Existen formalizaciones de este concepto [Li & Vitányi 1997; Hernández 1999].

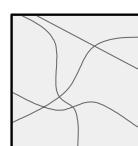
Para concluir, una idea interesante del diseño de algoritmos y que se aplica en algunos algoritmos de aprendizaje es el concepto de algoritmo *anytime*. Un algoritmo *anytime* produce soluciones mejores cuanto más tiempo se proporcione al algoritmo. Un algoritmo *anytime*, además, puede ser interrumpido en cualquier ocasión y mostrar la mejor solución obtenida hasta el momento. Esta idea es especialmente importante en el caso del aprendizaje, ya que muchas veces no sabemos cuánto tiempo va a costar un cierto aprendizaje (entre pocos segundos a muchas horas). Poder parar en cualquier momento y poder ver si la solución hasta ese momento es satisfactoria puede facilitar en gran medida la minería de datos.

6.4 El lenguaje de los patrones. Expresividad

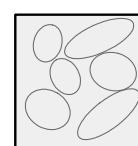
La característica más diferenciadora de los métodos de aprendizaje es la manera en la que se expresan los patrones aprendidos. Esta es, precisamente, la razón fundamental de por qué unos métodos van mejor para unos problemas que para otros. En realidad, cada método permite expresar mejor ciertos tipos de patrones. De ahí el hecho de que existan tantos métodos; la variedad de métodos permite capturar distintos tipos de patrones; si uno falla podemos probar con otros.



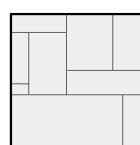
Perceptrón /
Discriminador lineal



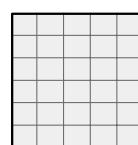
Redes neuronales
multicapa



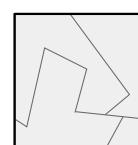
Radial Basis
Functions



Árbol de decisión:
C4.5/ID3/CART



Partición bayesiana
(con discretización)



Vecinos más
próximos, LVQ.

Figura 6.3. Representación de los tipos de patrones que son capaces de capturar distintos métodos.

En la Figura 6.3 podemos observar (de manera esquemática) qué tipos de patrones son capaces de expresar algunos de los métodos que veremos en los capítulos siguientes. Como se observa, muchos de estos métodos se basan en colocar fronteras, linderos o confines entre zonas. Cada una de las zonas es capaz de aglutinar una clase o un grupo. Los métodos anteriores se conocen popularmente como “cerca y rellena” (*fence & fill*). Otros métodos no establecen fronteras explícitamente, sino que se basan en el concepto de centros y clasifican / agrupan por la distancia a estos centros o zonas más densas. Por ejemplo, las funciones de base radial o los vecinos más próximos son de este estilo. Sea como sea, se consideran también métodos de “cerca y rellena”, ya que los individuos que están próximos suelen tener la misma clase o grupo.

Aun así, las diferencias son patentes entre unos métodos y otros, como se observa en la Figura 6.3. Algunos marcan los linderos mediante fronteras rectilíneas, a veces oblicuas a los ejes, otras veces paralelas. Otras veces son curvas o incluso circulares o elipsoidales. En algunos casos existe un único lindero para determinar las fronteras y a veces tenemos varios linderos combinados.

6.4.1 ¿Qué expresividad es necesaria? Subajuste y sobreajuste

A tenor de la variedad de métodos mostrados en la Figura 6.3 (que son sólo una muestra de todos los existentes) no debe existir ninguna figura geométrica que no pueda capturarse con buena aproximación por alguno o varios de los métodos anteriores.

El problema es que existen patrones que no dependen de la proximidad o similitud espacial o geométrica entre los individuos. Este tipo de patrones no pueden ser aprendidos por los métodos de “cerca y rellena”. Por ejemplo, la función de la Figura 6.4 se caracteriza precisamente porque cada valor de una clase está rodeado por elementos de la clase contraria. Esto puede parecer poco natural, porque, normalmente, es de esperar que los elementos se rodeen de elementos de su condición. Sin embargo, el patrón existe y las herramientas de minería de datos deberían ser capaces de capturarlo.

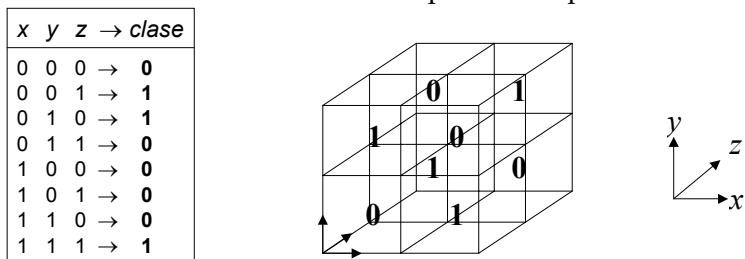


Figura 6.4. Problema de la paridad. Problema relacional sin zonas geométricas distinguibles.

La realidad es bien distinta y los métodos de “cerca y rellena” no pueden capturar este concepto. Los más expresivos (las redes neuronales, LVQ, etc.) pueden llegar a emular el patrón a un nivel de resolución dado, pero no son capaces de interpretar el patrón existente que no es otro que: “se retorna la paridad de los tres valores de entrada” (un “o exclusivo” entre ellos). Es decir, si aprendemos la función paridad con una red neuronal tenemos un patrón artificial, ya que la función paridad no queda definida a partir del número de atributos con un determinado valor sino como una combinación ‘arbitraria’ de pesos. Además, la red que calcula la paridad de un número de n dígitos (entradas) no es la misma que la que lo calcula para un número de $n+1$ dígitos (entradas).

La razón es que el patrón del problema de la paridad es “relacional”. El patrón depende de las *relaciones entre varios atributos*. Además, en este caso, el patrón está definido de tal manera que la distancia no puede ser aprovechada. En general, existen conceptos que dependen poco o nada de la cercanía geométrica de los individuos. En estos casos, los métodos “cerca y rellena” o basados en distancia no funcionarán bien. Existen, sin embargo, otros conceptos “relacionales” (como, por ejemplo, el concepto $x \leq y$) que, aunque no pueden ser capturados totalmente por los métodos “cerca y rellena” o basados en distancia, al menos sí que existe correspondencia entre el concepto y la cercanía geométrica.

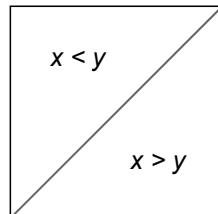


Figura 6.5. Problema relacional con zonas geométricas sencillas.

La expresividad relacional hace más bien referencia a las relaciones que se pueden establecer entre atributos y entre distintos ejemplos. De hecho, existen problemas en los que cada ejemplo consiste en un número variable de objetos y las relaciones entre estos objetos son importantes. En estos casos es necesario expresividad relacional. Además, cuando tenemos ejemplos con estructura, podemos necesitar *recursividad* para capturar el concepto a aprender. La recursividad se requiere cuando la profundidad (el nivel) de las relaciones no se conoce a priori (objetos que contienen o se relacionan con un número variable de objetos). De este tema trataremos en el Capítulo 12.

Es decir, existen muchas razones para que los métodos “cerca y rellena” o basados en distancia no funcionen bien, ya sea porque el patrón a aprender no sigue una pauta de similitud por cercanía, o porque el patrón requiere relaciones entre varios atributos, o porque los ejemplos tengan una estructura compleja, o porque haga falta recursividad, o por varias de estas razones a la vez.

En la Figura 6.6 se muestra un mismo problema clasificado con fronteras de expresividad diferente. En el de la parte izquierda, tenemos un discriminante lineal que lógicamente no puede separar limpiamente las dos clases. Como vemos el modelo no es capaz de ajustarse a los datos de ninguna manera, obteniendo un efecto de **subajuste** (*underfitting*).

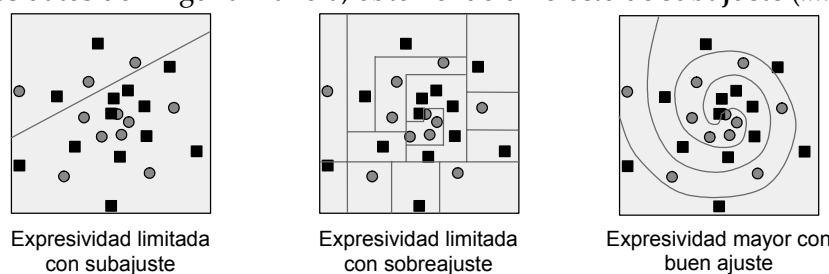


Figura 6.6. Diferencia entre capturar un patrón complejo por “rastering” y por “identificación”.

En el panel del medio de la misma figura tenemos un modelo que permite utilizar varias fronteras a la vez, pero éstas han de ser rectas y paralelas a los ejes (como un árbol de decisión). La clasificación realizada puede funcionar bien en algún caso de los no vistos,

pero en general realiza **sobreajuste** (*overfitting*) a los datos intentando realizar un “rasterizado” o pixelado por zonas. Las fronteras no corresponden con la regularidad o patrones *reales* existentes en los datos. Si la resolución es buena (y la densidad de ejemplos es alta) podemos tener una buena “imagen” del patrón, pero si hay pocos ejemplos, la imagen es más bien arbitraria, como se puede ver en el panel central de la Figura 6.6, especialmente en las zonas exteriores, que son menos densas. De hecho, cuando la ratio entre atributos (dimensiones) y ejemplos es alta, el espacio está muy vacío y la probabilidad de que ocurra este problema es mayor.

En cambio, si la expresividad permite capturar patrones más complejos podemos tener un delineamiento mejor sin sobreajustar, como se puede ver en la parte derecha de la Figura 6.6. El modelo de la derecha es capaz de identificar la regularidad de los datos, pero lo que es más importante, *toda y solamente* la regularidad existente. El problema es que la expresividad de la derecha no es alcanzada *directamente* por prácticamente ningún método de aprendizaje actual. Por ejemplo, ser capaz de tener fronteras en *espiral* sólo sería posible si la función espiral estuviera en el conocimiento previo y el método fuera lo suficientemente expresivo para poderla utilizar. La mayoría de métodos que son capaces de obtener la separación de la parte derecha de la Figura 6.6 lo hacen componiendo trozos o, dicho más formalmente, ajustándose localmente pero no globalmente.

Tener una alta expresividad no siempre es positivo; considérese una regresión ponderada local que se ajusta a los datos como se muestra en la Figura 6.7. En este caso, una regresión lineal sería, probablemente, un mejor modelo de los datos. No obstante, hay que clarificar que el problema no ocurre aquí por la mayor expresividad de la regresión ponderada local, sino en el buen o mal uso que se haga de ella.

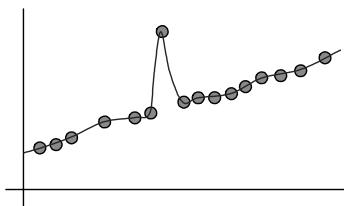


Figura 6.7. Ejemplo de sobreajuste por abuso de la expresividad.

Tanto para el problema del subajuste y del sobreajuste, tenemos la cuestión de que a priori (especialmente cuando hay miles de datos y muchas dimensiones) no sabemos qué tipo de expresividad nos va a ser necesaria y, en definitiva, si con los métodos tradicionales vamos a poder tener un buen modelo. Sería interesante poder determinar de qué manera un concepto depende de la proximidad geométrica de los elementos.

6.4.2 Medidas de separabilidad

Afortunadamente, existen una serie de funciones heurísticas que permiten determinar, aunque sea aproximadamente, la complejidad geométrica de un problema. Estas funciones, en realidad, permiten determinar el grado de *continuidad* o *separabilidad*, considerando una medida de distancia. Si la separabilidad es baja, debemos intentar métodos no basados en distancias.

La primera función heurística se denomina **separabilidad lineal** y fue introducida por Minsky y Papert (véase por ejemplo [Minsky & Papert 1969/1988]). Esta separabilidad se

define sencillamente de la siguiente manera: un concepto es separable linealmente si existe una función lineal (una recta en 2D, un plano en 3D o un hiperplano en más dimensiones) que separe los dos grupos o clases nítidamente. Existen variantes graduales (teniendo en cuenta la limpieza de cada zona) y variantes para más de dos clases. Por ejemplo, en la parte izquierda de la Figura 6.8, se muestra en la parte izquierda el grado de separabilidad para un problema en concreto, como el número de elementos bien clasificados respecto el total, utilizando el mejor discriminador lineal posible. Para obtener la función lineal generalmente se puede utilizar un perceptrón, como se verá en el Capítulo 13. No obstante, este tipo de separabilidad es muy pobre. La mayoría de técnicas desarrolladas basadas en distancia son capaces de aprender conceptos que no sean separables linealmente.

Otra función heurística más apropiada es la **separabilidad geométrica** (SG) [Thornton 1997]. Ésta se define de la siguiente manera:

$$SG(f) = \frac{\sum_{i=1}^n eq(f(e_i), f(nn(e_i)))}{n}$$

donde f es la función definida por los datos, n es el número de ejemplos, nn es el vecino más cercano (según una medida de distancia, por ejemplo la euclídea) y $eq(a,b)=1$ si $a=b$ y 0 en caso contrario.

Esta función ya ajusta bastante mejor a otro tipo de patrones que, aunque no sean lineales, sí pueden tener un gran componente geométrico. El valor para el mismo problema se muestra en la parte central de la Figura 6.8, como el porcentaje de elementos cuyo vecino más próximo es de la misma clase.

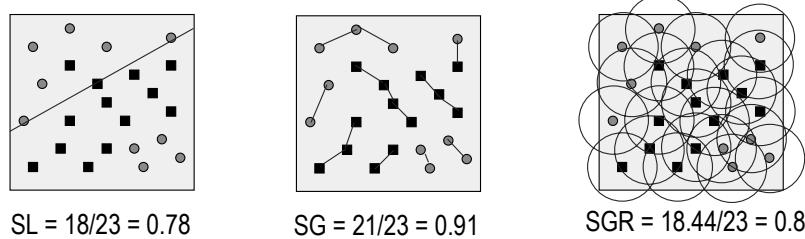


Figura 6.8. Distintos valores de separabilidad (separabilidad lineal, geométrica y geométrica radial).

El problema de la separabilidad geométrica es que depende mucho del número de ejemplos. Un espacio muy denso con fronteras muy complejas puede tener alta separabilidad debido, simplemente, a esta densidad. Una mejora de la función anterior es la **separabilidad geométrica radial** (SGR), que se define de la siguiente manera¹⁶:

$$SGR(f) = \frac{\sum_{i=1}^n \sum_{e_j: dist(e_i, e_j) < r} eq(f(e_i), f(e_j))}{n |e_j : dist(e_i, e_j) < r|}$$

Donde $dist$ representa la función distancia entre dos ejemplos y puede ser cualquier distancia definida (por ejemplo, la euclídea). El valor de la separabilidad geométrica radial para el mismo problema se muestra en la parte derecha de la Figura 6.8, como el porcentaje medio de los ejemplos tales que sus ejemplos próximos en un radio r son de la misma clase.

¹⁶ También se puede calcular una variante de la SGR que no incluya al propio punto.

Una cuestión en la función anterior es establecer un radio adecuado. Esto se puede estimar a partir de la densidad de los ejemplos.

Las funciones anteriores pueden utilizarse antes de *abordar* un problema de clasificación, para decidir a priori qué métodos pueden ir mejor. Por ejemplo, la separabilidad (lineal, geométrica y geométrica radial) del problema de la paridad es cercano a 0, mientras que la separabilidad de la función $x > y$, por ejemplo, es cercana a 1.

En la Figura 6.9 se muestra que estas funciones pueden utilizarse para ver entre los casos de alta separabilidad y los casos de separabilidad casi nula. En la parte izquierda, los grupos son fáciles de分开. Una simple línea es suficiente para separarlos. A medida que nos movemos hacia la derecha tenemos que si nos limitamos en modelos basados en distancias o en fronteras lineales, será mucho más difícil capturar los patrones. Por ejemplo, la distribución de la parte derecha es un perfecto tablero de ajedrez. Si usamos fronteras lineales, necesitaremos decenas de ellas para poder repartir los ejemplos entre una y otra clase. Cuanto más a la derecha, tenemos conceptos más “relacionales” y menos “geométricos”.

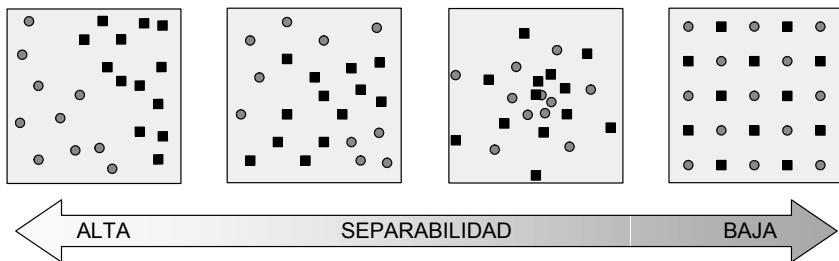


Figura 6.9. Dificultad de separación creciente.

En los ejemplos anteriores, hemos estado asumiendo que el problema era de clasificación. Para problemas de regresión se pueden adaptar fácilmente las medidas anteriores (en especial las geométricas) sustituyendo la función *eq* por el error cuadrático medio de dos ejemplos. Para problemas de agrupamiento, el discurso de complejidad anterior también es válido, pero necesariamente se deben utilizar otras medidas (márgenes entre grupos, distancias entre centros, etc.), que dependen más del algoritmo y por tanto no se puede hacer esta evaluación previa, sino que, más bien, debe ser iterativa. También hemos supuesto que los atributos son numéricos. Las definiciones anteriores son igualmente válidas si somos capaces de definir una función de distancia que incluya también los atributos nominales.

El concepto de no linealidad está relacionado también con la dimensión VC que hemos comentado anteriormente. No obstante esta dimensión es en algunos casos más difícil de estimar.

6.4.3 Técnicas para aumentar la expresividad

Una vez somos capaces de determinar que la separabilidad de un problema es baja y, muy probablemente, las técnicas de “cerca y rellena” o basadas en distancia van a funcionar mal, por su expresividad limitada, la cuestión que aparece es, claramente, ¿cómo podemos aumentar la expresividad?, o bien, ¿qué métodos debemos utilizar en estos casos?

A continuación enumeramos una serie de técnicas para aumentar la expresividad:

- Aumento de la dimensionalidad y/o creación de atributos: algunos conceptos no pueden aprenderse con los atributos originales. Generalmente, si aumentamos la dimensionalidad podemos conseguir separabilidad lineal o más sencilla cuando antes no lo había, al tener más dimensiones con las que definir el separador (en el caso de más de tres dimensiones, un hiperplano). Por ejemplo, ésta es la técnica utilizada por las funciones de núcleo (*kernel*), que transforman el espacio a una dimensionalidad mayor para poder definir funciones discriminadoras de margen en los métodos de máquinas de soporte vectorial (que veremos en el Capítulo 14). No obstante, se pueden probar aumentos de la dimensionalidad más sencillos, como la numerización “1 a n ” (vista en el Capítulo 4), para convertir un valor nominal en n numéricos, o la creación de atributos mediante generación (*pick&mix*) que relacionen dos o más atributos (con lo que obtenemos relaciones no lineales o incluso no geométricas entre varios atributos), también visto en el Capítulo 4.
- Localidad: para espacios de hipótesis iguales, los métodos perezosos y los métodos locales pueden representar funciones más complejas. Por ejemplo, la regresión ponderada local es mucho más expresiva que la regresión global. Existen problemas que globalmente son muy poco separables, pero a un nivel más local sí que pueden aproximarse mejor. El riesgo de aplicar métodos locales es que podemos caer en el sobreajuste, es decir, realizar un *rastering* de los datos sin patrón alguno.
- Combinación de modelos y métodos: Otra manera de aumentar la expresividad es mediante la combinación de modelos, es decir realizar modelos que estén formados por la combinación de varios modelos. En realidad, algunos métodos simples son en realidad combinaciones de fronteras. La combinación de modelos va más allá y pondera las votaciones de varios modelos, obteniendo un suavizado y una generalización que supera la expresividad de cada uno de los modelos por separado. Esta capacidad la veremos en el Capítulo 18.

En algunos casos, las aproximaciones anteriores consiguen mejorar la calidad de los modelos y capturar la mayor parte de la regularidad *real* existente en los datos. Incluso en los casos en los que estas aproximaciones funcionen, no obstante, hay que tener en cuenta que el patrón o patrones obtenidos dependen de las nuevas dimensiones, de zonas locales o de una combinación de modelos, con lo que, muchas veces, no se pueden examinar fácilmente y no se es capaz de extraer *conocimiento* sobre estos patrones, ya que no son comprensibles.

Ya sea porque las aproximaciones anteriores no funcionen o bien porque sí funcionen pero deseemos analizar y comprender los patrones extraídos, existe una alternativa: el uso de lenguajes universales para expresar los modelos.

Cuando nos referimos a un lenguaje (de capacidad) universal, nos referimos a un lenguaje que pueda expresar cualquier concepto o, dicho más formalmente, que pueda expresar cualquier función computable. Los lenguajes de programación y el lenguaje natural (por ejemplo el castellano) son lenguajes universales.

Los lenguajes universales permiten establecer relaciones de cualquier complejidad; permiten relacionar atributos del mismo ejemplo, relacionar varios ejemplos, utilizar funciones auxiliares en la definición de otras, etc.

El aprendizaje inductivo utilizando lenguajes universales es relativamente reciente. Aunque existen aproximaciones para el aprendizaje de gramáticas recursivas en la década de los 60 y 70, en realidad los primeros pinitos de aprendizaje con un lenguaje universal aparecen en los años 70, utilizando LISP como lenguaje de representación de ejemplos e hipótesis. Es decir, los ejemplos se presentaban en LISP y el algoritmo de aprendizaje obtenía un programa en LISP que, al ejecutarse, obtenía los ejemplos que habían servido para su aprendizaje.

En realidad estos sistemas tenían muchas limitaciones y no es hasta la década de los 80 (con los trabajos de Shapiro y su sistema MIP [Shapiro 1983]) y muy especialmente en la década de los 90, con el desarrollo del área de la programación lógica inductiva (ILP), en el que se empieza a considerar seriamente su uso para aplicaciones prácticas y, en nuestro caso, para la minería de datos. La ILP, como veremos en el Capítulo 12, se basa en utilizar la lógica (o lenguajes de programación basados en la lógica, como el Prolog) como lenguaje de representación. En concreto, los ejemplos, conocimiento previo e hipótesis se pueden expresar en forma de reglas lógicas (en Prolog, por ejemplo).

En la Figura 6.10 se representa un modelo recursivo utilizando reglas de Prolog para representar el concepto de “ser familia de” (*relative*). Aunque no se conozca Prolog, no es difícil entender las reglas si se sabe que el símbolo “:-” se interpreta como “si” o como una implicación de derecha o izquierda. Por ejemplo, la primera regla indica que X es ancestro de Y si X es padre (o madre) de Y. Nótese que este modelo permite relacionar distintos objetos y, además, es recursivo (el predicado *ancestor* en la segunda regla aparece tanto en la cabeza (parte izquierda) de la regla como en el cuerpo (parte derecha)).

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
relative(X,Y) :- ancestor(X,W), ancestor(Y,W).
```

Figura 6.10. Modelo de gran expresividad representado mediante reglas lógicas.

Debido a la correspondencia entre bases de datos relacionales y teorías lógicas, esta área ha sido rebautizada recientemente como Minería de Datos Relacional y existen grandes posibilidades y expectativas puestas en su desarrollo, pese a la notoria complejidad computacional (intratabilidad) que supone aumentar la expresividad a nivel universal. El Capítulo 12 se dedica específicamente a este tipo de aprendizaje relacional y otros lenguajes declarativos de capacidad universal.

6.5 Breve comparación de métodos

En el Apartado 6.2 vimos diferentes métodos para resolver las tareas también expuestas en el mismo apartado. En las secciones siguientes hemos comentado ciertos rasgos fundamentales en toda técnica de minería de datos: comprensibilidad de los modelos obtenidos (en el caso de que obtengan modelo), expresividad, precisión, eficiencia del método, estabilidad de muestra, robustez al ruido, facilidad de uso, etc.

En los capítulos siguientes describiremos las técnicas más importantes de minería de datos y destacaremos las ventajas e inconvenientes de cada una de ellas según los rasgos anteriores, o, de una manera más precisa, las ventajas e inconvenientes de cada algoritmo en concreto. No obstante, y a título de orientación, presentamos a continuación una muy

breve comparación de algunos de los métodos más señalados, destacando sus rasgos más importantes.

- Técnicas de modelización estadística: las técnicas paramétricas son muy eficientes, disponibles en multitud de herramientas y en muchos casos comprensibles. Las técnicas no paramétricas son mucho más expresivas, aunque suelen perder en comprensibilidad debido a la utilización de núcleos y de aproximaciones locales. Las técnicas no paramétricas son bastante más ineficientes para grandes volúmenes de datos. Tanto las técnicas paramétricas como no paramétricas trabajan más cómodamente con datos numéricos (cuantitativos, en la terminología estadística), aunque mediante numerización se pueden utilizar con datos nominales (categóricos, en la terminología estadística).
- Técnicas bayesianas: son fáciles de usar, muy eficientes, pueden tratar muchos atributos (cientos o miles), son muy robustos al ruido, la expresividad es limitada y depende de la discretización, son estables a la muestra. Al igual que las técnicas anteriores no construyen modelos, sólo estiman una serie de probabilidades (a excepción de los modelos gráficos probabilísticos, donde la red bayesiana creada puede ser muy informativa).
- Técnicas basadas en árboles de decisión y sistemas de aprendizaje de reglas: son una de las estrellas de la minería de datos. Son fáciles de usar, admiten atributos discretos y continuos, tratan bien los atributos no significativos, los valores faltantes y el ruido (mediante la poda). Son bastante eficientes y obtienen resultados para clasificación bastante buenos (para regresión y agrupamiento existen variantes pero no son muy utilizadas), aunque la mayor ventaja de estos métodos es su inteligibilidad; los métodos obtenidos se pueden expresar como conjuntos de reglas. Uno de los inconvenientes de los árboles de decisión es su limitada expresividad y que son inestables ante variaciones de la muestra.
- Técnicas relacionales y declarativas: son técnicas muy expresivas que permiten tratar datos con estructuras y capturas patrones relaciones y recursivos, así como expresar el conocimiento previo en forma de reglas. Los mayores inconvenientes de estas técnicas son la dificultad de manejo (hay que saber expresar los ejemplos convenientes) y la poca eficiencia, en general, de los métodos existentes. Sobre el resto de rasgos (robustez al ruido, estabilidad ante la muestra, precisión, etc.), la variedad de métodos difieren en muchos de ellos. Quizá la ventaja más importante, además de la gran expresividad, sea que los modelos son comprensibles.
- Técnicas basadas en redes neuronales artificiales: aunque existen redes con valores por defecto y donde el número de capas y de nodos internos se calcula automáticamente, requieren de cierta experiencia para poderles sacar el máximo partido. Su ventaja principal es que, cuando están bien ajustadas, obtienen precisiones muy altas. Además son muy expresivas y permiten capturar modelos no lineales. Entre sus inconvenientes se suelen nombrar su sensibilidad a valores anómalos (aunque son robustos al ruido no extremo y a los atributos no significativos), necesitan muchos ejemplos para el aprendizaje y son relativamente lentas y, fundamentalmente, su incomprensibilidad. Las redes neuronales se consideran generalmente cajas negras, que obtienen unas salidas a partir de unas entradas pero sin saber cómo.

- Técnicas basadas en núcleo y máquinas de soporte vectorial: son técnicas muy eficientes que permiten trabajar con datos con alta dimensionalidad. Proporcionan modelos muy precisos. El inconveniente más importante es que a veces es necesario elegir una buena función de núcleo (*kernel*) para obtener buenos resultados. Además, esta función transforma los atributos iniciales, con lo que el modelo final se define en función de los atributos transformados, perdiéndose la comprensibilidad.
- Técnicas estocásticas y difusas: este tipo de técnicas son demasiado diversas para intentar dar unas características conjuntas a todas ellas. Quizás una característica de las técnicas estocásticas (y las evolutivas y genéticas en particular) es su coste computacional; en general requieren bastante tiempo (y memoria) para llegar a una convergencia. Entre sus ventajas está la flexibilidad de poder cambiar el heurístico o el criterio de calidad sin cambiar el algoritmo. En cuanto a las técnicas difusas, una característica positiva es su expresividad, debido al uso de fronteras difusas, en vez de fronteras estrictas.
- Técnicas basadas en casos, en densidad o distancia: son fáciles de usar en general, eficientes si el número de ejemplos no es excesivamente grande. Al ser técnicas generalmente locales tienen bastante expresividad. Al estar basadas en distancias, no digieren bien los atributos no significativos, ya que se aumenta la dimensionalidad y aparecen distancias ficticias. No construyen modelo y, por tanto, no producen un conocimiento comprensible.

La relación anterior es somera y puede no ser aplicable a todas las técnicas del tipo, en especial a los híbridos o a mejoras particulares que intentan subsanar las dificultades principales de cada tipo.

Una vez comentadas las características principales, los siguientes capítulos describen con detenimiento estas y otras técnicas.

Capítulo 7

MODELIZACIÓN ESTADÍSTICA

PARAMÉTRICA

Tomàs Aluja y Pedro Delicado

En este capítulo y el siguiente tratamos el tema del aprendizaje a partir de los datos utilizando el razonamiento estadístico. En Estadística se viene aprendiendo de los datos desde finales del siglo XIX. Sin embargo, ha sido con la aparición de los problemas propiciados por la revolución informática, cuando ciertos métodos estadísticos ya conocidos han cobrado todo su valor y han aparecido nuevos métodos para hacer frente a los retos planteados.

En este primer capítulo abordamos la modelización lineal paramétrica, en un contexto de minería de datos, tanto de variables numéricas, dando lugar a métodos para el problema de regresión, como de variables categóricas, dando lugar a métodos para el problema de discriminación (o clasificación supervisada). La regresión y la clasificación son dos de las tareas más frecuentes en minería de datos. Ésta es la parte central del capítulo y conforma la teoría clásica de modelización estadística. El capítulo también incluye extensiones, como son la utilización de variables no correlacionadas y los modelos lineales generalizados.

En el siguiente capítulo se presenta la modelización no paramétrica, que permite construir modelos más flexibles, capaces de modelar fenómenos complejos. Tanto en este capítulo como en el siguiente no entramos en los conceptos básicos de estadística, como son estadísticos descriptivos como la media, la varianza, la correlación, etc., o las representaciones gráficas, ni tampoco en el concepto de prueba de hipótesis ni la presentación de las distribuciones estadísticas más frecuentes (z , t , F y χ^2), que damos por conocidas¹⁷; el lector interesado podrá referirse a [Peña 2001] para el detalle de estos temas.

¹⁷ Las damos por conocidas al menos en el sentido de que el lector sea capaz de obtener los valores a distintos niveles de confianza, ya sea mirando en tablas en cualquier libro de estadística o, de una manera más cómoda, usando alguna herramienta informática.

7.1 Concepto de modelización estadística

El objetivo de la modelización estadística consiste en explicar el comportamiento de una variable a partir del conocimiento de otras. Subyacente al concepto de modelización está la idea de que una variable tiene una cierta variabilidad y que esta variabilidad está relacionada con el comportamiento de otras variables, por ejemplo, el saldo total bancario de las personas de una cierta edad, con un mismo nivel profesional y residiendo en el mismo distrito no es igual para todos ellos sino que sigue una cierta distribución, pero no hay duda de que si sabemos la edad de una persona, su nivel profesional y su distrito de residencia podremos aproximar bastante fidedignamente su saldo bancario.

A la variable que queremos estudiar, en este ejemplo el saldo bancario, se le denomina variable de salida (*output*), de respuesta, endógena... según el contexto en que nos encontremos y se denota por la letra y , mientras que las variables edad, nivel profesional, distrito, se denominan variables de entrada (*input*), predictoras, explicativas, regresores, exógena... y se denotan por x_j . En este capítulo utilizamos la terminología de variable de respuesta y variables explicativas o regresores para referirnos a ambos conjuntos de variables.

La modelización estadística es seguramente la técnica estadística más empleada. Puede utilizarse tanto si el problema planteado consiste en la predicción de una cierta variable de respuesta, como si se trata de encontrar un modelo causal, en cuyo caso las variables explicativas son *causa* de la variación de la respuesta, permitiendo por tanto su intervención modificando las variables explicativas (si es posible tal modificación). En minería de datos una parte importante de problemas es de predicción, de forma que basta que las variables explicativas estén asociadas a la variable de respuesta, para que sabiendo el valor que toman aquellas podamos hacer predicciones sobre el valor que tomará la variable de respuesta.

Por otro lado, en regresión es normal suponer las variables explicativas fijadas por un diseño experimental, esto es, no aleatorias. Éste no es el caso en minería de datos, donde las variables explicativas se obtienen en contextos observacionales, por ejemplo una base de datos. Esto, sin embargo, no será un problema, dado que en regresión se estudia el comportamiento de la respuesta *condicional* a las variables explicativas.

La modelización estadística consiste pues en descomponer los valores que toma la variable y para cada individuo en dos componentes, uno función de las variables explicativas y otro que es específico al individuo en cuestión:

$$y_i = r(x_{i1}, \dots, x_{ip}) + \varepsilon_i,$$

donde r es la función relacionando los valores de la variable de respuesta con las explicativas, mientras que ε_i es la parte específica al individuo i que no viene explicado por ninguna variable.

La función r representa pues la parte determinista, estructural del modelo, que nos permite explicar el comportamiento de la variable de respuesta y hacer predicciones sobre ella, mientras que la parte específica representa la parte impredecible, aleatoria y se denomina, sin ninguna connotación moral, término de error. Este término se caracteriza a partir de una distribución de probabilidad generadora de los distintos valores para cada

individuo, que sin pérdida de generalidad podemos suponer centrado, es decir, su valor esperado es 0:

$$E[\varepsilon_i] = E[y_i - r(x_{il}, \dots, x_{ip})] = 0$$

Los modelos estadísticos se diferencian respecto al tipo de la variable de respuesta, que puede ser numérica, binaria o categórica, el tipo de variables explicativas, que pueden ser numéricas o categóricas, respecto de la función r y respecto a la distribución de probabilidad del término de error.

7.2 Modelo de regresión

Hablamos de modelo de regresión cuando la variable de respuesta y las variables explicativas son todas ellas cuantitativas. Si sólo disponemos de una variable explicativa hablamos de *regresión simple*, mientras que si disponemos de varias variables explicativas se trata de un problema de *regresión múltiple*.

Para visualizar la relación entre la variable de respuesta y una variable explicativa, obtendremos el diagrama bivariante entre ambas variables. La forma de dicho diagrama aporta información sobre el tipo de relación entre la variable de respuesta y la variable explicativa, esto es sobre la función r .

Veamos un ejemplo de diagrama bivariante. Utilizamos el archivo *Boston Housing Data*, que representa datos de las características de las viviendas de 506 barrios de Boston (se puede encontrar más información sobre este conjunto de datos en el Apéndice B). El diagrama bivariante de la variable *value*, valor medio de las viviendas (en miles de dólares), en función del porcentaje de residentes en la zona de clase baja representado por la variable *lstat*, muestra claramente una relación no lineal, como se muestra en la Figura 7.1.

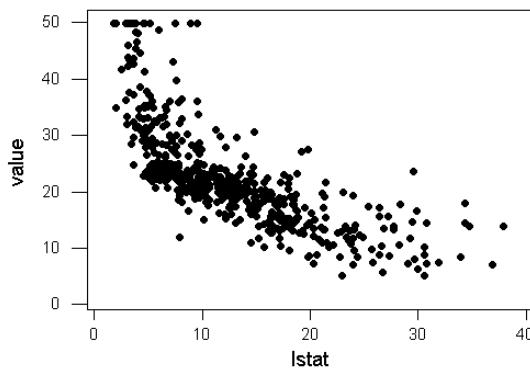


Figura 7.1 Diagrama bivariante entre dos variables.

Vemos que el valor de las viviendas desciende muy rápidamente al incrementarse el porcentaje de clase baja hasta el 7 por ciento, para después descender de forma menos brusca y terminar estabilizándose a partir del 25 por ciento de clase baja.

Para estimar dicha función r , buscaremos una función tal que, en promedio, las desviaciones al cuadrado respecto de los puntos sean mínimas. Esto es, minimizaremos el error cuadrático medio:

$$\min_r E \left[(y_i - r(x_{il}, \dots, x_{ip}))^2 \right]$$

El mínimo de esta función corresponde a la media condicional de y_i respecto de los valores de las variables explicativas.

$$r(x_{il}, \dots, x_{ip}) = E \left[y_i | x_{il}, \dots, x_{ip} \right].$$

A esta función r se le llama *función de regresión*.

7.2.1 Regresión lineal

La función de regresión más simple es la lineal, esto es, cada variable explicativa participa de forma aditiva y constante para todo el dominio observado en la formación de la respuesta:

$$E \left[y_i | x_{il}, \dots, x_{ip} \right] = \beta_0 + \beta_1 x_{il} + \dots + \beta_p x_{ip}$$

Por tanto, el modelo de regresión lineal se escribe:

$$y_i = \beta_0 + \beta_1 x_{il} + \dots + \beta_p x_{ip} + \varepsilon_i$$

Por ejemplo, si ajustamos este modelo a los datos de la Figura 7.1, obtenemos la recta de regresión que se muestra en la Figura 7.2.

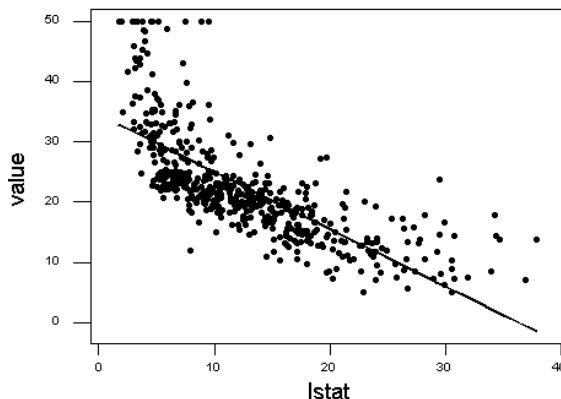


Figura 7.2. Ajuste lineal del valor medio de las viviendas en función del porcentaje de clase baja.

Se observa que para los porcentajes de clase baja (lstat) inferiores al 7 por ciento el modelo ajustado predice valores de las viviendas muy inferiores a los reales, mientras que para los porcentajes intermedios de clase baja, el modelo ajustado sobrevalora los precios de las viviendas, y para los valores de clase baja superiores al 25 por ciento, de nuevo los valores ajustados son más bajos que los observados. Claramente el ajuste obtenido con este modelo es un mal ajuste.

La suposición de linealidad puede parecer pues muy restrictiva pero en realidad no lo es tanto. Si el número de individuos con que se cuenta es pequeño, los modelos lineales muestran una capacidad de generalización mayor que otros modelos más sofisticados y flexibles, debido a la excesiva dependencia de estos últimos respecto de los datos utilizados. Por otro lado las variables explicativas utilizadas pueden ser transformaciones de las originales, lo que amplía la versatilidad del modelo lineal. Por ejemplo pueden ser transformaciones logarítmicas o de raíz cuadrada para normalizar las variables originales, o pueden ser transformaciones polinómicas (x, x^2, x^3, \dots) para ajustar una función r no lineal.

También podemos tomar como variables explicativas las variables binarias fruto de la recodificación en intervalos de las variables originales (*binning*) y aproximar relaciones no lineales cualesquiera mediante funciones en escalera, o bien podemos incluir en el modelo términos de interacción entre variables explicativas mediante el respectivo producto ($x_2 \cdot x_3$), ampliando la flexibilidad del método a relaciones no aditivas (muchas de estas posibilidades de transformación se vieron en el Capítulo 4). La linealidad del modelo se refiere pues a la forma de la función de regresión, independientemente de si las variables explicativas son una transformación no lineal del espacio original.

Por otro lado existen modelos no lineales de amplio uso que se convierten en modelos lineales con simples transformaciones. Por ejemplo, el modelo econométrico de elasticidad constante o doble logarítmico (representado en la parte izquierda de la Figura 7.3), $y = \alpha x^\beta$, siendo β el coeficiente de elasticidad, se convierte en un modelo lineal tomando logaritmos de ambas variables: $\hat{y} = \ln(y)$, $\hat{x} = \ln(x)$, obteniendo $\hat{y} = \ln(\alpha) + \beta \hat{x}$. Este modelo expresa que una variación porcentual de la variable explicativa produce una variación porcentual constante en la variable de respuesta.

El modelo de crecimiento exponencial $y = \alpha \exp\{\beta x\}$ (representado en la parte central de la Figura 7.3), se linealiza tomando logaritmos de la variable de respuesta $\hat{y} = \ln(y)$, resultando $\hat{y} = \ln(\alpha) + \beta x$. En este modelo, variaciones absolutas de la variable explicativa producen una variación porcentual constante en la variable de respuesta.

El modelo logístico, que indica la tasa de variación entre cero y uno de una respuesta (representado en la parte derecha de la Figura 7.2), definida por:

$$y = \frac{\exp\{\alpha + \beta x\}}{1 + \exp\{\alpha + \beta x\}},$$

se linealiza mediante la transformación:

$$\hat{y} = \ln \frac{y}{1 - y},$$

obteniendo entonces $\hat{y} = \alpha + \beta x$.

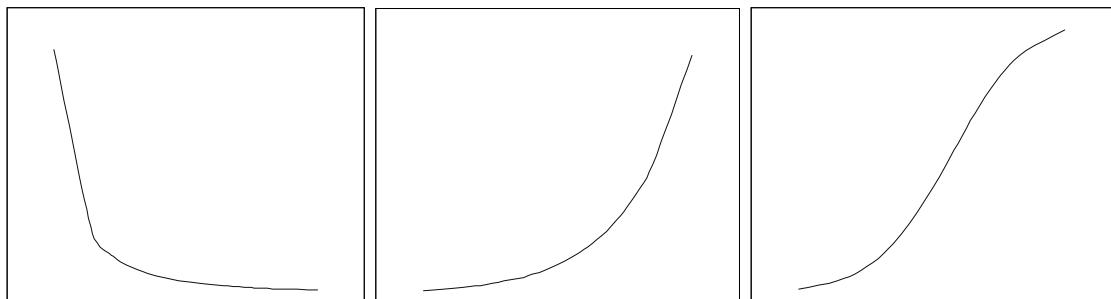


Figura 7.3. Modelo de elasticidad constante, crecimiento exponencial y logístico.

El modelo logístico se aplica en situaciones en que el modelo de crecimiento exponencial indefinido no es adecuado por llegar a situaciones de saturación en el crecimiento.

En el ejemplo anterior, en el que se relacionaba el valor de las viviendas `value`, con el porcentaje de clase baja `lstat`, el gráfico de la Figura 7.1 sugiere un modelo próximo al doble logarítmico, luego tomando logaritmos de ambas variables, obtenemos el ajuste de la Figura 7.4.

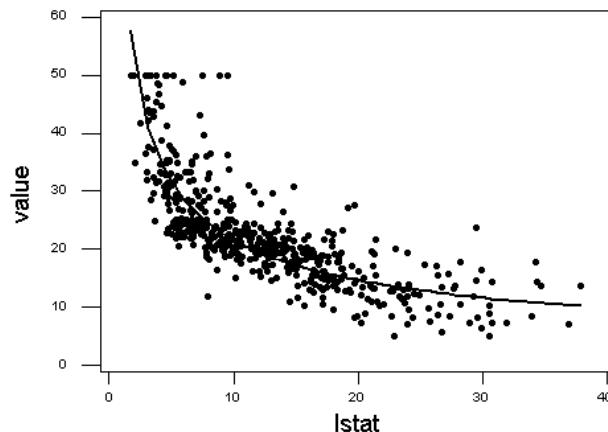


Figura 7.4. Ajuste doble logarítmico del valor de las viviendas respecto del porcentaje de clase baja.

Estimación de la función de regresión lineal

Estimar la función de regresión lineal significa estimar los coeficientes β_j . Para ello dispondremos de una muestra de aprendizaje de valores de la variable de respuesta con sus correspondientes variables explicativas X .

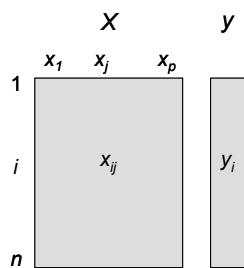


Figura 7.5. Variables explicativas y variable de respuesta de la muestra de aprendizaje.

Para estimar dichos coeficientes usamos el mismo criterio utilizado para definir la función de regresión, aplicado ahora a los datos muestrales:

$$\min_{b_0, \dots, b_p} \sum_{i=1}^n \left(y_i - b_0 - \sum_{j=1}^p b_j x_{ij} \right)^2 = \sum_{i=1}^n e_i^2$$

Llamamos residuos a los errores calculados con los datos muestrales. Por tanto, el criterio de estimación es la minimización de la suma del cuadrado de los residuos (SCR).

Esta minimización tiene una interpretación geométrica simple. Llamemos \hat{y}_i al valor ajustado:

$$\hat{y}_i = b_0 + b_1 x_{i1} + \dots + b_p x_{ip}$$

Por tanto, el ajuste mínimo cuadrático se escribe

$$y_i = \hat{y}_i + e_i \quad i = 1, \dots, n$$

el cual, expresado en notación matricial, queda:

$$\begin{matrix} y & \hat{y} & e \\ \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} & = & \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} & + \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} \end{matrix}$$

Donde y es el vector cuyos términos son los valores observados de la variable de respuesta, \hat{y} el vector de valores ajustados (vector de ajuste) y e el vector de residuos.

Esto es, descomponemos el vector y en suma de dos términos, uno que procuramos sea un vector lo más parecido posible a y , y que necesariamente debe estar contenido en el espacio generado por las variables explicativas (por ser combinación lineal de éstas) y otro vector de residuos, diferencia entre los dos anteriores. Claramente, hacer mínima la cantidad SCR es hacer mínima la longitud del vector residual. Y la forma de obtener un vector de residuos mínimo es mediante la proyección ortogonal de la variable de respuesta sobre el espacio generado por las variables explicativas (Figura 7.6).

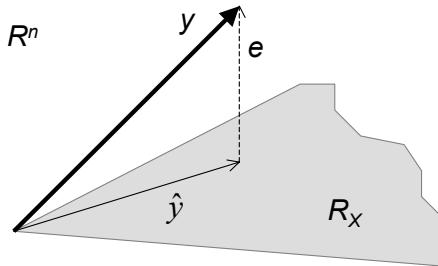


Figura 7.6. Proyección ortogonal de la variable de respuesta sobre el subespacio de las variables explicativas.

Por tanto, el vector ajustado debe pertenecer al espacio R_X generado por las variables independientes $\hat{y} = Xb$ y debe ser ortogonal al vector de residuos $\langle \hat{y}, y - \hat{y} \rangle = 0$. Desarrollando esta expresión se llega a que $b'X'y = b'X'Xb$ y por tanto suponiendo X de rango completo:

$$b = (X'X)^{-1}X'y$$

Así, el vector ajustado es

$$\hat{y} = Xb = X(X'X)^{-1}X'y = Hy$$

A la matriz H se le llama matriz sombrero (*hat* en inglés) porque pone el sombrero (^) en la y . H es pues la matriz de proyección ortogonal de y sobre el espacio R_X generado por las variables x_j .

Análogamente podemos ver que el vector de residuos e también se obtiene por proyección ortogonal de la variable de respuesta y :

$$e = y - \hat{y} = (I - H)y,$$

siendo I la matriz identidad, y $I - H$ el operador de proyección sobre el espacio R_X^\perp , ortogonal al generado por los vectores x_j .

Puede verse que la estimación mínima cuadrática es un estimador insesgado de la función de regresión: $E[b] = (X'X)^{-1}X'E[y] = (X'X)^{-1}X'X\beta = \beta$ y por tanto,

$$E[\hat{y}] = X\beta = E[y|x_1, \dots, x_p]$$

Respecto de la varianza de los estimadores se tiene el siguiente resultado:

$$b = (X'X)^{-1} X' (X\beta + \varepsilon) = \beta + (X'X)^{-1} X' \varepsilon,$$

$$\text{var}(b) = E[(b - \beta)(b - \beta)'] = E[(X'X)^{-1} X' \varepsilon \varepsilon' X (X'X)^{-1}] = \sigma^2 (X'X)^{-1}$$

Los coeficientes b_j cumplen la propiedad de ser óptimos (no sesgados y de mínima varianza) dentro de los estimadores lineales de los valores observados y_i (Teorema de Gauss-Markov).

7.2.2 Interpretación de los coeficientes

A efectos prácticos el término b_0 puede tomarse como el coeficiente del vector constante unidad, $u = (1, \dots, 1)'$, e incluir este vector dentro de la matriz X de las variables explicativas. Este término refleja el valor de la variable de respuesta cuando las variables explicativas valen 0.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}$$

Es fácil demostrar que si trabajamos con *todas* las variables centradas, el término independiente es nulo. El vector de residuos es ortogonal a todas las variables generadoras de R_X , y en particular al vector unidad u , luego $u'e = \sum_{i=1}^n e_i = 0$. Esto es, los residuos están centrados. Como consecuencia de ello tenemos, sumando para todos los individuos:

$$\sum_{i=1}^n y_i = \sum_{i=1}^n \hat{y}_i.$$

Por tanto, las medias del vector de respuesta y y del vector ajustado coinciden, por lo que, despejando el término independiente, tenemos $b_0 = \bar{y} - b_1 \bar{x}_1 - \dots - b_p \bar{x}_p$.

En lo que sigue, suponemos sin pérdida de generalidad que todas las variables vienen centradas¹⁸, en caso de no ser así deberemos añadir el vector unidad y su término correspondiente en las ecuaciones. En cualquier caso, el valor del término independiente siempre lo podemos calcular con la fórmula anterior.

El resto de coeficientes indican la influencia de cada variable explicativa en la variable de respuesta (b_j indica la variación en y cuando la variable x_j incrementa una unidad¹⁹) cuando en el modelo se mantienen constantes el resto de variables. Esto es, en un modelo sobre predicción del peso de las personas a partir de su altura y amplitud de brazos, posiblemente encontraríamos coeficientes más pequeños que al hacer la regresión del peso sobre cada variable explicativa por separado, porque saber la amplitud de brazos cuando ya se sabe la altura añade un conocimiento marginal pequeño sobre el peso de la persona y viceversa.

¹⁸ Calcular la media de una variable equivale a hacer la regresión de esta variable respecto el vector unidad. De este modo, centrar una variable (en este caso, la variable de respuesta y) significa aplicar el operador de proyección ortogonal a u :

$$(I - u(u'u)^{-1}u')y = y - \frac{1}{n}u \sum_{i=1}^n y_i = y - \bar{y}$$

y por tanto perder un grado de libertad (la dimensión correspondiente al vector unidad u).

¹⁹ O de otra forma, b_j es la derivada de y respecto x_j .

Por ejemplo, el ajuste obtenido en la Figura 7.4 entre el logaritmo del valor de la vivienda, denotado por `l1stat`, y el logaritmo del porcentaje de clase baja `lvalue`, será:

The regression equation is
`lvalue = 4.36 - 0.560 l1stat`

La interpretación de los coeficientes debe hacerse de acuerdo con las transformaciones efectuadas. Un incremento de una unidad en el logaritmo de `l1stat`, produce un decremento de 0,560 unidades del logaritmo del valor de la vivienda. O expresado en variaciones porcentuales de las variables originales, un incremento de un 1% de clase baja provoca un descenso de los precios del 0,56 por ciento (este coeficiente es la elasticidad de los precios respecto al porcentaje de clase baja residiendo en el barrio). El término independiente 4,36, refleja el precio (en logaritmos) de las viviendas en un barrio con el cero por ciento de clase baja y debe entenderse simplemente como la constante necesaria en el modelo para obtener un buen ajuste en el rango de valores observados de la variable explicativa, puesto que en los barrios de Boston considerados no hay ninguno con un cero por ciento de clase baja.

El porcentaje de clase baja es un buen indicador de estatus social y por tanto del nivel de precios de los barrios, por lo que será también un buen predictor para los precios de las viviendas. Sin embargo, podemos intentar mejorar el ajuste, incluyendo nuevas variables. Por ejemplo, podemos añadir en el modelo regresores como el índice de criminalidad (variable `crim`), y el número de habitaciones de las viviendas (variable `room`). El índice de criminalidad permite detectar las zonas conflictivas y por tanto matizar mejor el valor del suelo para cada barrio, mientras que el número de habitaciones es un indicador de la superficie de las viviendas y por tanto puede actuar como variable sustituta (*proxy*) del valor de construcción de las viviendas. La ecuación del ajuste que se obtiene así es ésta:

The regression equation is
`lvalue = 3.43 - 0.415 l1stat - 0.0118 crim + 0.100 room`

Observamos en particular que el coeficiente de `l1stat` ha cambiado respecto el ajuste precedente. Ello es debido a que ahora los coeficientes dan el efecto condicional, esto es, indican el cambio en la variable de respuesta para unos valores fijados de las otras variables del modelo. Vemos que ahora el decremento en el precio de la vivienda es del 0,415 por ciento cuando el porcentaje de clase baja se incrementa un uno por ciento. Respecto de los coeficientes de las otras variables, dado que no se ha efectuado transformación alguna sobre ellas, deben interpretarse en el sentido de que un incremento absoluto de una unidad de la variable explicativa provoca un incremento porcentual constante en la variable de respuesta. Esto es, un incremento de una unidad en el índice de criminalidad produce un decremento de 0,0118 en el logaritmo del valor de la vivienda, o sea, el precio baja un 1,2 por ciento ($e^{-0,0118}=0,988=1-0,012$). Mientras que disponer de una habitación más en la vivienda, implica un incremento de 0,1 unidades del logaritmo del valor de la vivienda, o sea, un incremento del 10,5 por ciento ($e^{0,1}=1,105$) del precio de la vivienda²⁰.

²⁰ La escala logarítmica permite pasar de incrementos constantes en esta escala a incrementos porcentuales constantes en la escala de la variable original.

La magnitud de los coeficientes b_j depende también de la unidad de medida de la variable explicativa, lógicamente no será indiferente medir una variable económica en euros, en dólares o en pesos. Una forma para poder comparar los coeficientes entre sí es medir *todas* las variables (incluida la de respuesta) en la misma unidad. Esta unidad común a todas las variables es el número de desviaciones tipo que una observación se aleja de su media, es decir, realizamos una tipificación, como vimos en el Capítulo 4 (Sección 4.5):

$$z_i = \frac{x_i - \bar{x}}{s_x}$$

En este caso, la fórmula para el cálculo de los coeficientes b_j se simplifica y podemos comparar la influencia de cada variable comparando sus respectivos coeficientes:

$$b_{norm} = R^{-1} \begin{bmatrix} cor(x_1, y) \\ \vdots \\ cor(x_p, y) \end{bmatrix},$$

donde b_{norm} es el vector de los b_i normalizados y R es la matriz de correlaciones de las variables explicativas. En el caso de regresión simple, el coeficiente coincide con la correlación entre variables.

La regresión lineal no sólo sirve para poder predecir un valor numérico a partir del resto, sino que los coeficientes asignados a cada variable aportan conocimiento sobre cómo se realiza dicha predicción.

7.2.3 Naturaleza del término de error y suposiciones sobre el mismo

Los coeficientes de regresión indican la importancia de cada variable explicativa en la formación de la respuesta. Estos valores se han obtenido con unos datos particulares, así que nada garantiza su estabilidad al trabajar con otras muestras. Para asegurar esta estabilidad contrastaremos la significación de los coeficientes de regresión. Necesitamos pues conocer la distribución de referencia de estos coeficientes, que depende de la distribución del término de error. Esta distribución es desconocida pero es posible hacer suposiciones razonables sobre ella.

El término de error ε se supone que depende de innumerables factores, cada uno de ellos con una influencia sobre la variable de respuesta muy pequeña, tales que están nada o poco relacionados entre sí. Representa pues la parte impredecible de la variable de respuesta. En estas condiciones se aplica el Teorema del Límite Central, por el que el término de error sigue aproximadamente una distribución normal centrada:

$$\text{Hipótesis 1: } \varepsilon \sim \text{Normal} \quad E[\varepsilon] = 0$$

De existir un factor preponderante sobre los demás debe incluirse como una variable explicativa más en el modelo. Además suponemos que los términos de error están generados por una distribución de probabilidad con la misma varianza para todos los individuos:

$$\text{Hipótesis 2: } \text{var}(\varepsilon_i) = \sigma^2.$$

Por otro lado el valor del error en un individuo es independiente del valor del error en otro individuo (observaciones independientes) que, bajo normalidad, equivale a incorrelación:

Hipótesis 3: $\text{cor}(\varepsilon_i, \varepsilon_j) = 0$.

Las anteriores hipótesis se reducen a $\varepsilon \sim N(0, \sigma^2 I)$.

Por último, el valor del error es independiente de los valores de las variables explicativas:

Hipótesis 4: $\text{cor}(\varepsilon_i, x_j) = 0$.

En caso de tener varianzas distintas para cada individuo y/o correlaciones no nulas entre distintas observaciones, también se podría efectuar su ajuste a condición de tener una estimación de estas varianzas y correlaciones. Puede verse que los coeficientes de regresión se obtienen entonces por mínimos cuadrados ponderados [Peña 2002a], donde la matriz de ponderación es la inversa de la matriz de covarianzas del término de error $\text{var}(\varepsilon) = V$.

$$\hat{b} = X(X'V^{-1}X)^{-1}X'V^{-1}y$$

Siendo su matriz de covarianzas $\text{var}(\hat{b}) = (X'V^{-1}X)^{-1}$.

7.2.4 Estimación de la varianza de la perturbación aleatoria

Una vez establecida la forma de la distribución del término de error, nos queda por estimar su varianza, supuesta constante. Esta varianza es uno de los resultados fundamentales de la regresión, ya que representa el ruido aleatorio del fenómeno que estamos estudiando. Para estimar la varianza σ^2 de la perturbación aleatoria, parece lógico utilizar los residuos, sólo que a diferencia de lo que acontece al estimar una varianza de una distribución, donde las desviaciones respecto de la media están calculadas sobre la misma distribución, ahora cada residuo está calculado en la distribución condicional $f(y|x_1, \dots, x_p)$ correspondiente a cada individuo, luego están calculados en distribuciones distintas pero con la misma varianza (por hipótesis). Podemos pues sumar las desviaciones (residuos) producidos en cada distribución para estimar σ^2 , dividiendo la suma del cuadrado de los residuos por sus respectivos grados de libertad para obtener un estimador no sesgado de σ^2 . Los grados de libertad son la dimensión del espacio donde está definido el vector de residuos $n-p-1$, de la misma forma que $n-1$ es el divisor de la estimación no sesgada de una varianza muestral, por ser $n-1$ la dimensión del vector centrado. La varianza de e es pues su longitud al cuadrado unitaria, esto es, relativizada respecto de la dimensión del espacio:

$$s^2 = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - p - 1}$$

Calculada de esta forma puede verse que la varianza residual es un estimador insesgado de la varianza común del término de error $E[s^2] = \sigma^2$. Por otro lado, se verifica que la varianza residual (o la suma del cuadrado de los residuos) sigue una distribución escalada de Jiquadrado: $(n - p - 1)s^2 \sim \sigma^2 \chi^2_{n-p-1}$, [Peña 2002a].

7.2.5 Significación de los regresores

Ahora ya estamos en condiciones de validar la influencia de los regresores en la formación de la variable de respuesta. Diremos que una variable explicativa es significativa, cuando la variable de respuesta depende efectivamente de esta variable, esto es, su coeficiente β_j no es

nulo. Por tanto, la estimación de un coeficiente b_j próximo a cero nos puede hacer pensar en una variable no significativa para explicar la variable de respuesta.

Sin embargo el tamaño del coeficiente no es suficiente para decidir sobre la significación de la variable. Debemos disponer de una prueba de hipótesis para decidir cuándo un coeficiente b_j es grande o pequeño, para lo que necesitamos disponer de la distribución de referencia de los coeficientes b_j cuando no existe relación. Para ello utilizamos las suposiciones hechas sobre la naturaleza del error ε . Entonces $b \sim N(\beta, \sigma^2(X'X)^{-1})$.

Estos resultados permiten probar la hipótesis nula de no relación entre x_j y y :

$$\begin{cases} H_0: \beta_j = 0. \\ H_1: \beta_j \neq 0, \end{cases}$$

siendo el estadístico de la prueba

$$\frac{b_j}{\sqrt{\text{var}(b_j)}} \sim t_{n-p-1},$$

donde $\text{var}(b_j)$ es el término diagonal correspondiente de $s^2(X'X)^{-1}$, véase la página 172.

La prueba anterior, de acuerdo con lo dicho en el Apartado 7.2.2 sobre interpretación de los coeficientes, siempre debe interpretarse como la significación de la variable x_j cuando en el modelo se mantienen el resto de variables explicativas.

Este estadístico se obtiene de forma rutinaria en los paquetes de regresión. Veamos los resultados obtenidos en el ejemplo de aplicación sobre el valor de las viviendas.

Predictor	Coef	SE Coef	T	P
Constant	3.4330	0.1497	22.94	0.000
lstat	-0.41515	0.02196	-18.91	0.000
crim	-0.011826	0.001175	-10.07	0.000
room	0.10001	0.01768	5.66	0.000

S = 0.2081

La columna Coef da los valores de los coeficientes de regresión ajustados, la columna SE Coef da las desviaciones tipo de los coeficientes, esto es la raíz cuadrada de $\text{var}(b_j)$, y la columna T es el estadístico de Student, cociente entre las dos columnas anteriores. Por último se añade el *p*-valor P correspondiente a este valor de la *t* de Student. Podemos observar que en este caso todos los coeficientes son significativos (todos los *p*-valores son inferiores al nivel de significación usual del 0,05), luego todas las variables son buenas predictoras del valor de la vivienda. El estadístico T (en valor absoluto) da idea de la fuerza de la relación de cada variable explicativa respecto de la variable de respuesta.

Prueba sobre un conjunto de variables explicativas

Podemos efectuar una prueba de nulidad simultánea de un conjunto de regresores, que sin pérdida de generalidad podemos suponer los últimos.

$$\begin{cases} H_0: \beta_{q+1} = \dots = \beta_p = 0. \\ H_1: \text{alguno de los } p-q \text{ últimos } \beta_j \text{ no es nulo.} \end{cases}$$

Para ello procedemos a ajustar dos modelos, uno resultante de la hipótesis nula (por tanto con q regresores) que llamaremos modelo reducido, y otro resultante de la hipótesis

alternativa (con todos los regresores disponibles) que llamaremos modelo completo. El ajuste de cada modelo produce su respectiva suma de cuadrados residual SCR_0 y SCR_1 . La diferencia entre ambas cantidades indica la pertinencia de incluir las $p-q$ últimas variables en el modelo. Obviamente esta diferencia debemos relativizarla respecto de sus grados de libertad que son igual a $p-q$, el número de variables cuyo efecto deseamos probar. Por último, para ver si este efecto es significativo debemos compararlo con la estimación disponible del ruido aleatorio s^2 .

$$F = \frac{(SCR_0 - SCR_1)/p - q}{SCR_1/n - p - 1} \sim F_{p-q, n-p-1}$$

Bajo la hipótesis nula el numerador es del mismo orden que el denominador y ambos siguen distribuciones de χ^2 independientes, luego su cociente sigue una F de Fisher con $p-q$ y $n-p-1$ grados de libertad.

Este estadístico sirve también para probar la significación de una variable, en cuyo caso puede verse su coincidencia con el cuadrado de la t de *Student* correspondiente. Análogamente puede utilizarse para probar el efecto de todas las variables del modelo.

7.2.6 Medidas de bondad del modelo de regresión

Existen numerosos estadísticos para medir la calidad de ajuste de un modelo, aquí solo daremos los más utilizados. Consultar [Montgomery & Peck 1992] para una estudio más exhaustivo del tema.

Una medida de la bondad del modelo de regresión viene dada precisamente por la varianza residual s^2 . No cabe duda de que valores pequeños de s^2 nos indican modelos muy precisos alrededor del valor poblacional $E[y|x_1, \dots, x_p]$. Sin embargo, la interpretación de este valor no es fácil.

Otra forma de medir la calidad del modelo de regresión es a partir de la descomposición de la suma de cuadrados de la variable de respuesta, que vemos a continuación:

Descomposición de la suma de cuadrados

Todo ajuste de un modelo de regresión implica la descomposición de la suma de cuadrados de la variable de respuesta:

$$\sum_{i=1}^n (y_i - \bar{y})^2$$

en dos componentes ortogonales, una debida al ajuste del modelo y otra residual, siendo esta última de longitud mínima y el ajuste de longitud máxima.

Esta descomposición es inmediata al ajustar el modelo después de haber centrado las variables, esto es, después de haber ajustado el modelo mínimo.

Aplicando el teorema de Pitágoras al triángulo formado por el vector de respuesta centrado, el vector de ajuste y el de residuos (Figura 7.7), se obtiene la descomposición siguiente:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n e_i^2$$

Esta descomposición se interpreta como la descomposición de la variabilidad total de la variable de respuesta en variabilidad explicada por el modelo y variabilidad residual.

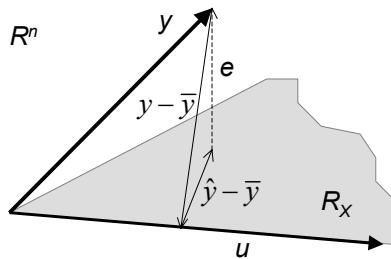


Figura 7.7. Descomposición de la suma de cuadrados.

La descomposición de la suma de cuadrados se acostumbra a presentar en forma de tabla, con el encabezamiento de Análisis de la Varianza (*Analysis of Variance*) o ANOVA. Veamos el resultado obtenido en el ejemplo sobre el valor de la vivienda.

Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	3	62.631	20.877	481.94	0.000
Residual Error	502	21.746	0.043		
Total	505	84.376			

Las sumas de cuadrados aparecen en la columna SS (*Sum of Squares*). En este caso, vemos que la suma de cuadrados total de la variable de respuesta es de 84,376. Al ajustar las tres variables predictoras conseguimos explicar 62,631 y nos quedan sin explicar 21,746. De forma geométrica, las sumas de cuadrados son las longitudes al cuadrado de los respectivos vectores de ajuste, residual y de respuesta centrados. La primera columna DF (*Degrees of Freedom*) se refiere a los grados de libertad, esto es a la dimensión de los respectivos vectores de ajuste, residual y de respuesta centrado. El número de observaciones del archivo *housing.data* es de $n=506$ barrios, luego la dimensión del vector de respuesta centrado será 505, el vector de ajuste está contenido en el espacio definido por los tres regresores, luego su dimensión es 3 y por último el vector residual está contenido en el espacio ortogonal a R_X , luego su dimensión será la diferencia de las dos anteriores, esto es 502. La columna MS (*Mean Sum of squares*) se obtiene dividiendo las dos anteriores, esto es la longitud de los vectores por la dimensión donde están definidos, 20,877 es el numerador de la fórmula sobre la significación simultánea de todos los regresores del Apartado 7.2.5, mientras que 0,043 es s^2 , el denominador de la mencionada fórmula. F es el cociente de ambas cantidades y P su *p-valor*. Vemos que el modelo ajustado es claramente significativo al ser el *p-valor* menor que 0,05.

Coeficiente de determinación

La proximidad entre el vector de respuesta y su ajuste puede medirse mediante el coseno al cuadrado entre ambos vectores, esto es, entre $y - \bar{y}$, $y - \hat{y} - \bar{y}$. Obviamente esta cantidad está relacionada con la suma de los cuadrados residual (SCR) que se minimiza:

$$R^2 = \cos^2(y - \bar{y}, \hat{y} - \bar{y}) = \frac{\|\hat{y} - \bar{y}\|^2}{\|y - \bar{y}\|^2} = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^n e_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}.$$

Multiplicado por 100 este coeficiente puede interpretarse como el porcentaje de variabilidad explicada por el modelo respecto la variabilidad total de la respuesta. Este coeficiente varía entre cero y uno. Un valor igual a cero indica que el modelo propuesto es igual a la media ($\hat{y}_i = \bar{y}$), es el modelo mínimo, mientras que un valor igual a uno indica un ajuste perfecto ($\hat{y}_i = y_i$) que corresponde al modelo saturado. Este coeficiente se llama *coeficiente de determinación* y es igual al cuadrado del coeficiente de correlación entre y e \hat{y} , por este motivo se le denota como R^2 . Es el primer resultado que se mira en una regresión. Da una idea optimista de la calidad del modelo, dado que el efecto de introducir nuevos regresores en el modelo sólo puede hacer disminuir la suma de cuadrados residual y consiguientemente aumentar su R^2 , independientemente de la pertinencia de introducir estos regresores. De hecho, en el límite, si introducimos $n-1$ regresores linealmente independientes obtendríamos un R^2 de 1, un ajuste perfecto.

Coeficiente de determinación ajustado

Otro criterio para medir la calidad global de la regresión es comparar la ganancia en varianza, respecto de la varianza original de la variable de respuesta s_y^2 (normalmente la varianza residual s^2 es menor que la varianza de y):

$$R_{adj}^2 = \frac{s_y^2 - s^2}{s_y^2} = 1 - \frac{\sum_{i=1}^n e_i^2 / n - p - 1}{\sum_{i=1}^n (y_i - \bar{y})^2 / n - 1} = 1 - \frac{n - 1}{n - p - 1} (1 - R^2)$$

A este coeficiente se le llama *coeficiente de determinación ajustado* porque tiene en cuenta la complejidad del modelo, al tener en cuenta el número de variables que lo componen. En este sentido es más adecuado que el R^2 para medir la bondad del modelo. Puede alcanzar valores negativos.

Error cuadrático medio de predicción

Sin embargo debe tenerse en cuenta que en minería de datos el objetivo final de los modelos de regresión es realizar buenas predicciones de individuos anónimos, esto es, con valor de la variable de respuesta desconocido. Por tanto, interesan modelos con errores de predicción lo más pequeños posible.

Consideremos un individuo anónimo con valor (desconocido) de la variable de respuesta:

$$y_0 = r(x_0) + \varepsilon_0$$

Sea $\hat{y}_0 = x_0' b$ el valor predicho por el modelo lineal. El error cuadrático medio de predicción puede descomponerse en el error en la predicción de la función de regresión más el error debido a la predicción de un valor individual y_0 :

$$ECMP(\hat{y}_0) = E[y_0 - \hat{y}_0]^2 = E[\hat{y}_0 - r(x_0)]^2 + \sigma^2$$

El primer término es el error cuadrático medio entre la estimación efectuada por el ajuste y el verdadero valor de la función de regresión, mientras que el segundo es la varianza de y_0 . A su vez el error cuadrático medio de la estimación puede descomponerse en dos términos

$$E[\hat{y}_0 - r(x_0)]^2 = \text{var}(\hat{y}_0) + (E[\hat{y}_0] - r(x_0))^2$$

El primer término es la varianza de la predicción mientras el segundo corresponde al sesgo de la predicción. Si las hipótesis del modelo son ciertas el sesgo es nulo y la varianza de la

predicción es óptima (Teorema de Gauss-Markov). En este caso el error cuadrático medio de la estimación es igual a la varianza y vale:

$$\text{var}(\hat{y}_0) = E[(x'_0 b - x'_0 \beta)(x'_0 b - x'_0 \beta)'] = x'_0 E[(b - \beta)(b - \beta)'] x_0 = \sigma^2 x'_0 (X'X)^{-1} x_0.$$

Sin embargo podemos fácilmente tener estimadores sesgados: basta que no se cumpla alguna de las hipótesis del modelo o simplemente no haber incluido todas las variables relevantes en él. Por ello puede parecer prudente incluir el mayor número posible de variables explicativas en el modelo, a fin de minimizar el riesgo de sesgo. Sin embargo, añadir variables no relevantes significa añadir ruido al modelo, esto es, aumentar la varianza de las predicciones. Se concluye que una buena estrategia de modelización es conseguir el equilibrio entre sesgo y varianza. Por lo tanto la medida relevante de la calidad de un modelo es el error cuadrático medio de predicción $ECMP(\hat{y})$.

Una forma empírica y honesta de estimar el error de predicción es disponer de suficientes datos, situación habitual en minería de datos, y dividirlos en dos grupos, como se comentó en los capítulos iniciales, uno que conformará la muestra de aprendizaje (conjunto de entrenamiento), con la que estimaremos el modelo de regresión y otra muestra, independiente de la anterior, que formará la muestra de prueba (test). Esta muestra servirá pues para estimar el error de predicción del modelo:

$$ECMP_{test}(\hat{y}) = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_i^{test} - \hat{y}_i^{test})^2,$$

donde n_{test} es el número de individuos de la muestra test y y_i^{test} sus valores. Lo podemos expresar en términos del R^2 calculado sobre la muestra test de validación, lo que nos dará una idea más realista del porcentaje de explicación del modelo:

$$R_{test}^2 = 1 - \frac{ECMP_{test}(\hat{y})}{\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_i^{test} - \bar{y}^{test})^2}$$

En caso de no disponer de suficientes datos, siempre podemos utilizar la técnica de *validación cruzada* (esta técnica se ve en detalle en el Capítulo 17). En el caso de la validación cruzada total, consiste en utilizar $n-1$ individuos para estimar el modelo y evaluar el error producido en el individuo dejado aparte. Repitiendo este proceso para cada uno de los individuos de la muestra, obtenemos una estimación honesta del error de predicción. Es el llamado estadístico *PRESS* (*PRediction Error Sum of Squares*). Una ventaja de este estadístico es que no es necesario, en la práctica, calcular los n ajustes con $n-1$ datos para obtenerlo, sino que se obtiene algebraicamente. En concreto, se puede demostrar que el error cuadrático medio de predicción por validación cruzada es:

$$ECMP_{CV}(\hat{y}) = \frac{1}{n} \text{PRESS} = \frac{1}{n} \sum_{i=1}^n \frac{e_i}{(1-h_{ii})'}$$

donde los h_{ii} son los términos de la diagonal de la matriz sombrero H [Montgomery & Peck 1992].

Los valores de estos estadísticos en el ejemplo de aplicación *housing* son los siguientes:

$S = 0.2081$	$R\text{-Sq} = 74.2\%$	$R\text{-Sq(adj)} = 74.1\%$
$\text{PRESS} = 22.4060$	$R\text{-Sq(pred)} = 73.45\%$	

La desviación tipo del término de error es 0,081, el R^2 ($R\text{-sq}$) es de 74,2 por ciento, esto es, el modelo ajustado casi explica las $\frac{3}{4}$ partes de la variabilidad del valor de las viviendas. Porcentaje muy parecido obtenemos con el R^2 ajustado ($R\text{-sq(adj)}$), debido a que el número de observaciones es relativamente alto ($n=505$). Mientras que el R^2 de predicción ($R\text{-Sq(pred)}$), medido por validación cruzada, mediante el estadístico PRESS es ligeramente inferior, pero representa una estimación honesta de la calidad predictiva del modelo.

7.2.7 Selección de las variables del modelo

En el contexto de minería de datos tendremos habitualmente un número muy grande de regresores potenciales. Veremos que incluir todas las variables en el modelo no es la mejor solución, sino que debemos proceder a una selección de los *mejores* regresores. Lógicamente, la selección de variables en un modelo debe estar guiada por su significación, por su aporte a la calidad global del modelo (por ejemplo por la reducción del error de predicción que implica) y por una base teórica sustantiva sobre la pertinencia de incluir un determinado regresor.

Si las variables explicativas son independientes (con matriz de correlaciones cercana a la matriz diagonal unidad), la selección de las variables no es una cuestión difícil, puesto que la explicación que aportan sobre la variable de respuesta también es independiente, por lo que podemos hablar de la significación de cada variable independientemente de las demás. Pero si las variables explicativas están correlacionadas, entonces la significación de una variable depende de qué otras variables han sido seleccionadas previamente en el modelo. Lo normal en problemas de minería de datos es que las variables explicativas estén correlacionadas entre sí y a veces altamente correlacionadas. Es el caso, por ejemplo, de la altura y la amplitud de brazos para predecir el peso de la persona. En este caso, incluir las dos variables es una mala solución. Tener variables explicativas muy correlacionadas significa tener mal definido el espacio de proyección R_X y por tanto pequeñas variaciones en los datos provocan grandes cambios en los coeficientes del modelo. Cuando existen correlaciones elevadas entre las variables explicativas entonces decimos que tenemos un problema de *colinealidad* (significa que las variables explicativas están sobre la misma dirección o casi y por tanto la información aportada acerca de la y es redundante).

La matriz de correlaciones entre las variables explicativas es la primera opción para detectar variables colineales, como se vio en la Sección 5.4.2. Por ejemplo, dos variables con correlación alta ($>0,8$ por ejemplo), indica que son dos variables bastante redundantes, pero no permite detectar situaciones de tener una variable explicativa correlacionada con combinaciones lineales de otras variables explicativas. Una forma general de detectar el problema de colinealidad es mediante los coeficientes VIF (*Variance Inflation Factor*). La varianza de cada coeficiente b_j puede escribirse [Peña 2002a]:

$$\text{var}(b_j) = \frac{s^2}{(n-1)s_{x_j}^2} \frac{1}{1-R_j^2},$$

donde $s_{x_j}^2$ es la varianza del regresor x_j y R_j^2 es el coeficiente de determinación de la variable x_j respecto del resto de variables explicativas (es decir, de la regresión lineal de x_j respecto el resto de variables explicativas). Un coeficiente de determinación cercano a uno indica una variable x_j casi colineal con el resto de variables, luego redundante. Entonces el factor

$1/(1-R_j^2)$ tiende a infinito, haciendo crecer a su vez la varianza del coeficiente. A este factor se le llama VIF (*Variance Inflation Factor*) y se calcula fácilmente como el elemento j -ésimo de la diagonal de la inversa de la matriz de correlaciones de las variables explicativas [Peña 2002a]. El coeficiente VIF, o mejor aún su raíz cuadrada, indica directamente el incremento de amplitud de los intervalos de confianza de los coeficientes del modelo. Valores de VIF superiores a cuatro indican situaciones de colinealidad preocupante.

Una forma de soslayar el problema de la colinealidad es seleccionar las variables que se incluyen en el modelo mediante el algoritmo de Furnival-Wilson para detectar los subconjuntos de regresores óptimos desde el punto de vista de su calidad predictiva [Furnival & Wilson 1974].

En caso de no ser posible la aplicación de este algoritmo (por tener un número demasiado elevado de regresores, por ejemplo) siempre es posible realizar una exploración secuencial del conjunto de regresores, como se comentó en la Sección 5.4.2 de selección de características, mediante la técnica de eliminar el regresor menos significativo a cada paso partiendo del modelo con todos los regresores disponibles (estrategia *backward*) o si ello todavía fuese demasiado costoso, mediante la técnica de añadir el regresor más significativo en cada paso partiendo del modelo más simple, esto es, con sólo el vector unidad (estrategia *forward*).

En el caso de los datos *housing*, la selección de la variable `crim` se ha efectuado como la segunda variable más significativa, mientras que la variable `room` se ha incluido para tener una variable indicadora del valor de la construcción de la vivienda. La relación ordenada de todos los ajustes obtenida con el algoritmo de Furnival-Wilson es la siguiente:

Best Subsets Regression: lvalue versus crim; zn; ...														
Vars	Response is lvalue		p			t			i			r		
	R-Sq	R-Sq(adj)	C-p	S										
1	67.7	67.7	299.7	0.23244										X
2	72.6	72.5	181.0	0.21445	X									X
3	74.6	74.5	132.2	0.20650	X									X X
4	76.0	75.8	100.8	0.20115	X					X				X X
5	77.2	76.9	73.3	0.19631	X			X		X				X X
6	78.0	77.8	54.0	0.19278	X		X	X		X				X X
7	78.6	78.3	41.6	0.19041	X		X			X X	X X			X
8	79.1	78.8	30.0	0.18816	X		X X			X X	X X			X
9	79.8	79.4	15.8	0.18536	X		X X			X X	X X	X X		X
10	80.1	79.7	10.5	0.18421	X		X X X			X X	X X	X X		X X X
11	80.2	79.7	10.5	0.18402	X		X X X	X X		X X	X X	X X		X X X
12	80.2	79.7	12.1	0.18412	X		X X X	X X	X X	X X	X X	X X		X X X
13	80.2	79.7	14.0	0.18430	X X X X X		X X X X X	X X X X X	X X X X X	X X X X X	X X X X X			X X X X X

Las X indican las variables incluidas en el modelo según el número de regresores (especificado en la columna `Vars`). Las columnas `R-Sq`, `R-Sq(adj)` y `s`, se refieren a los

estadísticos ya explicados, mientras que la columna C-p se refiere al estadístico C_p de Mallows (consultar [Peña 2002a] o la página 272 de [Montgomery & Peck 1992]). Este estadístico estima el cociente del error cuadrático medio de estimación dividido por la medida de la fluctuación aleatoria σ^2 ; es por tanto una combinación del sesgo de la estimación y de su varianza. La Figura 7.8 ilustra el decrecimiento de este estadístico al aumentar el número de variables para al final volver a crecer. Una buena política de selección de las variables explicativas es seleccionar el modelo que haga mínimo este estadístico C_p , lo que en el anterior problema nos llevaría a seleccionar el modelo con diez variables (las marcadas con una X), lo que da un ajuste con un R^2 del 80,1 por ciento.

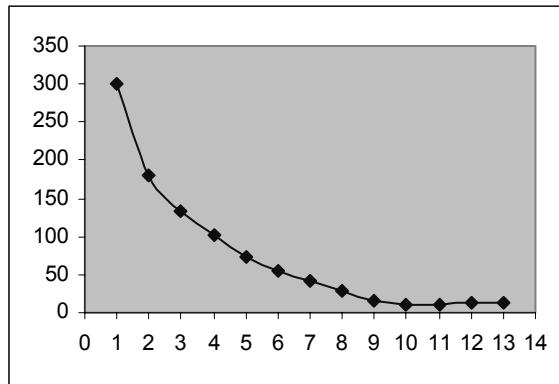


Figura 7.8. Evolución del estadístico C_p de Mallows en función del número de regresores.

También se puede utilizar para seleccionar el modelo, el gráfico de $R^2(\text{adj})$ según el número de variables incluidas del modelo. Cuando el crecimiento del $R^2(\text{adj})$ se estabiliza se tiene una indicación de que no es necesario incluir más variables en el modelo.

7.3 Modelos de regresión sobre componentes incorrelacionados

Una forma de solucionar el problema de la colinealidad sin necesidad de tener que seleccionar las variables es efectuar la regresión sobre una transformación no correlacionada de las variables originales. Una forma de realizar esto es mediante la transformación Gram-Schmidt de las variables originales [Hastie et al. 2001]. Sin embargo, la forma más usual de obviar la colinealidad es gracias a los componentes principales de X .

7.3.1 Regresión sobre componentes principales

Como vimos en el Capítulo 4 (Sección 4.3.1), las componentes principales f_k son combinaciones lineales de las variables originales X , tales que cada componente retenga el máximo de información (varianza) de las variables originales, sin que estos nuevos componentes estén correlacionados entre sí:

$$f_k = Xu_k$$

donde u_k es el k -ésimo vector propio de la matriz de covarianzas $\frac{1}{n-1}X'X = U'\Lambda U$, siendo U la matriz de vectores propios por columnas y Λ es la matriz diagonal de valores propios. Expresamos en F los componentes principales por columna: $F = XU$. El hecho de que X no

sea de rango completo (esto es, hay alguna variable explicativa colineal con las demás) no representa ningún problema, basta retener en la matriz U los vectores propios asociados a los valores propios positivos.

Es fácil ver que los componentes principales están no correlacionados entre sí y que sus varianzas son iguales a los valores propios encontrados:

$$\frac{1}{n-1} F'F = \Lambda$$

Los componentes principales definen el mismo espacio de proyección R_X , por tanto el vector de ajuste \hat{y} es el mismo. La regresión sin embargo queda muy simplificada.

El ajuste en función de los componentes principales se escribe $y = Fc + e$, donde c es ahora el vector de coeficientes de regresión y e es el vector de residuos. Luego,

$$c = (F'F)^{-1} F'y = \frac{1}{n-1} \Lambda^{-1} F'y$$

La matriz de covarianzas de estos coeficientes es:

$$\text{var}(c) = \frac{\sigma^2}{n-1} \Lambda^{-1}$$

y, por tanto los coeficientes c_j son independientes. Por último sólo resta expresar las predicciones en función de las variables originales para su mejor interpretabilidad:

$$\hat{y} = Fc = XUc = Xb$$

La utilización de los componentes principales permite a su vez afrontar otros problemas de la regresión. Por un lado el análisis de observaciones influyentes (al que nos referiremos en la Sección 7.5), también el problema de tener más variables explicativas que individuos ($p > n$), presencia de valores faltantes en la matriz X y, en particular, el problema de errores de medición en las variables explicativas. El hecho de que en regresión las variables x_j no se supongan aleatorias, no significa que su medición no incorpore una parte de fluctuación aleatoria que obviamente formará parte de la definición del espacio de proyección y por ende del ajuste. De este modo, puede ser interesante definir un espacio de proyección más robusto, que elimine parte del ruido de las variables x_j .

En Análisis de Componentes Principales se postula que estas fluctuaciones aleatorias quedan recogidas en los últimos componentes principales, los de menor varianza y que consiguientemente darán coeficientes c_j con más variabilidad. Parece pues lógico eliminar estos componentes de la regresión. Queda el problema de decidir qué componentes expresan ruido aleatorio. Para ello podemos utilizar una técnica directamente relacionada con el objetivo del problema planteado, esto es, seleccionar los componentes que se retendrán a partir del gráfico del incremento de R^2 en la muestra de validación, según modelos con un número creciente de componentes. El modelo con el máximo R^2 , o donde este coeficiente se estabilice, indicará el número de componentes que deben retenerse²¹.

²¹ Otra técnica para reducir la colinealidad es la Regresión Contraída (*ridge regression*), la cual se reduce a efectuar una regresión sobre los componentes principales ponderados de forma inversa a su varianza, en vez de truncar el número de componentes a retener [Hastie et al. 2001].

7.3.2 Regresión sobre componentes PLS

Una limitación de utilizar los componentes principales aparece subyacente en el párrafo anterior. Dado que los componentes principales están pensados para explicar la variabilidad de X sin tener en cuenta su relación con la variable de respuesta y , puede suceder que los primeros componentes, los más importantes, sean unos malos predictores de y y que predigan mejor los componentes posteriores. Tiene interés pues encontrar unos componentes función de las variables explicativas, no correlacionadas entre sí y que a su vez sean explicativas de la variable de respuesta. Una forma de resolver esto es mediante regresión sobre las componentes PLS (*Partial Least Squares*) [Wold 1985].

La regresión PLS consiste en buscar combinaciones lineales de las variables explicativas $t_k = Xv_k$ incorrelacionadas entre sí y tales que su covarianza con la variable de respuesta sea máxima.

Observemos que los componentes PLS están a medio camino de los componentes principales (maximizan la varianza de t_k) y de la regresión lineal (maximizan la correlación entre la y y t_k):

$$\text{cov}(t_k, y) = \text{cor}(t_k, y) \times \text{var}(t_k)$$

Esta maximización lleva a $v_{j1} \approx \text{cov}(x_j, y)$, luego $t_1 = Xv_1$. A continuación, para asegurar la ortogonalidad de los sucesivos componentes PLS, se deflacta el espacio R_X respecto del último componente encontrado (se proyectan las variables x_j al subespacio ortogonal a la dirección definida por t_k). Con la nueva matriz residual X se procede de forma análoga para buscar el segundo componente PLS y así tantas veces hasta llegar al rango de la matriz X . Una vez encontrados los componentes PLS se procede como en componentes principales, realizando su regresión respecto de la variable de respuesta [Tenenhaus 1998].

En este caso, los componentes PLS tienen una interpretación en función de la explicación de la variable de respuesta y y vienen ordenados de forma natural *según su capacidad predictiva* de y .

7.4 Modelos de regresión con variables categóricas

Las variables categóricas, más que ser una excepción, a menudo son la norma en las aplicaciones de minería de datos, luego debemos tener una forma de tratarlas. Afortunadamente, las variables explicativas categóricas se pueden introducir sin problema en un modelo de regresión lineal mediante métodos de numerización como los que vimos en la Sección 4.4.2.

La forma habitual y más general de actuar es convertirlas a variables binarias, esto es, una variable categórica de q modalidades genera q variables binarias (numerización “1 a n ”). Por ejemplo, la variable *CSP* (categoría socio-profesional) de cinco categorías (*clase baja, media-baja, media, media-alta* y *alta*) se puede codificar como cinco variables binarias²², cada una correspondiente a una modalidad:

²² Otra forma de numerizar una variable, cuando ésta es ordinal, es mediante una numerización “1 a 1”, por ejemplo convirtiendo los valores (*baja, media-baja, media, media-alta* y *alta*) a valores numéricos que preserven el orden (0, 1, 2, 3 y 4, por ejemplo). Para asegurar la corrección de esta transformación

$$\begin{array}{c}
 \begin{matrix} CSP & b & \%_m & m & \%_a & a \\ \hline
 \text{media} & 0 & 0 & 1 & 0 & 0 \\
 \text{baja} & 1 & 0 & 0 & 0 & 0 \\
 \dots & \cdot & \cdot & \cdot & \cdot & \cdot
 \end{matrix} \\
 \Rightarrow
 \end{array}$$

Por tanto, cada variable categórica da lugar a la introducción en la matriz X de un bloque de variables binarias correspondiente a sus modalidades.

El modelo se escribe entonces como:

$$y = \beta_0 + \beta_1 x_1 + \dots + \alpha_1 b + \alpha_2 \%_m + \alpha_3 m + \alpha_4 \%_a + \alpha_5 a + \varepsilon,$$

donde los coeficientes α_k representan el incremento o decremento en la variable de respuesta por el hecho de que el individuo posee la modalidad k de la variable categórica.

Puede verse que entonces la matriz X no será de rango completo, pues la suma de cada bloque es siempre igual al vector unidad. Es decir, un valor siempre es dependiente de los demás. Ello implica la necesidad de introducir restricciones sobre los coeficientes de las modalidades binarias de las variables categóricas. La restricción más simple consiste en eliminar una modalidad por variable categórica (lo que se conoce como numerización “1 a $n-1$ ”), entonces el efecto de esta modalidad vendrá confundido con el coeficiente del término independiente, actuando esta modalidad de nivel de base a partir del cual se miden los efectos de las demás modalidades presentes en el modelo. Por ejemplo, eliminando la última modalidad en el ejemplo anterior, el ajuste se escribe

$$\hat{y} = b_0 + b_1 x_1 + \dots + a_1 b + a_2 \%_m + a_3 m + a_4 \%_a,$$

donde b_0 representa el valor de y para los individuos de clase alta, puesto que esa es la modalidad eliminada, mientras a_1 representa la variación en la variable de respuesta al pasar de clase alta a clase baja, etc.

Otras restricciones sobre los coeficientes de las modalidades son posibles, por ejemplo:

$$\sum_{k=1}^q \alpha_k = 0$$

conduciendo todas ellas al mismo vector de ajuste, si bien con una interpretación de los coeficientes distinta. De forma análoga pueden introducirse nuevas variables categóricas en el modelo.

Debe tenerse cuidado al mirar la significación de las modalidades de forma aislada. Un coeficiente no significativo de una modalidad indica simplemente un efecto no diferenciado de esta modalidad respecto el nivel de base y por tanto se podría pensar en agrupar dicha modalidad con el nivel de base de cara a una reducción de modalidades y por tanto de la complejidad del modelo. Sin embargo si deseamos probar el efecto significativo de la variable categórica deberemos realizar la prueba de nulidad simultánea de todos los coeficientes de las modalidades concernidas, tal como se ha visto en la Sección 7.2.5 (página 176).

El hecho de disponer de variables categóricas en el modelo permite probar de forma fácil la existencia de interacciones entre ellas. En efecto, la variable de respuesta puede

deberemos verificar si los coeficientes respectivos de una numerización “1 a $n-1$ ” cumplen el orden inducido por las categorías de la variable (esto es, son crecientes o decrecientes).

comportarse de forma no aditiva. Por ejemplo, si deseamos aproximar la propensión al ahorro en función del nivel de ingresos y de la clase social, es posible que el efecto del nivel de ingresos sea distinto según la clase social del individuo. Para ello basta introducir el producto de la variable “nivel de ingresos” con las modalidades de la *CSP* retenidas en el modelo y probar el efecto de la nulidad simultánea de los coeficientes de interacción ($\alpha_{jx_j} = 0$, para todo j). Así aparecen muchas combinaciones:

$$\hat{y} = b_0 + b_1 x_1 + \dots + a_1 b + a_2 \frac{b}{m} + a_3 m + a_4 \frac{m}{a} + a_{1x_1} (bx_1) + a_{2x_1} \left(\frac{b}{m} x_1\right) + a_{3x_1} (mx_1) + a_{4x_1} \left(\frac{m}{a} x_1\right)$$

Asimismo, mediante el producto de las modalidades retenidas en el modelo se probaría el efecto de la interacción entre dos variables categóricas.

7.5 Análisis de los residuos

Si el modelo es correcto, los residuos deben ser *ruido blanco*, esto es, deben ser fluctuación aleatoria pura y no contener información. La verificación de este supuesto es una etapa importante de la modelización y cubre los siguientes aspectos:

1. Normalidad de los residuos.
2. Observaciones mal ajustadas, alejadas e influyentes.
3. Varianza constante del error.
4. No linealidad.

A continuación, damos los conceptos básicos para verificar estos puntos.

De acuerdo con las suposiciones efectuadas en la Sección 7.2.3 la fluctuación aleatoria sigue una distribución normal, luego es lógico suponer que los residuos e_i siguen esa misma distribución. En realidad no es exactamente así porque los residuos están ligeramente correlacionados entre sí: $\text{var}(e_i) = \text{var}(y_i - \hat{y}_i) = \sigma^2(I - H)$. Una forma aproximada y visual para verificar la normalidad de los residuos es mediante el histograma de los mismos. La forma del histograma revela su adecuación o no a una campana de Gauss. Por supuesto podremos utilizar otras representaciones gráficas, en particular el *normal probability plot*²³ o el cálculo de estadísticos de ajuste para asegurar esta hipótesis de normalidad (consultar el ajuste a una distribución normal en [Peña 2001]). La no normalidad puede indicar la necesidad de efectuar una transformación sobre la variable de respuesta que normalice los residuos, o bien que no se han introducido todas las variables relevantes en el modelo.

El histograma sirve también para detectar la existencia de residuos altos. Si realizamos el histograma de residuos estandarizados (esto es, el residuo dividido por su desviación tipo), las observaciones con un residuo superior a tres indican una observación muy poco probable bajo el supuesto de normalidad, luego se trata de una observación con residuo alto. Un residuo alto indica una observación a la que el modelo no se ajusta bien. Estas observaciones no deben desecharse, sino que deben ser localizadas en la base de datos y

²³ El *normal probability plot* es un diagrama entre los residuos y los valores teóricos de estos residuos bajo el supuesto de que siguen una distribución normal. Si esta hipótesis es cierta el diagrama forma una línea recta. Diagramas con curvaturas indican situaciones de no normalidad en los residuos. También sirve para detectar observaciones con residuos altos.

procurar entender el porqué de su mal ajuste al modelo, ya que pueden contener información valiosa sobre variables no incluidas en el modelo.

Por otro lado, no debemos confundir observaciones con residuos altos con observaciones atípicas (del inglés *outliers*). Las observaciones atípicas respecto de las variables explicativas, definen puntos del espacio R_X alejados de los demás y, precisamente por este hecho, pueden atraer el hiperplano del ajuste hacia ellas, resultando con unos residuos pequeños, y por tanto no ser detectables en el histograma de los residuos. La presencia de observaciones atípicas puede invalidar la matriz de correlaciones entre las variables explicativas y por tanto todas las estimaciones del modelo. Las observaciones alejadas del resto, de las que depende en especial el ajuste realizado, se llaman observaciones influyentes. Debe tenerse en cuenta que no siempre una observación alejada es una observación influyente. Por ejemplo, el gráfico de la Figura 7.9 ilustra la diferencia entre una observación con residuo alto (alejada de la recta ajustada) relativamente influyente, una observación alejada (según la variable explicativa) del resto de observaciones, pero que si la omitimos no cambia de forma sustancial la recta de ajuste, y una observación alejada del resto cuya omisión provoca un gran cambio en la recta de regresión hallada y por tanto es muy influyente.

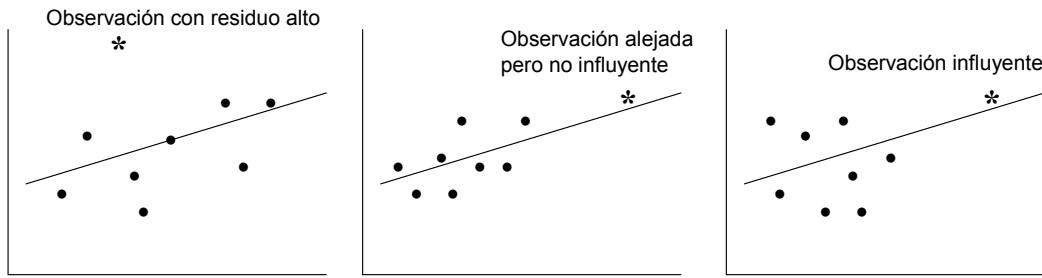


Figura 7.9. Observaciones con residuo alto, alejada pero no influyente y alejada que sí influye en la recta ajustada.

El término h_{ii} diagonal de la matriz sombrero H (véase la página 171), llamado en inglés *leverage*, permite detectar las observaciones alejadas de la nube de puntos, definida esta nube por las variables explicativas. Valores de h_{ii} superiores a $3p/n$ (donde p es el número de variables exploratorias y n el número de individuos) pueden considerarse excesivos debido al hecho que:

$$\sum_{i=1}^n h_{ii} = \text{traza}(H) = p.$$

Un estadístico para medir la influencia de una observación respecto al ajuste es la denominada Distancia de Cook, D_i , combinación entre el residuo de la observación y su "alejamiento":

$$D_i = \frac{e_i^2}{s^2(1-h_{ii})(p+1)} \frac{h_{ii}}{1-h_{ii}}$$

Los programas estadísticos proveen normalmente estos estadísticos que permiten la localización fácil de las observaciones mal ajustadas (con residuos altos) y observaciones influyentes de las que depende el ajuste obtenido. Consultese [Fox 1991] para un estudio más detallado del tema.

Por consiguiente, previamente al modelo de regresión, tendremos cuidado en realizar una buena exploración de los datos recogidos, de cara a detectar observaciones atípicas y

subsanar errores, variable a variable y también de forma multivariante. Podemos utilizar la técnica de identificación de grupos de datos atípicos (*outlier detection*) [Peña & Prieto 2001], o de forma más simple, mediante un Análisis de Componentes Principales de las variables explicativas, para detectar observaciones muy alejadas de la nube de puntos, mediante inspección de las representaciones factoriales de los individuos.

Otra hipótesis por validar es la de que la varianza de la fluctuación aleatoria es constante. Para ello realizaremos el gráfico de los residuos e_i respecto de los valores ajustados. Si la varianza es constante, debemos ver en el gráfico una dispersión también constante según crece el valor ajustado. Por el contrario un gráfico en forma de embudo indicaría una situación de heterocedasticidad, esto es, que la varianza crece (o decrece) al aumentar el valor ajustado.

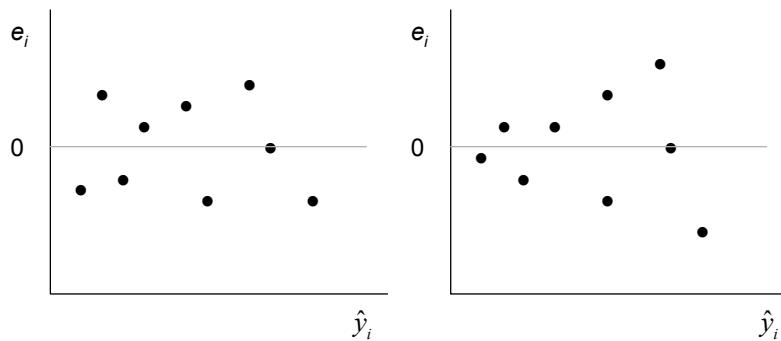


Figura 7.10. Gráfico con varianza residual constante y gráfico con varianza residual creciente.

Para solucionar este problema debe reespecificarse el modelo, en general mediante una transformación de la variable de respuesta, como las vistas en la Sección 7.2.1. Por fortuna, en general las mismas transformaciones que permiten solucionar el problema de la no normalidad de la variable de respuesta también corrigen el problema de la heterocedasticidad. Si es posible tener una estimación de la varianza de la fluctuación aleatoria para cada observación, entonces otra posible solución es efectuar una regresión por mínimos cuadrados ponderados, tal como se explicó en la página 82. En este punto será útil por ejemplo inspeccionar los diagramas bivariantes de los residuos e_i con cada una de las variables explicativas x_{ji} , para detectar eventuales dependencias no tenidas en cuenta en el modelo.

Otra suposición básica del modelo es la linealidad de las relaciones. Esta suposición es a menudo una aproximación a la verdadera relación existente entre la variable de respuesta y las variables explicativas, la cual en general será suficientemente aproximada para obtener resultados útiles. Sin embargo, ajustar un modelo lineal si las relaciones son claramente no lineales, conducirá a predicciones falsas, puesto que el modelo no incorporará toda la explicación sistemática de las variables explicativas. Para verificar esta hipótesis será básico efectuar los diagramas bivariantes de la variable de respuesta y con cada una de las variables explicativas x_j . Sin embargo, estos diagramas no cuentan toda la historia, puesto que ya sabemos que en una ecuación de regresión la relación entre la variable explicativa y la variable de respuesta es *condicional* al resto de variables explicativas presentes en el modelo. Una posibilidad entonces es efectuar gráficos parciales, esto es, entre la variable de respuesta y la variable explicativa, después de haber eliminado

en ambas el efecto del resto de variables explicativas²⁴, sin embargo estos gráficos no siempre son fáciles de interpretar. Una forma de soslayar el problema es introducir regresores que puedan servir para tener en cuenta la no linealidad (o también, efectuar transformaciones no lineales de las variables) y verificar a posteriori la significación de los regresores introducidos, así como la verificación de las hipótesis básicas del modelo, tal como ilustra el siguiente ejemplo.

7.6 Ejemplo: aplicación a los datos SERVO

Como ejemplo completo de aplicación de los modelos de regresión y los conceptos introducidos hasta ahora utilizamos los datos de Karl Ulrich (MIT) sobre el tiempo de respuesta de un servomecanismo en función de dos variables cuantitativas, que denominaremos x_1 y x_2 y dos variables categóricas A y B . La variable de respuesta expresa el tiempo de respuesta de un servomecanismo, con valores estrictamente mayores que cero. La transcripción del problema (véase el Apéndice B) no da más detalles sobre el significado de estas variables pero se señala que el problema es extremadamente no lineal. Una primera inspección de los 167 datos revela claramente este fenómeno.

La distribución de la variable de respuesta (que varía entre 0 y 8) es claramente no normal, tal como se aprecia en la Figura 7.11.

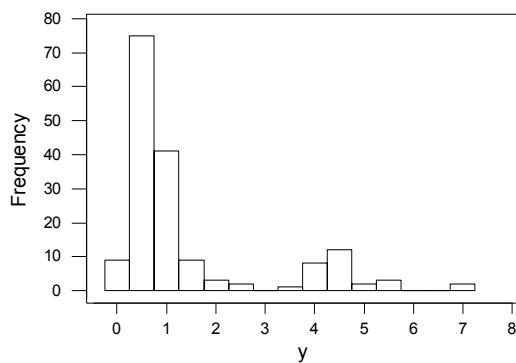


Figura 7.11. Distribución del tiempo de respuesta y .

Los diagramas bivariantes de la Figura 7.12 muestran claramente que las variables x_1 y x_2 no son lineales respecto la variable de respuesta, ni tampoco la varianza σ^2 se mantiene constante²⁵. Por otro lado los diagramas de puntos del tiempo de respuesta respecto de las variables categóricas A y B (Figura 7.13), muestran también diagramas multimodales.

Es incorrecto, por tanto, realizar el ajuste de un modelo lineal del tiempo de respuesta en función de las x_1 , x_2 , A y B . Una transformación adecuada en estos casos es tomar logaritmos de la variable de respuesta (esto equivale a suponer que el término de error es multiplicativo y de distribución lognormal: $y_i = e^{\mu_i} \varepsilon_i$ y por tanto $\ln y_i = \mu_i + \ln \varepsilon_i$, con

²⁴ Esto es, si tenemos tres variables explicativas (x_1 , x_2 y x_3) y queremos ver qué relación existe entre la variable de respuesta y y x_1 , efectuaríamos primero la regresión entre y y x_2 y x_3 , después entre x_1 y x_2 y x_3 y a continuación observaríamos el diagrama bivariante de los residuos de las dos regresiones anteriores. Este diagrama refleja la relación entre ambas variables condicional a x_2 y x_3 .

²⁵ A este fenómeno se le llama heterocedasticidad.

$\mu_i = \beta' x_i$). La transformación logarítmica hace más simétrica la distribución de la variable de respuesta, uniformiza la varianza residual (reduce la heterocedasticidad), lleva la variable de respuesta a toda la recta real y reduce la no linealidad, si bien no la elimina; por tanto, decidimos introducir el cuadrado de las variables, x_1^2 y x_2^2 y un término de interacción $x_1 x_2$ para dar cuenta de eventuales efectos no aditivos entre las variables cuantitativas. Las variables categóricas las introducimos en el modelo mediante numerización “1 a $n-1$ ” (véase la Sección 4.4.2), eliminando la última modalidad de cada una.

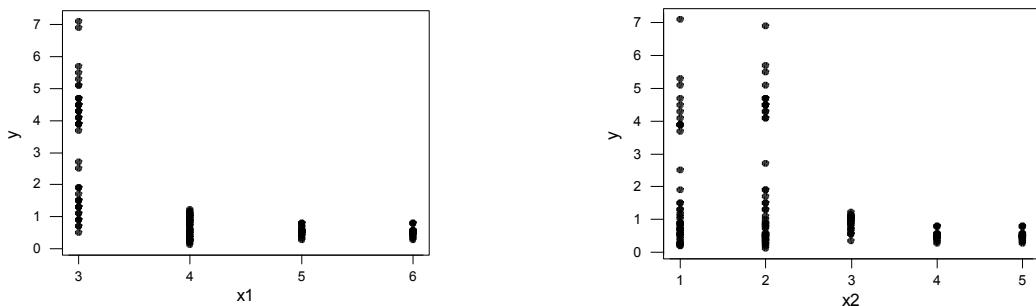


Figura 7.12. Diagramas bivariantes del tiempo de respuesta con x_1 y x_2 .

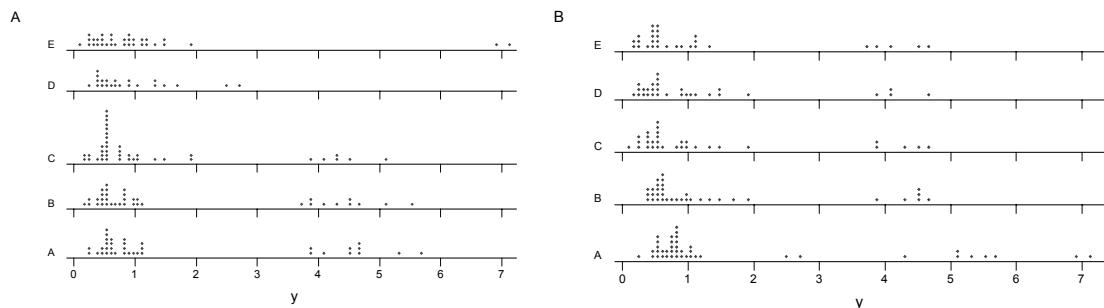


Figura 7.13. Diagramas de puntos del tiempo de respuesta con las variables categóricas A y B.

Los resultados obtenidos en el ajuste son los siguientes (utilizando el software MINITAB):

```
The regression equation is
ln(y) = 10.3 - 4.90 x1 + 0.882 x2 + 0.513 x1^2 + 0.160 x2^2 - 0.295 x1*x2
+ 0.444 A1 + 0.378 A2 + 0.231 A3 - 0.407 A4 + 0.774 B1 + 0.425 B2
+ 0.0842 B3 + 0.0095 B4
```

Predictor	Coef	SE Coef	T	P	VIF
Constant	10.2536	0.5900	17.38	0.000	
x1	-4.9004	0.3263	-15.02	0.000	142.4
x2	0.8823	0.1750	5.04	0.000	74.2
x1^2	0.51264	0.04651	11.02	0.000	230.1
x2^2	0.15999	0.05859	2.73	0.007	290.1
x1*x2	-0.29537	0.08024	-3.68	0.000	662.0
A1	0.44368	0.08653	5.13	0.000	1.6
A2	0.37834	0.08653	4.37	0.000	1.6
A3	0.23093	0.08656	2.67	0.008	1.8
A4	-0.4075	0.1002	-4.07	0.000	1.5

B1	0.77450	0.08986	8.62	0.000	2.0
B2	0.42505	0.09089	4.68	0.000	1.8
B3	0.08420	0.09271	0.91	0.365	1.7
B4	0.00948	0.09342	0.10	0.919	1.7
S = 0.3585		R-Sq = 85.4%		R-Sq(adj) = 84.1%	
PRESS = 23.5511		R-Sq(pred) = 82.46%			

Los valores S, R-Sq, PRESS, etc., se interpretan como vimos en la Sección 7.2.6. Todas las variables introducidas resultan ser significativas, tienen valores de la *t de Student* con valores *p* inferiores a 0,007. Sólo las modalidades tercera y cuarta de la variable categórica *B* podrían fusionarse con la modalidad de base, esto es, la última. Observamos sin embargo unos coeficientes VIF muy altos en los regresores continuos, fruto de las transformaciones efectuadas, lo que significa que existe colinealidad.

Para eliminarla, obtenemos las componentes PLS de los cinco regresores continuos, tal como vimos en la Sección 7.3.2, y efectuamos la regresión sobre las cuatro primeras componentes, habiendo agregando también las tres últimas modalidades de la variable *B*. Los resultados obtenidos entonces son los siguientes:

The regression equation is					
$\ln(y) = -0.602 + 0.0336 \text{ PLS_F1} + 0.139 \text{ PLS_F2} + 0.803 \text{ PLS_F3} + 2.22 \text{ PLS_F4} + 0.429 \text{ A1} + 0.364 \text{ A2} + 0.301 \text{ A3} - 0.385 \text{ A4} + 0.798 \text{ B1} + 0.380 \text{ B2}$					
Predictor	Coef	SE Coef	T	P	VIF
Constant	-0.60186	0.08022	-7.50	0.000	
PLS_F1	0.033595	0.002298	14.62	0.000	1.0
PLS_F2	0.13862	0.01008	13.75	0.000	1.1
PLS_F3	0.8030	0.1054	7.62	0.000	1.0
PLS_F4	2.2224	0.2421	9.18	0.000	1.0
A1	0.4291	0.1026	4.18	0.000	1.6
A2	0.3637	0.1026	3.55	0.001	1.6
A3	0.3007	0.1021	2.94	0.004	1.8
A4	-0.3849	0.1187	-3.24	0.001	1.5
B1	0.79791	0.08375	9.53	0.000	1.2
B2	0.38013	0.08562	4.44	0.000	1.1
S = 0.4253		R-Sq = 79.0%		R-Sq(adj) = 77.6%	
PRESS = 34.1730		R-Sq(pred) = 74.55%			
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	10	106.082	10.608	58.65	0.000
Residual Error	156	28.214	0.181		
Total	166	134.297			

Obsérvese que entonces desaparece el efecto de la colinealidad, con todos los coeficientes significativos. Sin embargo la reducción del porcentaje de explicación es alto, pasando de un *R*² del 85,4 por ciento al 79 por ciento.

Ello es debido a unas observaciones influyentes con valores altos de la variable *x*₁ y bajos de *x*₂ (observaciones 34, 64 y 94), cuando la correlación entre ambas variables es

positiva. Estas observaciones las detectamos fácilmente mediante el gráfico del estadístico h_{ii} , el cual da el “alejamiento” de cada observación respecto la nube de puntos.

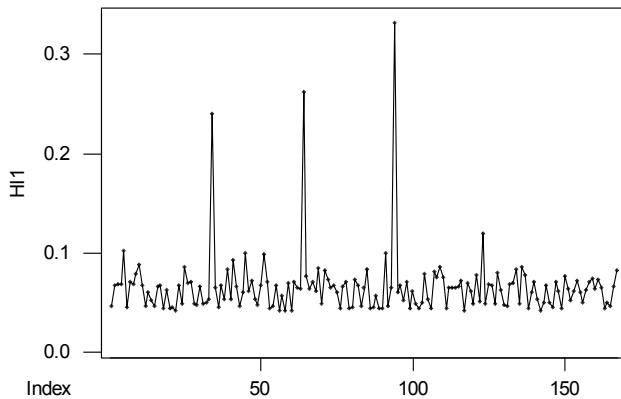


Figura 7.14. Gráfico de h_{ii} , dando las observaciones potencialmente influyentes del ajuste.

Esta asociación negativa entre variables es la que recoge la última componente PLS no incluida en el modelo. Por supuesto que eliminando estas observaciones se mejora sustancialmente la calidad del modelo, pero esta eliminación debe hacerse sobre bases teóricas bien fundadas, o sea, sobre la conveniencia o no de que el modelo incorpore el comportamiento de estas observaciones. Obviamente, de haber incluido todos los cinco componentes PLS hubiésemos obtenido el mismo ajuste que el obtenido con las variables originales. Obsérvese que el R^2 de predicción, del 74,55 por ciento, calculado con el estadístico PRESS, es inferior al valor del R^2 , así como también al R^2 ajustado, tal como es de esperar.

Bajo el epígrafe *Analysis of Variance* se obtiene la descomposición de la suma de cuadrados y la prueba F sobre la nulidad simultánea de todos los coeficientes del modelo (obviamente esta hipótesis se rechaza claramente).

Por último, los residuos, debido a las observaciones antes mencionadas, tienen algunos valores un poco altos, pero su histograma es de forma bastante normal (Figura 7.15).

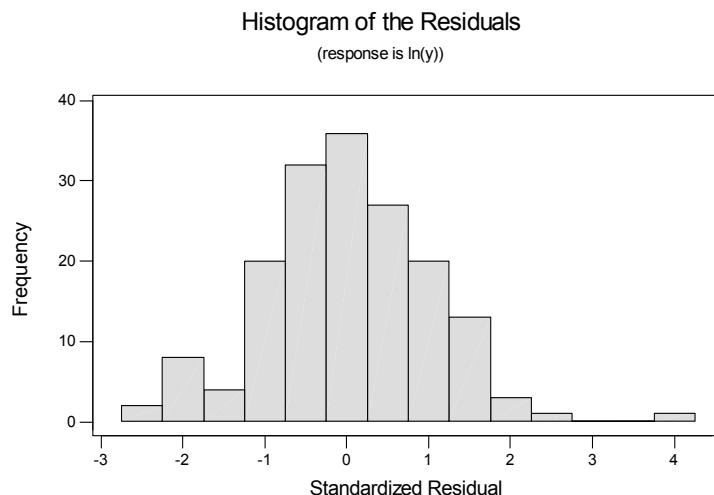


Figura 7.15. Histograma de los residuos del modelo ajustado sobre las componentes PLS.

En este ejemplo hemos visto la necesidad de tener en cuenta la naturaleza de la variable de respuesta antes de proceder a su modelización, de visualizar el tipo de relación con las diferentes variables explicativas y en consecuencia efectuar una transformación sobre la variable de respuesta que permita corregir eventuales disfunciones del modelo. También ha quedado clara la utilidad de incluir potencias crecientes de las variables originales para modelar comportamientos no lineales y la inclusión de términos de interacción para tener en cuenta situaciones de no aditividad. Hemos visto también cómo se incluyen las variables categóricas en el modelo por numerización. Por último hemos visto cómo la transformación de las variables originales en componentes incorrelacionados (en este caso los componentes PLS) permite solucionar el problema de la colinealidad. También se ha visto cómo el hecho de no tener en cuenta los últimos componentes disminuye la capacidad predictiva, aunque en la medida en que los últimos componentes reflejen ruido aleatorio, el modelo obtenido será más robusto. Por último, se ha mostrado la necesidad de efectuar un análisis de residuos para ver la conformidad a los supuestos del modelo de regresión y detectar observaciones y situaciones no tenidas en cuenta en el modelo.

7.7 Modelos lineales generalizados

Hasta ahora sólo hemos considerado modelos donde la variable de respuesta era cuantitativa, seguía una distribución normal y la relación con las variables explicativas podía expresarse mediante una función lineal. Los modelos lineales generalizados expanden estas suposiciones.

En minería de datos podemos encontrarnos con situaciones en que la variable de respuesta es, por ejemplo, la probabilidad de compra de un producto, o el número de ofertas de vacaciones aceptadas o el tiempo que lleva un cliente sin comprar por Internet. En todos estos casos no es correcta la suposición de normalidad de la respuesta. Tampoco la suposición de linealidad parece la más adecuada para modelar situaciones reales del comportamiento de los clientes. Los modelos lineales generalizados permiten modelar estas situaciones.

El modelo estadístico es, sin embargo, el mismo. La respuesta se descompone en dos componentes, uno que da el valor medio de la distribución condicional de la respuesta respecto de las variables explicativas y otro que expresa la fluctuación aleatoria alrededor del valor medio:

$$y_i = r(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) + \varepsilon_i, \\ r(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) = E[y_i | x_{i1}, \dots, x_{ip}] = \mu_i \quad E[\varepsilon_i] = 0,$$

donde la variable de respuesta (y por tanto el error) sigue una distribución de la familia exponencial y el componente sistemático está relacionado con las variables explicativas mediante una función r (generalmente no lineal).

Una distribución es de la familia exponencial si su función de densidad o de probabilidad puede escribirse de la siguiente forma:

$$f(y, \beta) = \exp \{a(y)b(\beta) + c(\beta) + d(y)\}$$

Las distribuciones normal, binomial, Poisson y gamma cumplen este requisito y por tanto pertenecen a la familia exponencial.

La componente sistemática del modelo se expresa ahora mediante una función r (invertible y diferenciable) de la combinación lineal de las variables explicativas. Esta combinación lineal se denomina predictor lineal. A la inversa de la función r , es decir, r^{-1} , se le llama función de unión.

Las funciones de unión que se pueden utilizar son varias, si bien no todas ellas apropiadas para todas las distribuciones del error. Existen, sin embargo, unas funciones de unión naturales para cada distribución del error especificada: se empareja la función de unión identidad con la distribución del error normal (como hemos visto hasta ahora), la función logística (que se explicará más adelante en la Sección 7.7.3) con la distribución binomial, la función logarítmica con la distribución de Poisson y la función inversa ($1/x$) con la distribución gamma. Es obvio pues que el modelo de regresión lineal es un caso particular de los modelos lineales generalizados, con variable de respuesta continua, error normal y función r identidad.

La estimación de los parámetros del modelo supone especificar claramente la función de distribución del error. Podemos entonces estimar los parámetros del modelo (β) como aquellos que hacen más verosímiles los datos recogidos dada la hipótesis (*estimación de máxima verosimilitud*, del inglés *maximum likelihood*). Dada la naturaleza no lineal de la función r^{-1} , la estimación requiere métodos iterativos (de Newton-Raphson o simplificados basados en las propiedades de los estimadores máximo-verosímiles (método *scoring*)) [Nelder & Wedderburn 1972].

7.7.1 Inferencia. Desviación

Los conceptos de inferencia son los mismos que los utilizados en el modelo de regresión lineal, pero adaptados al hecho de que la distribución del error es más general que la distribución normal considerada hasta ahora.

La utilización de la estimación de máxima verosimilitud implica que se tienen distribuciones asintóticas sobre los resultados. En particular, la distribución de los coeficientes del modelo b es normal insesgada y con matriz de covarianzas \mathfrak{I}^{-1} .

$$b \sim N(\beta, \mathfrak{I}^{-1}),$$

donde \mathfrak{I} es la matriz de información de Fisher, es decir:

$$\mathfrak{I} = -E \left[\frac{\partial^2 l}{\partial b_j \partial b_k} \right]$$

evaluada en la estimación máximo-verosímil (donde l significa el logaritmo de la función de verosimilitud) [Dobson 1991].

Este resultado permite contrastar la nulidad para cada coeficiente b_j del vector b . Bajo la hipótesis nula de que $\beta_j=0$, se tiene que

$$\frac{b_j}{\sqrt{i_{jj}}} \sim N(0,1),$$

donde i_{jj} es el j -ésimo elemento de la diagonal de \mathfrak{I}^{-1} . Este resultado permite pues probar la significación de una variable cuantitativa o la igualdad con el nivel de base en caso de corresponder a una modalidad de una variable categórica, tal como se hacía en la Sección 7.2.5 para los coeficientes del modelo de regresión lineal.

La calidad global del ajuste se mide a partir de la diferencia entre las funciones de (log)verosimilitud correspondiente al ajuste perfecto y al modelo propuesto. Esta diferencia se denomina Desviación D y sigue asintóticamente una distribución de χ^2 :

$$D = 2(l_{\text{perfecto}} - l_{\text{propuesto}}) \sim \chi^2_{n-p-1}$$

La Desviación (en inglés denominada *Deviance*, no confundir con la tradicional *deviation*) calculada en un modelo de regresión lineal coincide con la SCR (suma de cuadrados residual) dividida por σ^2 , la cual, recordemos, sigue exactamente una distribución de χ^2 . La Desviación expresa pues una medida de proximidad del ajuste producido a los datos observados.

7.7.2 Pruebas de hipótesis basadas en diferencias de desviaciones

El resultado anterior permite realizar pruebas de hipótesis para la inclusión de conjuntos de variables en el modelo. Para ello consideraremos un modelo con p variables explicativas y deseamos probar la significación de las $p-q$ últimas variables.

Tenemos dos modelos, uno completo con p variables $y = r(x_1, \dots, x_p) + \varepsilon$ y otro reducido a las primeras q variables $y = h(x_1, \dots, x_q) + \varepsilon$. Las Desviaciones obtenidas verifican asintóticamente:

$$\text{Modelo completo: } D_{\text{compl}} \sim \chi^2_{n-p-1}$$

$$\text{Modelo reducido: } D_{\text{reduc}} \sim \chi^2_{n-q-1}$$

Entonces, suponiendo que el modelo verdadero es el reducido, las $p-q$ últimas variables expresan ruido aleatorio, luego

$$\Delta D = D_{\text{reduc}} - D_{\text{compl}} \sim \chi^2_{p-q}$$

Así, diferencias en Desviaciones mayores que el umbral provisto por la distribución de χ^2 indican la pertinencia de incluir las $p-q$ variables en el modelo. Recordemos que para las distribuciones χ^2 se tiene $E[\chi^2_{p-q}] = p-q$. Está claro que para el caso de error normal y variable de respuesta continua utilizaremos la prueba exacta de la Sección 7.2.5.

7.7.3 Regresión logística

Aunque se usa el término *regresión logística*, debido a que es un modelo de regresión generalizada, también se le denomina *discriminación logística*, porque en realidad este método sirve para resolver problemas de clasificación, en concreto, de clasificación suave, tal y como se definió este término en el Capítulo 6. Este tipo de tarea obtiene una estimación de probabilidades para variables de respuesta categóricas y, cuando hablamos de sólo dos clases, hablamos de *ranking*.

Por ello, la regresión logística es uno de los modelos lineales generalizados más frecuentes, ya que modeliza una probabilidad (probabilidad de compra de un producto entre varios, probabilidad de fraude, probabilidad de morosidad, etc.). En este caso la variable de respuesta tiene dos (o más) posibilidades, cada una con su respectiva probabilidad, siendo la suma de probabilidades igual a uno. Aquí nos referiremos a la situación más habitual de tener dos posibilidades, valores o clases, la cual es también la de más fácil interpretación. Para el caso general de tener varias modalidades (ordenadas o no), el lector puede referirse a [Hastie et al. 2001] o [Peña 2002b].

Sea y_i una variable respuesta binaria:

$$y_i = \begin{cases} 1 & \text{Prob}_i(1) = p_i, \\ 0 & \text{Prob}_i(0) = 1 - p_i. \end{cases}$$

La variable de respuesta sigue pues una distribución binomial de parámetros 1 y p_i , donde p_i es la probabilidad de responder 1 para el individuo en cuestión, pudiendo ser diferente para cada individuo. Se ve que la parte sistemática del modelo (el valor esperado) es precisamente esta probabilidad:

$$\mu_i = E[y_i] = 1 \times p_i + 0 \times (1 - p_i) = p_i$$

La función de unión r^{-1} debe realizar una trasformación del predictor lineal en el intervalo [0,1]. Para ello la función más utilizada, si bien no la única, es la logística:

$$p_i = \frac{1}{1 + e^{-b'x_i}}$$

Esta relación también se escribe como:

$$\log \frac{p_i}{1 - p_i} = b'x_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$$

Esto es, el logit, definido como $\log(p_i/(1-p_i))$, es igual al predictor lineal clásico.

Para estimar los coeficientes del modelo, vamos pues a maximizar la función de (log)verosimilitud

$$\begin{aligned} l(b) &= \sum_{i=1}^n \log p_i = \sum_{i=1}^n (y_i \log p_i + (1 - y_i) \log(1 - p_i)) = \\ &= \sum_{i=1}^n \left(y_i \log \left(\frac{p_i}{1 - p_i} \right) + \log(1 - p_i) \right). \end{aligned}$$

Sustituyendo la función de unión en la expresión anterior, queda,

$$l(b) = \sum_i^n \left(y_i b' x_i - \log(1 + e^{b' x_i}) \right)$$

Obteniendo las derivadas respecto b , tenemos:

$$\frac{\partial l(b)}{\partial b} = \sum_{i=1}^n y_i x_i - \sum_{i=1}^n \left(\frac{1}{1 + e^{-b' x_i}} \right) x_i,$$

y expresando esta relación en notación matricial, queda,

$$\frac{\partial l(b)}{\partial b} = X'(y - p)$$

Por su lado, obteniendo las segundas derivadas de la función de (log)verosimilitud y haciendo las sustituciones apropiadas, se obtiene:

$$\frac{\partial^2 l(b)}{\partial b \partial b'} = - \sum_{i=1}^n \frac{e^{-b' x_i}}{(1 + e^{-b' x_i})^2} x_i x_i' = - \sum_{i=1}^n p_i (1 - p_i) x_i x_i',$$

la cual expresándolas en notación matricial, queda:

$$\frac{\partial^2 l(b)}{\partial b \partial b'} = -X'WX,$$

definiéndose W de la siguiente forma:

$$W = \begin{bmatrix} \ddots & & & \\ & p_i(1-p_i) & & \\ & & \ddots & \\ & & & \ddots \end{bmatrix}$$

Por tanto, la iteración de Newton-Raphson se escribe:

$$b^{t+1} = b^t + (X'WX)^{-1}X'(y - p) = (X'WX)^{-1}X'Wz,$$

con $z = Xb^t + W^{-1}(y - p)$.

Ello permite obtener el vector b de coeficientes a partir de los mínimos cuadrados ponderados, con matriz de ponderación W y tomando como vector de respuesta z , el cual a su vez depende de las probabilidades p , función a su vez de los coeficientes b .

Por tanto, el algoritmo de estimación de los coeficientes b consiste en la iteración por mínimos cuadrados ponderados a partir de una solución inicial. Un esquema del algoritmo es el siguiente:

1. Una solución inicial factible es $b = 0$.
2. Se estiman el vector p de probabilidades para cada individuo (que en esta primera iteración serán $p_i = 0,5$) y la matriz de ponderación W .
3. Se calcula el vector de respuesta z .
4. Se actualiza b por regresión ponderada.

A continuación se iteran los pasos 2 a 4 hasta la convergencia. Ésta no se garantiza siempre, aunque lo normal es que el algoritmo converja.

La desviación de la regresión logística tiene la conocida forma de la medida de información de Kullback:

$$D = 2 \sum_{i=1}^n o_i \log \frac{o_i}{e_i},$$

donde o_i denota las frecuencias observadas (y_i y $1-y_i$) y e_i denota las estimadas por el modelo (\hat{p}_i y $(1-\hat{p}_i)$).

7.7.4 Ejemplo. Análisis de los datos *Wisconsin Diagnostic Breast Cancer*

Como ejemplo de aplicación de los métodos de regresión logística tomamos los datos del archivo *wdbc.data* (véase Wisconsin Diagnostic Breast Cancer, Apéndice B). El problema consiste en encontrar una función de predicción del cáncer de pecho a partir de una serie de diez características medidas en los núcleos de las células. Estas características se han medido en imágenes digitalizadas de muestras del tejido orgánico. Para cada característica se ha anotado su valor medio en las células de la muestra, su desviación tipo y el caso más desfavorable, configurando en total 30 variables disponibles para la predicción. Existen 569 casos para los cuales se ha anotado el carácter benigno o maligno del tumor (la variable de respuesta).

A efectos de validar los resultados y para poder comparar los resultados con el Análisis Discriminante que se expondrá en la Sección 7.8.6, dividimos el conjunto de datos en dos

muestras tomadas al azar, una formada por 409 individuos que utilizaremos de entrenamiento y otra de 259 individuos que utilizaremos de validación.

Dado que las variables se dividen en realidad en tres grupos según el estadístico medido, podemos pensar en la existencia de correlaciones elevadas entre ellas. Para ello efectuamos un Análisis de Componentes Principales de las 30 variables explicativas (mediante el paquete estadístico SPAD).

En el gráfico de la Figura 7.16 damos el primer plano factorial obtenido representando en los ejes los dos primeros componentes principales y mostrando las 30 variables y sus condiciones con estos dos factores. Este gráfico es la mejor aproximación plana de la correlación existente entre las variables y ciertamente observamos variables altamente correlacionadas.

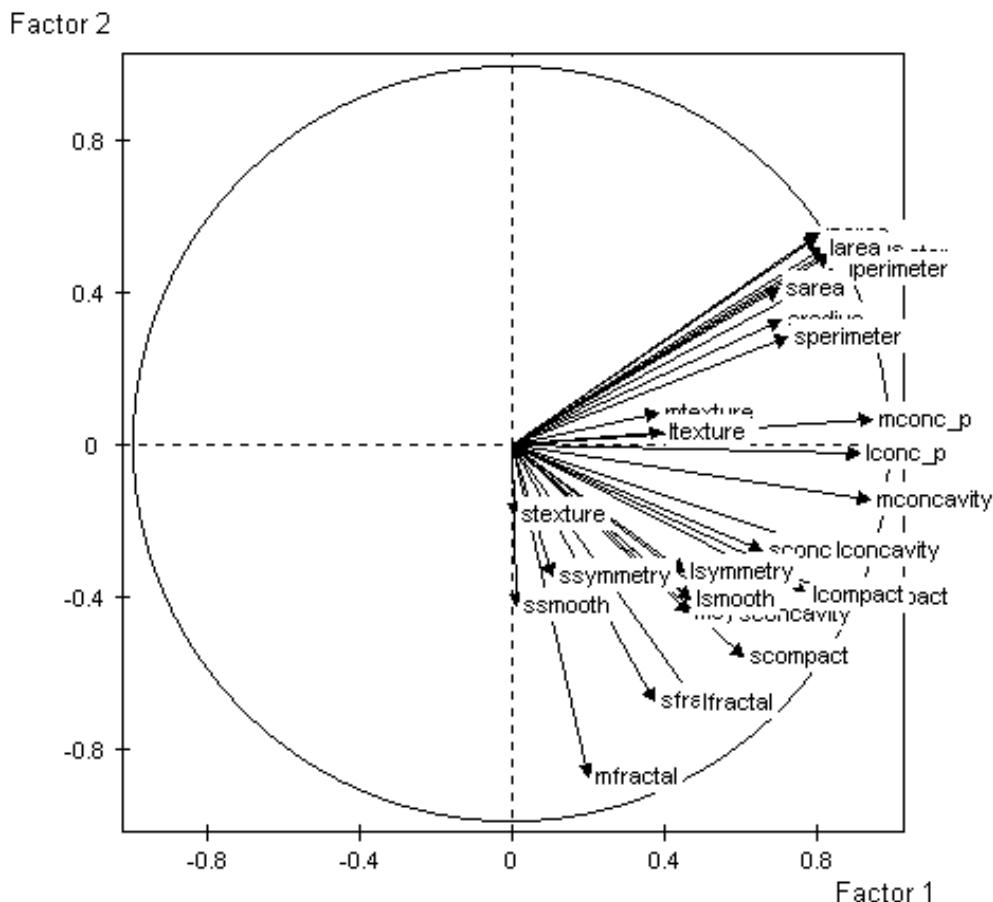


Figura 7.16. Primer plano factorial del ACP de las 30 variables explicativas del archivo Wisconsin Breast Cancer.

Como consecuencia de lo anterior, y observando también los *eigenvalues* (tanto por ciento de la varianza explicada) que no incluimos aquí por brevedad, decidimos efectuar la regresión logística de la variable de diagnóstico respecto los seis primeros factores del Análisis de Componentes Principales anterior (conteniendo casi el 90 por ciento de la información de las variables originales).

Los resultados obtenidos con la herramienta Minitab son los siguientes:

Link Function:	Logit
Response Information	
Variable	Value
diagnosi	M
	150 (Event)
	B
	259
	Total
	409

Observamos que por defecto se ha tomado el diagnóstico M (maligno) como suceso (*event*) lo cual significa que es la probabilidad de este suceso lo que se pretende estimar para cada individuo p_i .

En el cálculo de la función de (log)verosimilitud en cada iteración constatamos la convergencia del algoritmo.

Step	Log-Likelihood
0	-268.796
1	-106.368
2	-65.624
3	-45.467
4	-35.619
5	-31.505
6	-30.251
7	-30.067
8	-30.061
9	-30.061
10	-30.061

Finalmente, el modelo ajustado es:

Logistic Regression Table						
Predictor	Coef	SE Coef	Z	P	Odds Ratio	95% CI
					Lower	Upper
Constant	-0.7592	0.3613	-2.10	0.036		
F1_acp	3.2174	0.6138	5.24	0.000	24.96	7.50 83.14
F2_acp	1.6250	0.3678	4.42	0.000	5.08	2.47 10.44
F3_acp	0.5679	0.2123	2.68	0.007	1.76	1.16 2.68
F4_acp	0.7691	0.2440	3.15	0.002	2.16	1.34 3.48
F5_acp	-1.6238	0.4670	-3.48	0.001	0.20	0.08 0.49
F6_acp	0.5465	0.2602	2.10	0.036	1.73	1.04 2.88

Recordemos que este modelo da la combinación lineal de las variables explicativas (en este caso, las seis primeras componentes del ACP) que mejor predicen el “logit” de la probabilidad de padecer un cáncer maligno ($\log p_i/(1-p_i)$). En la columna Coef encontramos el valor de cada coeficiente b_j . La interpretación de estos coeficientes debe hacerse de acuerdo con la función de unión utilizada. El incremento de una unidad en un regresor x_j provoca un incremento lineal de b_j en el “logit”. La columna SE coef da las desviaciones tipo de estos coeficientes. Dividiendo la columna de coeficientes por sus desviaciones tipo, obtenemos la columna de los valores z , los cuales permiten probar la

hipótesis de no relación entre la variable de respuesta y cada regresor. Efectivamente, comprobamos en la columna P (p -valores del estadístico z) que todos los coeficientes son significativos (esto es, p -valores inferiores al umbral usual de 0,05). La columna Odds ratio se calcula como el exponencial del coeficiente y da el incremento de la ratio de la probabilidad de padecer cáncer respecto la probabilidad de no padecerlo²⁶ por cada unidad que aumenta la variable explicativa. Por ejemplo, en este caso, para cada unidad que aumenta la primera componente principal, implica que la ratio de la probabilidad de padecer cáncer respecto la probabilidad de no padecerlo, se multiplica por 25. Por último, la desviación tipo del coeficiente permite obtener el intervalo de confianza expresado en función de la “odds ratio”. Téngase en cuenta de que una “odds ratio” igual a 1 equivale a un coeficiente igual a 0, luego un intervalo que contenga el valor unidad significa que la variable explicativa no es significativa.

Deshaciendo la transformación de las componentes principales, tal como se explicó en la Sección 7.3.1, podemos expresar el modelo en función de las variables originales, lo cual es mucho más interpretable:

CONSTANTE		-49,4607
Mean	radius (mean of distances from center to points on the perimeter)	0,289
	texture (standard deviation of gray-scale values)	0,224
	perimeter	0,042
	area	0,003
	smoothness (local variation in radius lengths)	59,583
	compactness (perimeter ² / area - 1,0)	9,368
	concavity (severity of concave portions of the contour)	7,893
	concave points (number of concave portions of the contour)	26,1
	symmetry	16,712
	fractal dimension (“coastline approximation” - 1)	-42,65
Standard error	radius (mean of distances from center to points on the perimeter)	3,455
	texture (standard deviation of gray-scale values)	0,664
	perimeter	0,431
	area	0,02
	smoothness (local variation in radius lengths)	-18,996
	compactness (perimeter ² / area - 1,0)	-24,699
	concavity (severity of concave portions of the contour)	-18,702
	concave points (number of concave portions of the contour)	-14,137
	symmetry	-36,873
	fractal dimension (“coastline approximation” - 1)	-250,696
Worst	radius (mean of distances from center to points on the perimeter)	0,24
	texture (standard deviation of gray-scale values)	0,175
	perimeter	0,034
	area	0,002
	smoothness (local variation in radius lengths)	43,991
	compactness (perimeter ² / area - 1,0)	2,254
	concavity (severity of concave portions of the contour)	1,779
	concave points (number of concave portions of the contour)	12,822
	symmetry	7,45
	fractal dimension (“coastline approximation” - 1)	3,37

²⁶ Al cociente $p_i/(1-p_i)$ se le denomina *odd* en inglés sin una traducción consensuada al español.

Estos coeficientes permiten calcular fácilmente el valor del predictor lineal (“logit”) para cada individuo

$$\text{logit}_i = \log \frac{p_i}{1 - p_i} = -49.4607 + 0.289x_1 + 0.224x_2 + \dots$$

y por tanto, la probabilidad de padecer la enfermedad p_i .

$$p_i = \frac{e^{\text{logit}_i}}{1 + e^{\text{logit}_i}}$$

En la Figura 7.17 tenemos el gráfico de la transformación logística efectuada.

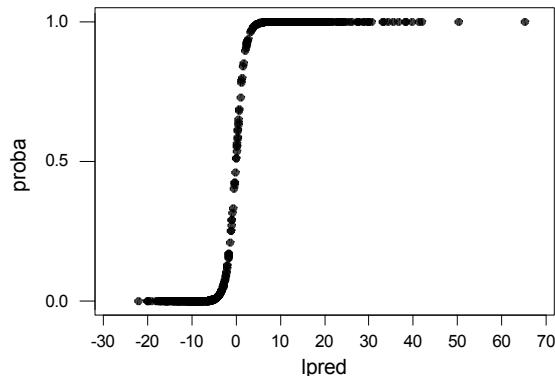


Figura 7.17. Función de transformación logística del predictor lineal en probabilidad.

Para convertir este estimador de probabilidades (clasificador suave) en un clasificador discriminante, podemos adoptar la política simple de considerar como personas en riesgo todas aquellas con probabilidad de 0,5 o superior y personas sin riesgo las personas con probabilidad inferior a 0,5, podemos obtener la tabla cruzando el diagnóstico real de la persona con la previsión efectuada por el modelo. Por supuesto, la fijación del umbral de riesgo, en la práctica, deben hacerla los profesionales del dominio que se está estudiando, usando por ejemplo los costes de clasificación errónea del problema o utilizando técnicas como el análisis ROC que se verá en el Capítulo 17.

Esta elección daría la siguiente matriz de confusión para el conjunto de entrenamiento:

Control: train = 1		Rows: diagnosi Columns: prob_rec	
		Sin riesgo	Riesgo
B	253	6	259
	97.68	2.32	100.00
M	4	146	150
	2.67	97.33	100.00
All	257	152	409
	62.84	37.16	100.00

Esto es, de las 259 personas con diagnóstico benigno, se han predicho correctamente 253 y de las 150 personas con diagnóstico maligno se han predicho correctamente 146, obteniéndose un porcentaje global de acierto del 97,5 por ciento.

Los datos anteriores se han obtenido con los mismos datos utilizados para la construcción del modelo, por lo que podemos validarla, utilizando los datos de la muestra test. Repitiendo el mismo proceso con la muestra test obtenemos la siguiente tabla:

Control: train = 0		
Rows: diagnosis	Columns: prob_rec	
Sin riesgo	Riesgo	All
B	98	0
	100.00	--
M	3	59
	4.84	95.16
All	101	59
	63.13	36.88
		100.00

Se obtiene un porcentaje global de acierto del 98,1 por ciento. Esto es, el porcentaje de aciertos es del mismo nivel. Este resultado, no muy habitual con otras técnicas, con ésta sí es bastante normal, puesto que utilizar las componentes principales como variables explicativas supone efectuar un especie de alisado (*smoothing*) sobre los datos y por tanto disminuir la variabilidad del ajuste (sobreajuste).

7.8 Análisis discriminante

El objetivo del análisis discriminante es encontrar reglas de asignación de individuos a una de las clases de una clasificación preestablecida. El término análisis discriminante es el nombre que se utiliza tradicionalmente en estadística para englobar las técnicas de clasificación supervisada.

Para resolver este problema disponemos de una muestra de n individuos, de los cuales sabemos su clase de pertenencia y en los que tenemos medidas un conjunto de p variables que suponemos cuantitativas y que permiten diferenciar a las clases.

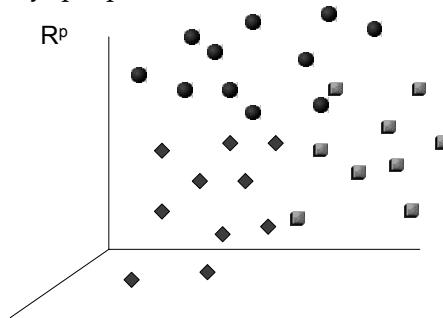


Figura 7.18. Nube de puntos en R^p según clase de pertenencia.

El problema del análisis discriminante no se diferencia del problema resuelto en modelización, para el caso de tener una variable de respuesta categórica, como es, por ejemplo, el caso de la regresión logística.

Geométricamente, las p variables permiten situar los individuos en un espacio R^p euclídeo (Figura 7.18). Por otro lado, suponemos que estas variables permiten diferenciar los individuos según su clase de pertenencia, es decir, suponemos que cada clase viene

definida por una distribución de probabilidad distinta de las restantes clases. Naturalmente, las variables x_j no tienen por qué ser las originales sino que pueden ser transformaciones que optimicen la separación entre las clases.

7.8.1 Las tres probabilidades

Sea q el número de clases. Denotamos por $\{C_1, \dots, C_q\}$ cada una de ellas. Para cada clase existe pues una función de densidad de probabilidad específica que denotamos por $f_k(x) = f(x|C_k)$, donde x representa el vector de p variables medidas en los individuos. Por otro lado, cada una de las clases puede tener una probabilidad *a priori* distinta $P(C_k)$. A partir del conocimiento de estas dos probabilidades determinaremos unas reglas de asignación a las clases de los nuevos individuos.

Las variables x_j no siempre van a permitir separar las clases entre sí, de hecho lo normal es que existan zonas de solapamiento entre las clases. Esto significa que al efectuar una asignación podemos equivocarnos (por ejemplo, un dígito que es un 7 lo podemos clasificar como un 9). Un criterio natural para efectuar la asignación es pues el de minimizar la probabilidad de mala clasificación o maximizar la de acierto.

Se demuestra que maximizar la probabilidad de acierto implica adoptar como regla de asignación la maximización de la probabilidad a posteriori $P(C_k|x)$ [McLachlan 1992; Peña 2002b]. La probabilidad a posteriori se calcula fácilmente mediante la fórmula de Bayes:

$$P(C_k | x) = \frac{f(x|C_k)P(C_k)}{f(x)}$$

siendo:

$$f(x) = \sum_{k=1}^q f(x|C_k)P(C_k),$$

la densidad de probabilidad en el punto x debida a todas las clases. Por tanto, la regla de asignación es dar la clase que maximice dicha probabilidad:

$$\arg \max_{k=1, \dots, q} P(C_k | x)$$

A esta regla de asignación se le denomina bayesiana.

Una justificación de esta regla es la siguiente: en ausencia total de información, la asignación de nuevos individuos la haremos al azar, por ejemplo en un problema de concesión automática de créditos, concederemos los créditos lanzando una moneda al aire, con lo que el porcentaje de error será del 50 por ciento. Si disponemos de la información de que el 60 por ciento de los créditos concedidos en el pasado resultaron buenos pagadores (esto es, disponemos de las probabilidades *a priori*), asignaremos los clientes a la clase de máxima probabilidad, esto es concederemos todos los créditos y nos equivocaremos en el 40 por ciento de los casos. Si además conocemos la edad de los clientes según tres tramos de edad con el mismo porcentaje de clientes y sabemos que la probabilidad de ser buen pagador en cada tramo es: 40 por ciento, 60 por ciento y 80 por ciento respectivamente (probabilidades *a posteriori*), utilizaremos estas probabilidades para la concesión de los créditos, es decir, denegaremos los créditos para el primer tramo de edad y concederemos en los otros dos, con lo que la probabilidad de error será del 33 por ciento.

7.8.2 Las funciones discriminantes

El denominador $f(x)$ de la fórmula de Bayes es constante, luego no hace falta evaluarlo, con lo que la regla puede simplificarse:

$$\arg \max_{k=1, \dots, q} f(x|C_k)P(C_k)$$

Esta regla define para cada punto del espacio su clase de asignación, luego define una partición del espacio R^p en regiones disjuntas (R_1, \dots, R_q). Cada región R_k queda asignada a una clase C_k . Puede comprobarse teóricamente que la regla definida minimiza la probabilidad de mala clasificación (véase, por ejemplo, el Apéndice 13.1 de [Peña 2002b]).

En la práctica, la asignación se realiza a partir de funciones monótonas de la probabilidad a posteriori $g_k(x)$, en particular la logarítmica $g_k(x) = \ln f_k(x) + \ln P(C_k)$. A estas funciones se les denomina *funciones discriminantes*. Dado un individuo x_0 , la asignación se hace evaluando cada función discriminante en este punto y asignándolo a la clase cuya función discriminante sea máxima.

$$x_0 \rightarrow C_k \quad \text{si } k = \arg \max_{j=1, \dots, q} \{g_j(x_0)\}$$

7.8.3 Discriminación paramétrica

En general, estimar las probabilidades a priori de las clases no representa un gran problema, una muestra representativa de la población bajo estudio puede proporcionar estimaciones razonables de las probabilidades de cada clase. Otra cuestión son las distribuciones condicionales $f_k(x)$. Estas distribuciones deben ser estimadas a partir de las muestras recogidas. La solución más simple consiste en suponer que siguen una distribución paramétrica especificada. La hipótesis clásica supone que siguen distribuciones normales multivariantes. Por tanto, se trata de estimar sus parámetros a partir de los datos recogidos.

Una distribución normal multivariante viene definida por su vector de medias μ , y su matriz de covarianzas Σ . El vector de medias identifica el punto del espacio R^p de máxima densidad, mientras que la matriz de covarianzas define la forma de la nube alrededor de este punto. Podemos imaginar pues cada distribución normal multivariante como un hiperelipsoide centrado en su vector de medias, como se ve en la Figura 7.19.

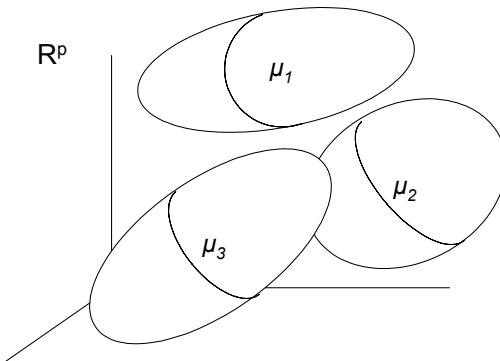


Figura 7.19. Tres distribuciones normales multivariantes con distintos vector de medias y matriz de covarianzas.

Discriminación cuadrática

El primer modelo para la discriminación es suponer que las q clases vienen definidas por q distribuciones multivariantes con media y matriz de covarianzas distintas.

$$C_k \sim N_p(\mu_k, \Sigma_k)$$

Su función de densidad de probabilidad se escribe como:

$$f(x|C_k) = \frac{1}{\sqrt{(2\pi)^p |\Sigma_k|}} \exp\left\{-\frac{1}{2}(x - \mu_k)' \Sigma_k^{-1} (x - \mu_k)\right\}$$

Por tanto, las funciones discriminantes son:

$$g_k(x) = -\frac{1}{2}(x - \mu_k)' \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \ln |\Sigma_k| + \ln P(C_k) \quad k = 1, \dots, q$$

A esta discriminación se le llama cuadrática porque los límites de las regiones R_k vienen definidos por hiperelipsoides (véase Figura 7.20).

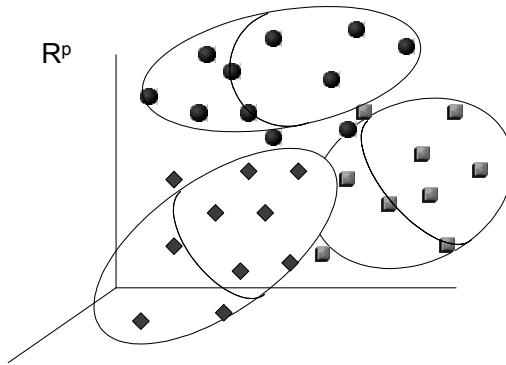


Figura 7.20. Representación simplificada de discriminación cuadrática por hiperelipsoides.

Esta regla de asignación es bastante flexible. Para utilizarla deberemos tener estimaciones fiables de los parámetros. Los vectores μ_k se estimarán por la media muestral \bar{x}_k y las matrices de covarianza Σ_k por las matrices de covarianza muestrales por clase V_k . La estimación de las medias generalmente no presenta problema, pero la estimación de cada matriz de covarianzas significa estimar $p(p+1)/2$ parámetros. Si disponemos de pocos datos por clase y/o mucha dimensionalidad, estas estimaciones serán imprecisas y dependientes de los datos recogidos, es decir, tendremos un sobreajuste (*overfitting*), por lo que una actitud prudente es simplificar el modelo.

Discriminación lineal

La simplificación consiste en suponer que todas las clases tienen idéntica matriz de covarianzas (esto es, los hiperelipsoides tienen idéntica forma). Entonces, el modelo paramétrico de discriminación se escribe ahora como:

$$C_k \sim N_p(\mu_k, \Sigma),$$

$$g_k(x) = \mu_k' x - \frac{1}{2} \mu_k' \Sigma^{-1} \mu_k + \ln P(C_k) \quad k = 1, \dots, q,$$

Como se puede observar, las $g_k(x)$ son funciones lineales de x . La estimación de la matriz de covarianzas conjunta W se hace ponderando las matrices de covarianzas de cada grupo:

$$W = \frac{\sum_{k=1}^q (n_k - 1)V_k}{n - q}$$

En caso de tener suficientes datos por clase, debe realizarse una prueba de igualdad de las matrices de covarianzas antes de proceder a una discriminación lineal [Krzanowski 1996].

Discriminación lineal con probabilidades a priori iguales

Existen situaciones en las cuales no es posible conocer las probabilidades a priori de las clases. En estos casos lo más razonable es suponer que todas las clases tienen la misma probabilidad; esto significa que a priori tenemos la máxima incertidumbre sobre la clase de pertenencia y que la asignación sólo tendrá en cuenta los atributos medidos en los individuos. Por supuesto, también utilizaremos este supuesto para el caso de que efectivamente las probabilidades a priori sean iguales.

Entonces las funciones discriminantes se escriben:

$$g_k(x) = \mu_k \Sigma^{-1} x - \frac{1}{2} \mu_k \Sigma^{-1} \mu_k \quad k = 1, \dots, q$$

Desaparece el término de la probabilidad a priori y manipulando la expresión anterior podemos escribirla como:

$$g_k(x) = -\frac{1}{2}(x - \mu_k)' \Sigma^{-1} (x - \mu_k)$$

Esta asignación resulta ser equivalente a minimizar la distancia de Mahalanobis²⁷ del individuo al centroide de cada grupo: $(x - \mu_k)' \Sigma^{-1} (x - \mu_k)$. El resultado sería algo similar a lo mostrado en la Figura 7.21.

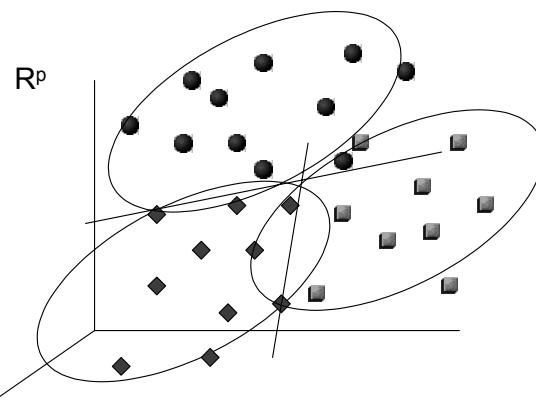


Figura 7.21. Representación simplificada de la discriminación lineal.

²⁷ La distancia o métrica de Mahalanobis se define utilizando la matriz de covarianzas (Σ^{-1}) y difiere de la euclídea cuando existe correlación entre las variables. En el Capítulo 16, veremos más información sobre distintas medidas de distancia.

7.8.4 Función discriminante de Fisher

La función discriminante de Fisher, llamada así en honor de quien la propuso por primera vez, se obtiene como la búsqueda de la combinación lineal a que maximiza la varianza de la separación entre las dos clases, respecto de la varianza dentro de las clases.

$$\text{Max } \frac{a' B a}{a' W a},$$

donde a representa una dirección cualquiera de R^p , B la matriz de covarianzas de los centros de gravedad de las clases de término general

$$b_{jj'} = \frac{1}{n-1} \sum_{k=1}^q n_k (\bar{x}_{jk} - \bar{x}_j)(\bar{x}_{j'k} - \bar{x}_{j'}),$$

y W es la matriz de covarianza conjunta de las clases. La maximización anterior significa buscar la dirección de R^p tal que la separación entre centros de gravedad sea máxima respecto la variabilidad de las clases (en análisis discriminante se espera que este cociente sea claramente mayor que uno). La Figura 7.22 ilustra la función a que se busca para el caso de tener sólo dos clases.

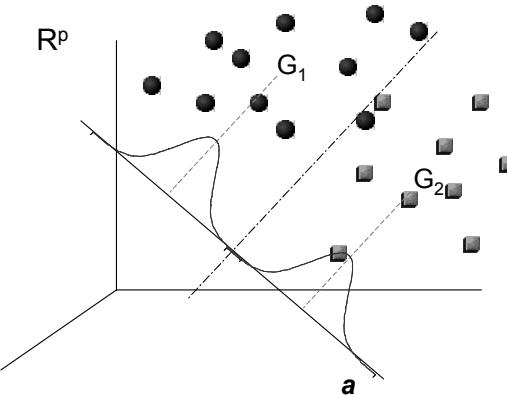


Figura 7.22. Función discriminante de Fisher para el caso de dos grupos.

El máximo corresponde al vector propio de $W^{-1}B$ asociado al mayor valor propio,

$$W^{-1} B a = \lambda a$$

La matriz B en el caso de dos clases es función de la diferencia entre los dos centros de gravedad de las clases:

$$B = \frac{n_1 n_2}{n} (\bar{x}_1 - \bar{x}_2)(\bar{x}_1 - \bar{x}_2)'$$

Por tanto, sustituyendo en la expresión anterior y teniendo en cuenta que $(\bar{x}_1 - \bar{x}_2)'a$ es una constante, resulta :

$$a = Cte \times W^{-1}(\bar{x}_1 - \bar{x}_2)$$

Prescindiendo de la constante, esta fórmula permite encontrar la función discriminante de Fisher de forma muy fácil.

La asignación de nuevos individuos x_0 se efectúa proyectando estos individuos sobre la función a y comparando esta proyección con el punto medio de la proyección de los centros de gravedad de los dos grupos. Esto es,

si $a'x_0 > \frac{a'G_1 + a'G_2}{2}$ entonces x_0 se asigna a C_1 ,
en caso contrario se asigna a C_2 .

La función discriminante de Fisher es proporcional a la función discriminante $g(x)$ para el caso de dos grupos con hipótesis de equicovarianza. En efecto, en el caso de dos grupos la función discriminante se define como el cociente:

$$g(x) = \ln \left(\frac{f(x|C_1) P(C_1)}{f(x|C_2) P(C_2)} \right) = \frac{1}{2} \left[(x - \mu_2)' \Sigma_2^{-1} (x - \mu_2) - (x - \mu_1)' \Sigma_1^{-1} (x - \mu_1) \right] + \ln \frac{P(C_1)}{P(C_2)}$$

Suponiendo la hipótesis de equicovarianza,

$$g(x) = (\mu_1 - \mu_2)' \Sigma^{-1} x - \frac{1}{2} \mu_1' \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_2' \Sigma^{-1} \mu_2 + \ln \frac{P(C_1)}{P(C_2)},$$

y suponiendo igual probabilidad a priori, la expresión anterior se escribe como:

$$g(x) = (\mu_1 - \mu_2)' \Sigma^{-1} x - \frac{1}{2} (\mu_1 - \mu_2)' \Sigma^{-1} (\mu_1 + \mu_2)$$

Sustituyendo Σ por su estimador W , y μ_1 y μ_2 por sus estimadores G_1 y G_2 , (o lo que es lo mismo \bar{x}_1 y \bar{x}_2) tenemos:

$$g(x) = a'x - \frac{a'G_1 + a'G_2}{2}$$

Es decir, la dirección de R^p que mejor separa las clases coincide (salvo una constante) con la función discriminante con hipótesis de normalidad multivariante, equicovarianza y probabilidades a priori iguales. El interés de la función discriminante de Fisher es que se trata de una regla geométrica, por tanto siempre la podremos aplicar y en el caso de verificarse las hipótesis reseñadas, es óptima. También puede demostrarse la proporcionalidad entre la función discriminante de Fisher y los coeficientes de regresión múltiple de la variable de respuesta binaria, en general codificada de forma particular (véase la página 347 de [Lebart et al. 1984]), respecto de las variables discriminantes X :

$$y = \begin{cases} \sqrt{\frac{n_2}{n_1}} & \text{si } C_1 \\ -\sqrt{\frac{n_1}{n_2}} & \text{si } C_2 \end{cases} \quad b = (X'X)^{-1} X'y = Cte \times a$$

Por tanto, los coeficientes de regresión son proporcionales a los coeficientes de la función discriminante (véase el Capítulo 4 de [Hastie et al. 2001]). Este resultado es útil porque permite obtener la función discriminante de Fisher mediante un programa de regresión lineal.

7.8.5 Consideraciones sobre la calidad de la discriminación

La calidad de las funciones discriminantes vistas en los apartados anteriores depende de muchos factores. Por ejemplo, cabe señalar que en análisis discriminante también se producen los problemas de colinealidad entre las variables x_j y presencia de fluctuación aleatoria, por lo que también tiene sentido realizar la discriminación sobre componentes ortogonales de un Análisis de Componentes Principales o PLS [Tenenhaus 1998].

En cualquier caso, la probabilidad de acierto (precisión, o su inversa el error) es la mejor medida de calidad de una discriminación. La forma usual de estimarla es a partir de los datos. Para obtener una estimación honesta de la calidad de la discriminación basta tener una muestra de aprendizaje (conjunto de entrenamiento), con la que se obtendrán las funciones discriminantes, y una muestra de validación (conjunto de prueba), independiente de la anterior, en la que se repetirá el proceso anterior. En caso de no poder disponer de las dos muestras o de tener muy pocos datos como para poder partirlos en dos muestras, podemos utilizar el método de validación cruzada, que se ha comentado brevemente en el Capítulo 2, en secciones anteriores y del que se trata en profundidad en el Capítulo 17.

7.8.6 Ejemplo. Análisis de los datos *Wisconsin Diagnostic Breast Cancer*

Retomamos los datos del *Wisconsin Diagnostic Breast Cancer* usados en el ejemplo de regresión logística, (Sección 7.7.4) con las mismas submuestras de entrenamiento y aprendizaje a fin de poder comparar los resultados. Igualmente tomamos las seis primeras componentes principales obtenidas allí para realizar el análisis discriminante. La función discriminante de Fisher obtenida en la situación de máxima incertidumbre (esto es con probabilidades a priori iguales a 0,5) es la siguiente:

TWO GROUPS DISCRIMINANT ANALYSIS		DISCRIMINANT LINEAR FUNCTION				
VARIABLES	CORRELATIONS	COEFFICIENTS	STANDARD	T	PROB.	
.....	VARIABLES	DISCRIM.	REGRESSION	DEV.	STUDENT	
NUM LABELS	WITH D.L.F.	FUNCTION				(RES. TYPE REG.)
		(THRESHOLD= 0.10)				
33 F1_acp	0.787	1.4708	0.2179	0.0076	28.52	0.000
34 F2_acp	0.151	0.4278	0.0634	0.0116	5.46	0.000
35 F3_acp	0.173	0.6924	0.1026	0.0164	6.27	0.000
36 F4_acp	0.122	0.5784	0.0857	0.0194	4.41	0.000
37 F5_acp	-0.085	-0.4520	-0.0670	0.0218	3.07	0.002
38 F6_acp	0.025	0.1483	0.0220	0.0239	0.92	0.358
CONSTANT		-1.295688	0.000009	0.0276	0.0003	0.9997

Las distintas columnas dan sucesivamente la correlación de las variables discriminantes (en este caso las seis componentes principales) con la función discriminante de Fisher (D.L.F.). Obsérvese la elevada correlación entre la primera componente principal y la función discriminante de Fisher. Threshold = 0.10 indica el umbral de significación para las correlaciones, es decir, en este caso las correlaciones inferiores a 0,10 en valor absoluto no son significativas. A continuación vienen los coeficientes de la función discriminante de Fisher, los coeficientes de la regresión equivalente con su respectiva desviación tipo, la *t* de Student y su respectivo *p*-valor, que permite probar la significación de los coeficientes. En este caso vemos que la última componente F6_acp tiene un *p*-valor superior a 0,05 y por tanto es no significativa. Sin embargo para mantener la comparabilidad con el modelo de regresión logística, decidimos mantenerla en la función discriminante.

Clasificando los 409 individuos de la muestra de entrenamiento obtenemos la siguiente matriz de confusión (ordenada por clasificaciones correctas e incorrectas):

CLASSIFICATION TABLE		CLASSIFICATION COUNTS AND (PERCENTAGES)		
ORIGINAL GROUPS		WELL CLASSIFIED	MISCLASSIFIED	TOTAL
	M	136.00 (90.67)	14.00 (9.33)	150.00 (100.00)
	B	257.00 (99.23)	2.00 (0.77)	259.00 (100.00)
	TOTAL	393.00 (96.09)	16.00 (3.91)	409.00 (100.00)

Obtenemos un porcentaje global de acierto del 96,1 por ciento, sin embargo bastante desequilibrado según se trate del grupo maligno (error 9,33 por ciento) o del grupo benigno (error 0,77 por ciento). Aplicando la misma regla de discriminación a la muestra test obtenemos la siguiente tabla

CLASSIFICATION TABLE		CLASSIFICATION COUNTS AND (PERCENTAGES)		
ORIGINAL GROUPS		WELL CLASSIFIED	MISCLASSIFIED	TOTAL
	M	58.00 (93.55)	4.00 (6.45)	62.00 (100.00)
	B	98.00 (100.00)	0.00 (0.00)	98.00 (100.00)
	TOTAL	156.00 (97.50)	4.00 (2.50)	160.00 (100.00)

Ahora el porcentaje de acierto es aún superior, del 97,5 por ciento, si bien ligeramente inferior al conseguido con regresión logística. Continúa dando un porcentaje de acierto superior en el grupo benigno que en el maligno²⁸. Por supuesto, deshaciendo el cambio de los componentes principales, podemos expresar la función discriminante en función de las variables originales.

Respecto de las ventajas de utilizar regresión logística o el análisis discriminante de Fisher, no existe consenso entre los autores sobre cuál de los dos métodos es más seguro; véanse los comentarios de comparación contradictorios en la página 103 de [Hastie et al. 2001] y en la página 434 de [Peña 2002b]. Existen numerosos estudios de comparación sin que a partir de ellos pueda establecerse una conclusión definitiva.

7.9 Sistemas, aplicabilidad y recomendaciones de uso

En este capítulo hemos visto muchas técnicas para realizar regresión y clasificación, y han inaugurado un gran abanico de técnicas diversas que se ven a lo largo de esta tercera parte del libro. Las técnicas de modelización paramétrica no presentan problemas especiales para el tratamiento de grandes volúmenes de datos. De hecho, la regresión lineal y sus variantes son de los métodos más eficientes de regresión. Del mismo modo, la regresión logística y

²⁸ En la práctica deberíamos considerar los distintos costos de mala clasificación o el análisis ROC antes de dar la regla definitiva de asignación de los pacientes en una de las dos clases.

las funciones discriminantes de Fisher son métodos muy eficientes de clasificación. Por tanto, son algunas de las primeras técnicas a utilizar ante problemas de regresión y clasificación, en especial cuando los atributos originales son mayoritariamente cuantitativos (numéricos).

Para facilitar aún más su utilización, la modelización estadística paramétrica se encuentra en todos los paquetes estadísticos generalistas (paquetes estadísticos de propósito general), comerciales, tales como SPSS, SAS, etc., o gratuitos, como R (<http://www.r-project.org>) [The R Development Core Team 2003], así como en muchas herramientas propias de minería de datos (WEKA, Clementine, etc.). En este capítulo, hemos utilizado en muchos ejemplos el paquete MINITAB (<http://www.minitab.com>), por su facilidad de uso y alto grado de interactividad. Las extensiones presentadas sobre la regresión sobre componentes no correlacionadas pueden encontrarse además en paquetes especializados, tales como SPAD (<http://www.decisia.fr>), SIMCA [Wold & Ketaneh 1996], entre otros, mientras que los modelos lineales generalizados (además de la regresión logística, que suele estar en casi todos) han sido desarrollados en GLIM [Aitkin et al. 1989], S-PLUS [Chambers & Hastie 1993], R, y otros muchos.

Capítulo 8

MODELIZACIÓN ESTADÍSTICA

NO PARAMÉTRICA

Pedro Delicado y Tomàs Aluja

En el capítulo anterior introdujimos métodos y técnicas paramétricas de modelización: regresión lineal, lineal generalizada (por ejemplo logística) y discriminantes paramétricos (por ejemplo de Fisher). En este capítulo se presenta la modelización no paramétrica, en particular aquella que hace uso de métodos núcleo (*kernel*) para poder construir modelos más flexibles, al ajustarse a los datos de una manera *local*. Se abarca aquí desde la regresión no paramétrica hasta los modelos aditivos generalizados aplicados al problema del análisis discriminante.

Este capítulo constituye, por tanto, una continuación del anterior, y se recomienda haberlo leído o conocer los conceptos de modelización paramétrica del capítulo anterior.

8.1 Introducción

Los métodos estadísticos descritos en el capítulo anterior se basan en la hipótesis de que los datos analizados siguen un cierto modelo matemático que determina tanto la parte sistemática de las observaciones como su componente aleatoria. La mayoría de estos modelos plantean el problema suponiendo que los datos son el resultado de observar variables aleatorias de distribución conocida, salvo por la presencia de algunos parámetros cuyo valor se desconoce. Por ejemplo, la relación entre el peso (y) y la altura (x) de un grupo de personas puede modelarse mediante regresión lineal con errores normales:

$$y = \beta_0 + \beta_1 x + \varepsilon, \text{ con } \varepsilon \sim N(0, \sigma^2)$$

Éste es un modelo estadístico con tres parámetros desconocidos: β_0 , β_1 y σ^2 . Los modelos que dependen de un número finito de parámetros se denominan *modelos paramétricos*. Se dice que se *ajusta* el modelo cuando se estiman sus parámetros a partir de un conjunto de observaciones que siguen dicho modelo. Entre las ventajas de los modelos paramétricos

está la de que pueden ser ajustados con una cantidad pequeña de datos (por ejemplo, puede usarse el método de mínimos cuadrados o el de máxima verosimilitud; véase [Peña 2001]). Además, en muchas ocasiones los parámetros tienen una interpretación intuitiva en términos relacionados con el problema en estudio (por ejemplo, β_1 es la derivada de y respecto de x en el modelo de regresión anterior).

Sin embargo, los modelos paramétricos presentan un problema fundamental: su estructura es tan rígida que no pueden adaptarse a muchos conjuntos de datos. Consideremos el siguiente ejemplo, en el que se analiza la relación entre dos variables del conjunto de datos referido a la vivienda en 506 barrios de Boston en 1978 (*Boston Housing Data*, descrito en el Apéndice B). Concretamente, se busca expresar la variable `room` (número medio de habitaciones por vivienda) como función de la variable `lstat` (porcentaje de población con estatus social en la categoría inferior). Para ello podemos utilizar un modelo de regresión lineal que ajuste la variable `room` como función de la variable `lstat`. El resultado se muestra en el panel izquierdo de la Figura 8.1. Se observa que el patrón lineal impuesto por el modelo paramétrico elegido es muy rígido para adaptarse a la relación existente entre las variables. Esto se debe a que la relación entre variables no es lineal: la variable `room` desciende bruscamente cuando la variable `lstat` pasa del cero por ciento al diez por ciento, pero después se mantiene prácticamente constante.

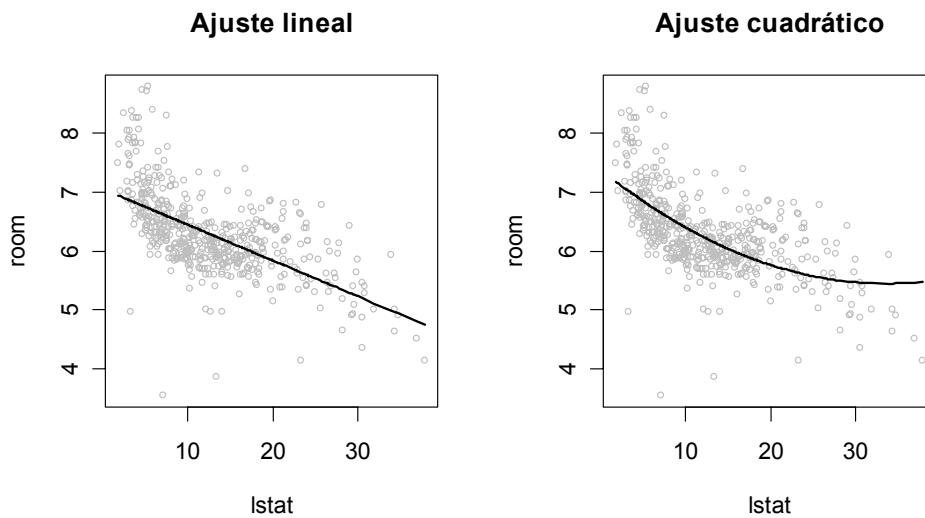


Figura 8.1. Ajustes paramétricos de la variable `room` como función de la variable `lstat`.

Una posible solución, como vimos en el capítulo anterior, sería introducir un término cuadrático (el cuadrado de `lstat`) en el modelo de regresión para reflejar la ausencia de linealidad. El segundo panel de la Figura 8.1 muestra el resultado. Pero, aun así, el nuevo modelo de regresión paramétrico no consigue adaptarse totalmente a los datos observados.

En este capítulo presentaremos una alternativa no paramétrica a los modelos estadísticos paramétricos usuales. Por ejemplo, la relación entre el peso y la altura de una persona podría modelizarse no paramétricamente diciendo que

$$y = r(x) + \varepsilon,$$

donde $r(x)$ es una función (posiblemente continua o derivable) cuya forma no se especifica, y ε es una variable aleatoria con valor esperado igual a cero.

Para terminar esta introducción, y como adelanto de los siguientes apartados, realizamos el ajuste no paramétrico de la variable `room` como función de la variable `lstat` (conjunto de datos *Boston Housing Data*). La Figura 8.2 muestra los resultados de este ajuste no paramétrico. La relación entre las variables es distinta según si el porcentaje de población de extracción social más baja (`lstat`) es inferior al 10 por ciento, está entre el 10 por ciento y el 20 por ciento, o supera ese valor. En el tramo intermedio, el número medio de habitaciones por vivienda (`room`) se mantiene constante, mientras que en los otros tramos decrece al crecer `lstat`. Además, la disminución es más acusada en el primer tramo.

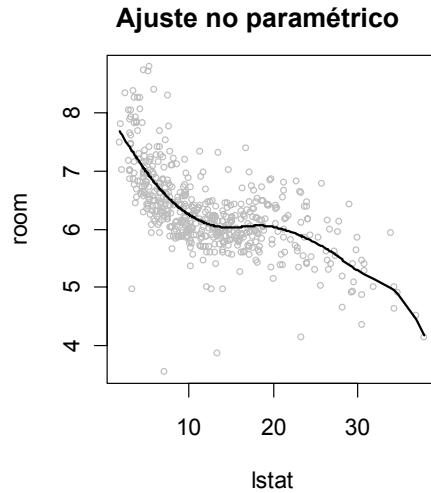


Figura 8.2. Ajuste no paramétrico de la variable `room` como función de la variable `lstat`.

Este ejemplo muestra que la modelización no paramétrica es mucho más flexible que la paramétrica y permite resaltar los patrones de dependencia presentes en los datos.

8.2 Regresión no paramétrica

Comencemos con el contexto del modelo de regresión simple: la variable de respuesta y es continua y sólo hay una variable explicativa x , también continua (el caso de la regresión múltiple lo abordaremos después). Se supone que se observan n pares de datos (x_i, y_i) que siguen el modelo de regresión no paramétrico siguiente:

$$y_i = r(x_i) + \varepsilon_i$$

Además, se suele considerar que el valor esperado $E(\cdot)$ del error es 0 y su varianza $V(\cdot)$ es constante (homocedasticidad), es decir,

$$E(\varepsilon_i) = 0 \text{ y } V(\varepsilon_i) = \sigma^2 \text{ para todo } i$$

No se especifica la forma funcional de la función de regresión $r(x)$, aunque sí se supone que es una función suficientemente regular (por ejemplo, es habitual la hipótesis de que $r(x)$ tiene segunda derivada continua). La hipótesis de *homocedasticidad* (varianza de los errores constante) puede relajarse y suponerse que esa varianza es función de la variable explicativa x .

Una vez establecido el modelo, el paso siguiente consiste en estimarlo (o ajustarlo) a partir de las n observaciones disponibles. Es decir, hay que construir un estimador $\hat{r}(x)$ de la función de regresión y un estimador $\hat{\sigma}^2$ de la varianza del error. Los procedimientos de estimación de $\hat{r}(x)$ también se conocen como *métodos de alisado o suavizado (smoothing)*.

El abanico de técnicas disponibles para estimar no paramétricamente la función de regresión es amplísimo e incluye, entre otras, las siguientes:

- *Ajuste local de modelos paramétricos.* Se basa en hacer varios (o incluso infinitos) ajustes paramétricos teniendo en cuenta únicamente los datos cercanos al punto donde se desea estimar la función. Son las que desarrollaremos en este capítulo.
- *Métodos basados en series ortogonales de funciones.* Se elige una base ortonormal del espacio vectorial de funciones y se estiman los coeficientes del desarrollo en esa base de la función de regresión. Los ajustes por series de Fourier y mediante *wavelets* son los dos enfoques más utilizados. Una introducción a este tema (y referencias para profundizar) puede verse en la Sección 2.5 de [Fan & Gijbels 1996].
- *Suavizado mediante splines.* Se plantea el problema de buscar la función $\hat{r}(x)$ que minimiza la suma de los cuadrados de los errores ($e_i = y_i - \hat{r}(x_i)$) más un término que penaliza la falta de suavidad de las funciones $\hat{r}(x)$ candidatas (en términos de la integral del cuadrado de su derivada segunda). Se puede probar que la solución es un *spline* cúbico con nodos en los puntos x_i observados. Véanse [Wahba 1990; Green & Silverman 1994] y [Eubank 1999].
- *Técnicas de aprendizaje supervisado.* Las redes neuronales, los k vecinos más cercanos y los árboles de regresión se usan habitualmente para estimar $r(x)$. De hecho cualquier técnica de aprendizaje supervisado que admita respuesta continua y predictor continuo puede usarse para estimar no paramétricamente la función de regresión. Trataremos todas estas técnicas en los capítulos siguientes de esta Parte III del libro. Para una visión desde un punto de vista estadístico pueden consultarse los capítulos 9, 11 y 13 de [Hastie et al. 2001].

En gran parte, este abanico no sólo existe para regresión sino que también se repite para discriminación.

8.2.1 Estimadores núcleo y ajuste local de polinomios

La observación de la nube de puntos formada por las observaciones de las variables `lstat` y `room`, así como del resultado del ajuste de la regresión lineal simple a estos datos mostrada en la Figura 8.1, sugieren que un único modelo lineal no es válido para todo el rango de la variable explicativa `lstat`. La primera idea que surge para solventar ese problema es dividir el rango de esta variable en intervalos, de forma que la relación entre las dos variables sea aproximadamente lineal en cada intervalo. Así, parece razonable considerar los cuatro intervalos delimitados por los valores 10 por ciento, 20 por ciento y 30 por ciento de la variable `lstat`. Hecho esto, en cada intervalo se ajusta un modelo de regresión lineal simple y se obtiene el resultado que muestra la Figura 8.3.

Este sencillo ejercicio refleja todavía más claramente que los datos sugieren una relación entre las dos variables que no se ajusta a una regresión simple. Sin embargo el resultado obtenido dista mucho de ser completamente satisfactorio por varios motivos. En primer lugar, la función de regresión estimada mediante este procedimiento es discontinua

en los puntos que definen los intervalos. En segundo lugar, en la estimación del valor de la función de regresión en puntos cercanos a los extremos de los intervalos (por ejemplo, en $lstat=9$) no intervienen muchos datos cercanos a ese valor de la variable explicativa y que pertenecen a un intervalo distinto (no se usan los datos para los que $lstat$ está entre 10 y 14, por ejemplo) mientras que esa estimación sí se ve afectada por algunos datos del mismo intervalo que están más lejos (en nuestro ejemplo los datos para los que $lstat$ es menor que 4).

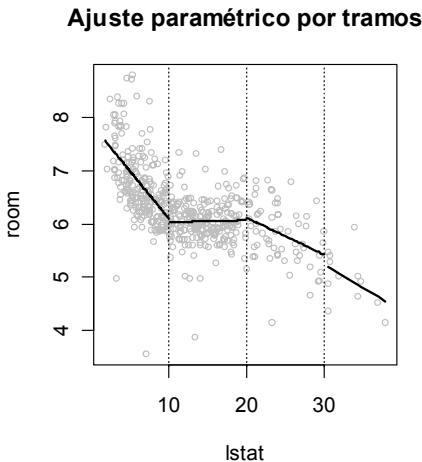


Figura 8.3. Ajuste de modelos de regresión lineal simple en cuatro intervalos del recorrido de la variable $lstat$.

Una forma de atacar la segunda de las deficiencias mencionadas es la siguiente. Para estimar la función de regresión en un valor concreto t de la variable explicativa, se debe usar un intervalo de la variable explicativa específico para ese valor t , centrado en el valor t y que sólo contenga datos en los que la variable explicativa tome valores cercanos a t . Así, si se desea estimar la función de regresión en el valor $lstat=9$, se usarán únicamente las observaciones para las cuales $4 < lstat < 14$ (si es que queremos que los nuevos intervalos sigan teniendo 10 unidades de amplitud). Este procedimiento se ilustra en el panel izquierdo de la Figura 8.4.

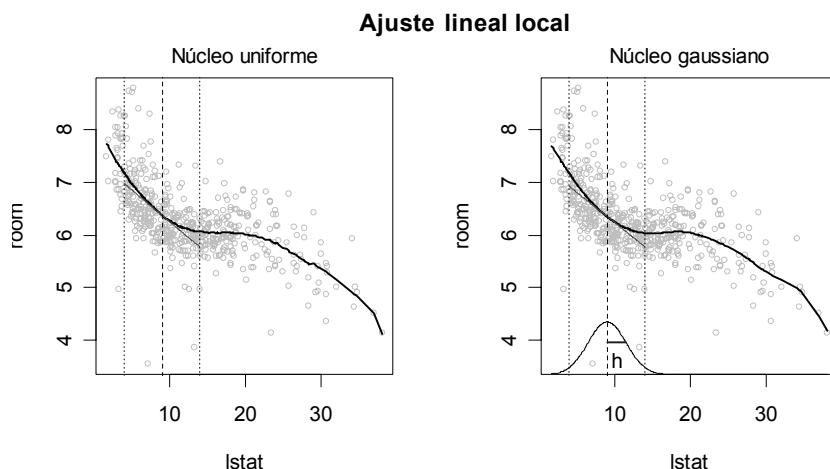


Figura 8.4. Ajuste lineal local en el punto $lstat=9$. Ponderación uniforme (izda.). Con núcleo gaussiano (dcha.).

Pese a la apariencia de continuidad de la función de regresión representada en este gráfico, el método descrito no proporciona estimadores continuos. Ello se debe a que, al desplazar a la derecha el intervalo que determina los datos activos para el cálculo de la regresión simple local, habrá puntos que dejen de ser activos (los situados más a la izquierda) y otros que pasen a serlo (los que estaban cerca del intervalo en su parte derecha). El hecho de que un dato pase de ser activo a no serlo, y viceversa, de forma abrupta (su peso en la regresión simple pasa de ser 0 a ser 1) hace que la función de regresión estimada no sea continua.

La continuidad del estimador puede conseguirse si se ponderan los datos de forma que el peso de una observación (x_i, y_i) sea función decreciente (que tienda a 0 y sea continua) de la distancia de su ordenada x_i al punto t donde se realiza la estimación. De esta forma, al desplazar el punto t , las observaciones irán tomando todos los valores posibles de la función peso de forma continua en t y, como resultado, se tendrá un estimador continuo de la función de regresión. Esto se ilustra en el panel derecho de la Figura 8.4.

La forma usual de asignar estos pesos es mediante una función núcleo (*kernel*) K (función simétrica no negativa, continua, decreciente en $[0, \infty)$ y que tiende a 0 cuando el argumento tiende a infinito). El peso de (x_i, y_i) en la estimación de $r(t)$ será

$$w_i = w(t, x_i) = \frac{K\left(\frac{x_i - t}{h}\right)}{\sum_{j=1}^n K\left(\frac{x_j - t}{h}\right)},$$

donde h es un parámetro de escala que controla la concentración del peso total alrededor de t : si h es pequeño únicamente las observaciones más cercanas a t tendrán peso relevante, mientras que valores grandes de h permiten que observaciones más alejadas de t también intervengan en la estimación de $r(t)$. A h se le denomina *parámetro de suavizado* (o *ventana*) del estimador no paramétrico y permite controlar el grado de localidad (o globalidad) de la estimación. La estimación final se ve notablemente afectada por cambios en la elección del parámetro de suavizado, por lo que esta tarea resulta crucial en la estimación no paramétrica. A la elección de h nos dedicaremos en una sección más adelante (página 221).

Una vez determinados los pesos w_i , se resuelve el problema de mínimos cuadrados ponderados siguiente:

$$\min_{a,b} \sum_{i=1}^n w_i (y_i - (a + b(x_i - t))^2$$

Los parámetros a y b así obtenidos dependen de t , porque los pesos w_i dependen de t : $a=a(t)$, $b=b(t)$. La recta de regresión ajustada localmente alrededor de t es

$$l_t(x) = a(t) + b(t)(x - t),$$

y la estimación de la función de regresión en el punto t es el valor que toma esa recta en $x=t$:

$$\hat{r}(t) = l_t(t) = a(t)$$

El panel derecho de la Figura 8.4 muestra el estimador así construido para los datos (`lstat, room`). Se indica también la recta de regresión estimada por mínimos cuadrados ponderados en el punto `lstat=9`. En este caso se ha usado como función núcleo K la función densidad de la variable aleatoria normal estándar, que se conoce como núcleo

gaussiano. Se ha representado el núcleo en el mismo gráfico para ilustrar cómo se asignan los pesos a los distintos datos según la proximidad a 9 de los valores de la variable `lstat`. Se ha señalado también el valor h del parámetro de suavizado.

NUCLEO	FÓRMULA DE $K(x)$	CARACTERÍSTICAS
Uniforme	$(1/2) I_{[-1,1]}(x)$	Discontinuo, con soporte compacto
Gaussiano	$(1/\sqrt{2\pi}) \exp(-x^2/2)$	Continuo, derivable, con soporte no compacto
Epanechnikov	$(3/4)(1-x^2) I_{[-1,1]}(x)$	Continuo, no derivable, con soporte compacto
Biweight	$(15/16)(1-x^2)^2 I_{[-1,1]}(x)$	Continuo, derivable, con soporte compacto

Tabla 8.1. Funciones núcleo usadas habitualmente en regresión no paramétrica.

La Tabla 8.1 muestra las funciones núcleo más usadas en estimación no paramétrica de la regresión (la función $I_{[a,b]}$ es la función unidad (valor 1) en el intervalo $[a,b]$). Obsérvese que todos esos núcleos son funciones de densidad unimodales centradas en el 0, aunque también son válidos núcleos que no son funciones de densidad. Es deseable que la función núcleo sea derivable (o al menos continua) porque esta característica es heredada por el estimador de $r(t)$. Por otra parte, los núcleos con soporte compacto (es decir, los que dan peso mayor que 0 únicamente en un intervalo cerrado y acotado) reducen el coste computacional de la estimación.

Obsérvese que usar un núcleo uniforme es equivalente a estimar la regresión localmente usando únicamente los puntos que están en un intervalo centrado en el punto donde se realiza la estimación, todos ellos con idéntico peso. Es decir, el procedimiento que seguimos para construir el estimador de la función de regresión representada en el panel izquierdo de la Figura 8.4 es el mismo que si usamos un núcleo uniforme con ventana $h=5$.

El estimador lineal local se generaliza fácilmente al ajuste local de regresiones *polinómicas* de mayor grado. Una regresión polinómica es en realidad una regresión lineal como la vista en el capítulo anterior, en las que se crean variables con términos cuadráticos, cúbicos, etc., hasta grado q . Es decir, de una variable x obtenemos el polinomio $\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots + \beta_q x^q$ y se actúa como en una regresión lineal con q regresores. En nuestro caso, en vez del valor de x_i , utilizamos el valor $(x_i - t)$.

A partir de aquí, el estimador de $r(t)$ por polinomios locales de grado q se construye como sigue. Primero se asignan los pesos w_i mediante una función núcleo, tal como se hace en el ajuste lineal local. Se plantea entonces el problema de regresión polinómica ponderada

$$\min_{\beta_0, \dots, \beta_q} \sum_{i=1}^n w_i \left(y_i - (\beta_0 + \beta_1(x_i - t) + \dots + \beta_q(x_i - t)^q) \right)^2$$

Obsérvese que los coeficientes obtenidos dependen del punto t donde se realiza la estimación: $\hat{\beta}_j = \hat{\beta}_j(t)$. Finalmente, se da como estimador de $r(t)$ el valor del polinomio $P_{q,t}(x - t)$ estimado localmente en torno a $x=t$:

$$\hat{r}_q(t) = \hat{\beta}_0(t)$$

El hecho de ajustar polinomios de grado mayor que 1 permite que la función estimada se ajuste mejor a los datos.

Además, a partir del polinomio $P_{q,t}(x-t)$ estimado alrededor de t se pueden dar estimadores de las primeras q derivadas de la función r en t . Por ejemplo, la derivada s -ésima de r_q en $x=t$ se estima como

$$\hat{r}_q^{(s)}(t) = \frac{d^s}{dx^s} \left(P_{q,t}(x-t) \right) \Big|_{x=t} = s! \hat{\beta}_s(t)$$

En el caso particular de que se ajuste localmente un polinomio de grado 0 (es decir una constante), se obtiene el conocido como estimador de Nadaraya-Watson o estimador núcleo de la regresión. Su expresión explícita es ésta:

$$\hat{r}_K(t) = \frac{\sum_{i=1}^n K\left(\frac{x_i-t}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x_i-t}{h}\right)} = \sum_{i=1}^n w(t, x_i) y_i$$

Históricamente el estimador de Nadaraya-Watson es anterior a los estimadores por polinomios locales. Obsérvese que $\hat{r}_K(t)$ es una media ponderada de los valores de la variable de respuesta, donde el peso de cada dato depende de la distancia entre el valor de la variable explicativa y el punto t donde se está estimando la función de regresión. Podemos ver el estimador núcleo como una media ponderada móvil. De hecho, puede probarse que todo estimador por polinomios locales puede expresarse como una media ponderada,

$$\hat{r}_q^{(s)}(t) = \sum_{i=1}^n w_q^*(t, x_i) y_i,$$

aunque los pesos $w_q^*(t, x_i)$ no necesariamente han de ser positivos.

Daremos algunas recomendaciones generales para elegir el grado de los polinomios que se ajustan localmente. Cuanto mayor es q , mejores son las propiedades teóricas del estimador no paramétrico, aunque en la práctica no se aconseja que q supere $(s+1)$, donde s es el orden de la derivada de $r(x)$ que se desea estimar. Para estimar la función de regresión, es preferible ajustar polinomios de grado impar a ajustar los del grado par inmediatamente anterior, porque los primeros se adaptan mejor a los datos en la frontera del soporte de la variable explicativa, en el caso de que éste no sea toda la recta real. Por tanto, el estimador lineal local ($q=1$) es preferible al estimador de Nadaraya-Watson ($q=0$). Señalemos finalmente que el efecto que la elección del grado q tiene en el estimador es mucho menor que el debido a la elección del parámetro de suavizado h .

La calidad del ajuste no paramétrico puede medirse usando la siguiente versión del coeficiente de determinación ajustado (véase página 179):

$$R_{ajd}^2 = 1 - \frac{\hat{\sigma}_\varepsilon^2}{\hat{\sigma}_y^2},$$

donde $\hat{\sigma}_y^2 = \sum_{i=1}^n (y_i - \bar{y})^2 / (n-1)$ y $\hat{\sigma}_\varepsilon^2$ se definirá más adelante (página 224).

Referencias específicas sobre estimación no paramétrica basada en polinomios locales o en estimadores núcleo son los libros de [Fan & Gijbels 1996] y [Wand & Jones 1995]. Por su

parte, [Simonoff 1996] y [Bowman & Azzalini 1997] son textos más generales, que abarcan además otros aspectos de la estimación no paramétrica.

8.2.2 Elección del parámetro de suavizado

Como se ha mencionado anteriormente, la elección del parámetro de suavizado h tiene una importancia crucial en el aspecto y propiedades del estimador de la función de regresión. En la práctica, valores distintos de h pueden producir estimadores completamente distintos. La Figura 8.5 muestra tres estimaciones de la función de regresión correspondientes a otros tantos valores del parámetro de suavizado: $h=0,25$ (excesivamente pequeño: el estimador es muy poco suave y tiene muchas irregularidades), $h=2,5$ (es el que se usó en la Figura 8.3; parece una buena elección) y $h=15$ (demasiado grande: se suaviza demasiado y el estimador no paramétrico es casi igual al paramétrico, la recta de regresión).

Tres valores de h : 0.25, 2.5 y 15

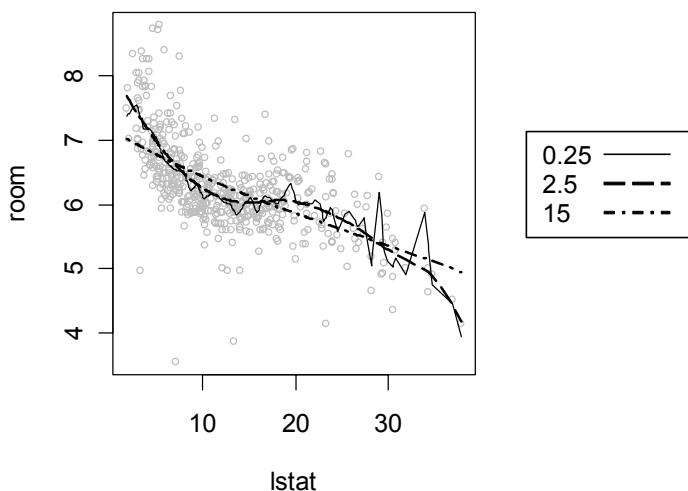


Figura 8.5. Ajuste lineal local con núcleo gaussiano y tres valores de suavizado (h).

El parámetro de suavizado controla el equilibrio que el estimador no paramétrico de la función de regresión debe mantener entre el buen ajuste a los datos observados y la capacidad de predecir bien observaciones futuras. Valores pequeños de h dan mucha flexibilidad al estimador y le permiten acercarse a todos los datos observados (cuando h tiende a 0 el estimador acaba por interpolar los datos), pero los errores de predicción asociados serán altos. Hay, por tanto, sobreajuste (*overfitting*). Por el contrario, si h tiene un tamaño moderado no se ajustará tan bien a las observaciones (tampoco es necesario, dado que los datos pueden contener ruido aleatorio) pero predecirá mejor. En el otro extremo, si h es demasiado grande, tendremos falta de ajuste (*underfitting*), como podía ocurrir con los modelos paramétricos globales.

En estadística se habla del equilibrio entre *sesgo* y *varianza* del estimador. Para h pequeño el estimador es muy variable (aplicado a muestras distintas provenientes del mismo modelo da resultados muy distintos) y tiene poco sesgo (el promedio de los estimadores obtenidos para muestras distintas es aproximadamente la verdadera función de regresión). Si h es grande ocurre lo contrario.

El parámetro de suavizado puede elegirse de forma *manual*: comparando los resultados obtenidos para distintos valores de h y eligiendo aquel que, a juicio del investigador, da el resultado más satisfactorio visualmente, o el más informativo (el que mejor resume la relación existente entre los datos). Esta forma de proceder está sujeta a la opinión subjetiva del usuario y no puede automatizarse, lo que la hace inviable cuando el número de estimaciones no paramétricas que se han de realizar es grande. Se necesitan, pues, métodos automáticos de selección del parámetro de suavizado. Citaremos aquí los más habituales.

Error de predicción en una muestra test

Como se ha visto en capítulos anteriores, si la cantidad de datos disponibles permite dividir éstos en una muestra para la estimación del modelo (conjunto de entrenamiento) y una muestra de prueba, entonces una buena medida de la calidad de un valor h del parámetro de suavizado es el error cuadrático medio de predicción en la muestra test:

$$ECMP_{test}(h) = \frac{1}{n_T} \sum_{i=1}^{n_{test}} (y_i^{test} - \hat{r}_h(x_i^{test}))^2,$$

donde $(x_i^{test}, y_i^{test}), i = 1, \dots, n_{test}$, es la muestra test y \hat{r}_h es el estimador no paramétrico construido con parámetro de suavizado h usando la otra parte de la muestra original. Se elige como parámetro de suavizado el valor h_{test} que minimice esa función.

Validación cruzada

La validación cruzada, además de un criterio de evaluación, como se ve en la Sección 17.2, es una técnica usada en muchos campos para la elección de parámetros que controlan el equilibrio entre precisión y variabilidad (o entre bondad del ajuste y capacidad predictiva) cuando no hay posibilidad de disponer de una muestra de test.

La validación cruzada *n-fold* o total consiste en sacar de la muestra consecutivamente cada una de las observaciones x_i , estimar el modelo con los restantes datos (sea $\hat{r}_{(i)}(t)$ el estimador así obtenido), predecir el dato ausente con ese estimador (así se está haciendo de hecho predicción fuera de la muestra) y, finalmente, comparar esa predicción con el dato real. Si este proceso se hace con cada posible valor de h , podemos construir la siguiente función:

$$ECMP_{CV}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{r}_{(i)}(x_i))^2,$$

que mide el error de predicción del estimador fuera de la muestra para cada h . El valor que minimice esa función, h_{CV} , será el valor del parámetro de suavizado elegido.

La Figura 8.6 muestra el gráfico de la función $ECMP_{CV}(h)$ en el ejemplo que venimos usando (relación entre las variables `room` y `lstat`). Se han usado polinomios de grado 1 con pesos dados por un núcleo gaussiano.

Se observa que tanto los valores de h excesivamente pequeños como los excesivamente grandes dan lugar a errores de predicción fuera de la muestra excesivamente grandes, y que el óptimo se encuentra en un valor intermedio, $h_{CV}=2,12$, que dista poco del valor $h=2,5$ usado en el panel derecho de la Figura 8.4.

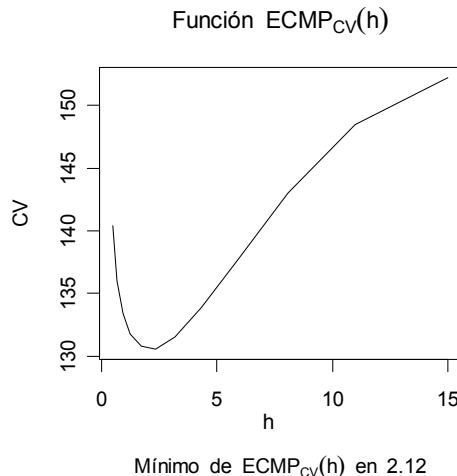


Figura 8.6. Valores del error (validación cruzada) según el parámetro de suavizado h , del problema `roomy lstat`.

Validación cruzada generalizada.

En la Sección 7.2.6 (página 177) vimos que se podía obtener analíticamente una estimación del error prácticamente equivalente a la validación cruzada. Es una modificación de la validación cruzada que puede aplicarse a los estimadores que, como el basado en ajuste de polinomios locales, son lineales en las observaciones de la variable dependiente. En estos casos la predicción de la función de regresión en cada valor observado x_i es

$$\hat{y}_i = \sum_{j=1}^n w^*(x_i, x_j) y_j$$

En forma matricial tenemos que

$$\hat{Y} = SY,$$

donde los vectores columna Y e \hat{Y} tienen elementos y_i e \hat{y}_i , respectivamente, y la matriz S (llamada *matriz de suavizado*) tiene su elemento (i, j) , s_{ij} , igual a $w_p^*(x_i, x_j)$. La matriz de suavizado es análoga a la matriz sombrero H en regresión paramétrica.

Se puede demostrar que

$$ECMP_{CV}(h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - s_{ii}} \right)^2$$

(igual que ocurre en el modelo de regresión lineal múltiple; véase la página 180 del capítulo anterior), con lo que para evaluar esta función no es necesario ajustar n regresiones no paramétricas, sino que basta con ajustar únicamente la que hace intervenir todos los datos y anotar la matriz S . El criterio de validación cruzada generalizada se define sustituyendo en esa fórmula los valores s_{ii} de la diagonal de S por su valor promedio:

$$ECMP_{GCV}(h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - \text{Traza}(S)/n} \right)^2,$$

donde:

$$Traza(S) = \sum_{i=1}^n S_{ii}$$

es la suma de los elementos de la diagonal de S . Tras manipular esta expresión se tiene que

$$ECMP_{GCV}(h) = \frac{n\hat{\sigma}_{\varepsilon}^2}{n - k_h},$$

donde

$$k_h = Traza(S) \text{ y } \hat{\sigma}_{\varepsilon}^2 = \left(\sum_{i=1}^n (y_i - \hat{y}_i)^2 \right) / (n - k_h)$$

es un estimador de la varianza de los errores del modelo. El valor h_{GCV} que minimiza esa función es el parámetro de suavizado elegido.

A k_h se le denomina *número equivalente de parámetros* del estimador no paramétrico correspondiente al valor h del parámetro de suavizado. Ello se debe a que en el modelo de regresión paramétrico con k parámetros, se tiene que $k = Traza(H)$. En general k_h es decreciente en h . Dado que la interpretación de k_h , como número de parámetros equivalentes, es la misma sea cual sea el estimador (siempre que sea lineal en el sentido expresado más arriba) algunos paquetes estadísticos admiten que el grado de suavidad de la estimación no paramétrica se exprese en términos de k_h , en vez de hacerlo en términos de h .

Ventana variable

En ocasiones es preferible usar parámetros de suavizado que dependan del punto t donde se está estimando la función de regresión ($h(t)$) o de los valores observados de la variable explicativa ($h(x_i)$). Éste será el caso cuando la densidad de la variable explicativa varíe considerablemente a lo largo de su recorrido (en zonas con muchos datos la ventana puede ser más pequeña que en zonas donde haya pocas observaciones) o cuando la varianza de los errores sea función de la variable explicativa (en zonas con gran variabilidad en los errores es recomendable usar valores grandes de la ventana). En el ejemplo `lstat` y `room` (Figura 8.1 en adelante) vemos que la densidad es mucho menor con valores altos de `lstat`.

La forma más habitual de incluir una ventana variable en el estimador no paramétrico es fijar la proporción s de puntos que se desea usar en la estimación de cada valor $r(t)$ y definir $h(t)$ tal que el número de datos (x_i, y_i) con x_i perteneciente al intervalo $(t - h(t), t + h(t))$ sea $s \times n$. La proporción s se denomina *span*. Obsérvese que si se ajusta un polinomio de grado $q=0$ (estimador de Nadaraya-Watson), se usa el núcleo uniforme y se elige $s = k/n$, el estimador resultante coincide con el estimador de los k vecinos más cercanos (*k-nearest neighbours*, en inglés), que se tratan en el Capítulo 16. La elección de s (o de $k = s \times n$) puede hacerse mediante validación cruzada o usando una muestra test.

8.2.3 Regresión múltiple. Modelos aditivos

Hasta ahora hemos visto modelos de regresión con una variable de respuesta y una sola variable de entrada (regresor). La extensión del modelo de regresión no paramétrica al caso en el que hay p regresores es directa:

$$y_i = r(x_{i1}, \dots, x_{ip}) + \varepsilon_i, \text{ con } E(\varepsilon_i) = 0 \text{ y } V(\varepsilon_i) = \sigma^2 \text{ para todo } i = 1, \dots, n.$$

Aquí la función de regresión r indica cómo varía y en función de las variables explicativas $x = (x_1, \dots, x_p)$ de dimensión p .

Para definir en este marco los estimadores de la función de regresión mediante polinomios locales, necesitamos, por una parte, definir los pesos w_i de cada observación y , y, por otra, especificar qué variables explicativas se incluyen en cada modelo de regresión local.

La definición de los pesos w_i ha de seguir la misma lógica que en el caso univariante: si se quiere estimar r en el punto $t = (t_1, \dots, t_d)$, los datos $(y_i; x_{i1}, \dots, x_{ip})$ que más peso deben tener son aquéllos con valores de las variables explicativas $x = (x_1, \dots, x_p)$ más cercanos a $t = (t_1, \dots, t_p)$. Hay que tener en cuenta que ahora las distancias entre x y t se deben medir en un espacio de dimensión p , y que hay muchas formas razonables de definir estas distancias.

Una forma sencilla de asignar pesos w_i que da buenos resultados en la práctica es la siguiente:

$$w_i = w(t, x_i) \propto \prod_{j=1}^p K\left(\frac{x_{ij} - t_j}{h_j}\right),$$

donde K es un núcleo univariante, h_j es un parámetro de suavizado adecuado para la j -ésima variable explicativa y el símbolo \propto indica proporcionalidad. Si se toman núcleos gaussianos, esto equivale a asignar pesos alrededor de t usando como núcleo p -dimensional la función de densidad de una normal multivariante con p coordenadas independientes, cada una de ellas con varianza h_j^2 .

La definición de la distancia entre x_j y t será más precisa si se tiene en cuenta cómo es la relación de las variables explicativas entre sí. Para ello, en vez de tomar núcleos multivariantes con coordenadas independientes (es lo que ocurre si tomamos el producto de núcleos univariantes) se toma como núcleo la función de densidad de una variable aleatoria cuya matriz de varianzas y covarianzas sea un múltiplo h de la matriz de varianzas y covarianzas muestral C de los datos (x_{i1}, \dots, x_{ip}) , $i = 1, \dots, n$. Por ejemplo, si se toma un núcleo gaussiano multivariante con estas características se tiene que

$$w_i = w(t, x_i) \propto \frac{1}{h^p} \exp\left\{-\frac{1}{2h}(x_i - t)' C^{-1} (x_i - t)\right\}$$

La definición de los modelos de regresión polinómica ponderada que se ajustan localmente sigue también la lógica de lo expuesto en el caso univariante. Si se desea ajustar polinomios p -variantes de grado q , se deben incluir todos los términos posibles de la forma:

$$\beta_{s_1 \dots s_p} \prod_{j=1}^p (x_{ij} - t_j)^{s_j},$$

cuyo grado,

$$\sum_{j=1}^p s_j$$

sea menor o igual que q . La estimación de la función de regresión será el término independiente del polinomio ajustado alrededor del punto t :

$$\hat{r}(t) = \hat{r}(t_1, \dots, t_p) = \hat{\beta}_{0 \dots 0}$$

Para entender la notación $\beta_{s_1 \dots s_p}$ veamos, por ejemplo, que si hay dos variables explicativas, el polinomio de grado 2 ajustado será:

$$\beta_{00} + \beta_{10}(x_{i1} - t_1) + \beta_{01}(x_{i2} - t_2) + \beta_{11}(x_{i1} - t_1)(x_{i2} - t_2) + \beta_{20}(x_{i1} - t_1)^2 + \beta_{02}(x_{i2} - t_2)^2$$
y la estimación de $r(t)$ en $t=(t_1, t_2)$ será $\hat{\beta}_{00}$, el estimador del término independiente del polinomio.

La Figura 8.7 muestra la regresión no paramétrica bivariante de la variable `room` como función de las variables `lstat` y `age` (esta variable mide en cada barrio de Boston el porcentaje de viviendas construidas antes de 1940). Se ha usado un núcleo producto de dos núcleos gaussianos univariantes. Los valores de los parámetros de suavizado son $h_1=2,5$ en la dimensión `lstat`, y $h_2=10$ en la dimensión `age`. Puede observarse que, para cualquier valor fijo de la variable `lstat`, el número de habitaciones tiene un máximo en un valor alto de la variable `age` (aproximadamente en el 75 por ciento de viviendas anteriores a 1940). Posiblemente lo que ocurre es que las viviendas anteriores a 1940 eran en promedio más grandes que las construidas con posterioridad, y eso hace que el tamaño medio de las viviendas sea mayor en barrios con un porcentaje considerable de casas anteriores a esa fecha. El máximo local que la función de regresión estimada tiene en niveles altos de la primera variable explicativa y valores intermedios de la segunda, indica que la diferencia entre tamaños medios de las viviendas según la antigüedad de las mismas es más acentuada en los barrios pobres (valores altos de `lstat`) que en los ricos.

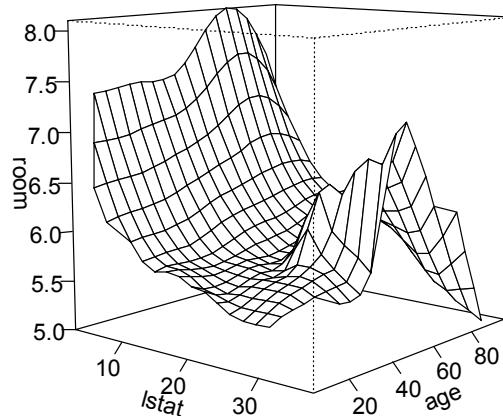


Figura 8.7. Regresión no paramétrica bivariante de la variable `room` en función de las variables `lstat` y `age`.

Tal como hemos expuesto el problema de la regresión múltiple no paramétrica, parece que no tiene características diferenciadas de la regresión simple. Sin embargo, la regresión múltiple plantea un problema específico difícil de solventar. Es el fenómeno conocido como *la maldición de la dimensionalidad* (*curse of dimensionality*, en inglés), que consiste en que en dimensiones altas en los entornos de un punto t casi no hay datos observados (esos entornos están casi vacíos). Como ilustración de ello consideremos una muestra tomada de una variable aleatoria con distribución uniforme en el hipercubo unitario de dimensión p , es decir $[-1, 1]^p$, y la bola de radio 1 centrada en el origen de coordenadas. Para $p=1$ el 100 por cien de los datos muestrales está en la bola, para $p=2$ este porcentaje baja al 79 por ciento, para $p=5$ ya es de tan sólo el 16 por ciento y si $p=10$ únicamente el 0,25 por ciento de los datos observados estarán a una distancia menor que 1 del origen. Es decir, a medida que aumentamos la dimensionalidad, el espacio se “vacía”, disminuyendo rápidamente su densidad. Otra forma de expresar el problema de la dimensionalidad es éste: la bola centrada en t que contiene, digamos, a la cuarta parte de las observaciones muestrales será

tan grande y contendrá puntos tan alejados de t , que no podremos decir que esa bola sea un entorno local de t . Por lo tanto, el comportamiento de la función de regresión en los puntos de esa bola refleja mal cómo es esa función cerca de t y, como consecuencia, los estimadores basados en regresión polinómica local no serán muy fiables.

La única forma de superar la maldición de la dimensionalidad es disponer de muestras de datos de enorme tamaño (esto puede ocurrir, precisamente, en la minería de datos). Si éste no es el caso, hay que ser consciente de que el comportamiento de los estimadores basados en polinomios locales se deteriora al aumentar el número de variables explicativas. Es recomendable no ir más allá de tres o cuatro dimensiones.

Existen métodos alternativos para estudiar la relación funcional entre la variable de respuesta y las variables explicativas a los que afecta menos la maldición de la dimensionalidad. Aquí expondremos únicamente los modelos aditivos. En el Capítulo 7 de [Fan & Gijbels 1996] puede encontrarse información sobre otras posibilidades.

Modelos aditivos

Se plantea un modelo de regresión múltiple no paramétrico menos flexible que el que hemos visto hasta ahora. La pérdida en flexibilidad se ve compensada por el hecho de que el modelo es más fácilmente interpretable y se puede estimar con buenos resultados, incluso con alta dimensionalidad (muchas variables explicativas). El modelo aditivo es éste:

$$y_i = \alpha + \sum_{j=1}^p g_j(x_{ij}) + \varepsilon_i,$$

con $E(\varepsilon_i) = 0$ y $V(\varepsilon_i) = \sigma^2$ para todo $i = 1, \dots, n$, y, además, $E(g_j(x_j)) = 0$ para todo $j = 1, \dots, p$. Las funciones $g_j(x_j)$ tendrán que ser estimadas no paramétricamente, puesto que no se especifica qué forma tienen. La única condición (hipótesis) es que estas funciones se combinan aditivamente para dar lugar a la función conjunta que relaciona la variable de respuesta con las p variables explicativas. En cierto modo el modelo aditivo está a medio camino entre el modelo de regresión lineal múltiple paramétrico (que combina aditivamente transformaciones lineales de las variables, $\beta_j x_{ij}$) y el modelo de regresión múltiple no paramétrico.

Obsérvese que $E(y_i) = \alpha$ (ya que $E(\varepsilon_i) = 0$ y $E(g_j(x_j)) = 0$). Además, si el parámetro α y todas las funciones g_j fuesen conocidas, excepto la función g_k , entonces ésta podría estimarse mediante cualquier estimador no paramétrico univariante (por ejemplo, mediante un ajuste lineal local). Bastaría con aplicar ese estimador al conjunto de datos $(x_i, y_i^{(k)})$, donde:

$$y_i^{(k)} = y_i - \alpha - \sum_{j=1, j \neq k}^p g_j(x_{ij})$$

Esta observación lleva a proponer el algoritmo conocido como *backfitting* (véase Figura 8.8) para estimar el modelo aditivo. En [Hastie & Tibshirani 1990] pueden encontrarse más detalles sobre este algoritmo y, en particular, sobre su convergencia y la unicidad de la solución a la que converge.

Para ejemplificar esta técnica, se ha ajustado un modelo aditivo a los datos de viviendas en los barrios de Boston. Se ha usado la librería `mgcv` del paquete `R` (véase la Sección 8.4 para más detalles sobre este paquete). Utilizaremos como estimador no paramétrico

univariante el suavizado mediante *splines*, que se mencionó brevemente al comienzo de la Sección 8.2, ya que es el implementado en esta librería.

ALGORITMO Backfitting

Estimar α mediante $\hat{\alpha} = (1/n) \sum_{i=1}^n y_i$

Dar como estimaciones iniciales de las funciones g_k funciones cualesquiera $\hat{g}_k = g_k^0$, para $k = 1, \dots, p$ (por ejemplo, $g_k^0(x_{ik}) = \hat{\beta}_k x_{ik}$, donde los coeficientes $\hat{\beta}_k$ son los estimados en el modelo de regresión lineal múltiple).

REPETIR

PARA CADA $k = 1, \dots, p$,

estimar g_k mediante el ajuste no paramétrico univariante de los datos $(x_i, \hat{y}_i^{(k)})$, donde

$$\hat{y}_i^{(k)} = y_i - \hat{\alpha} - \sum_{j=1, j \neq k}^p \hat{g}_j(x_{ij}).$$

FIN PARA

HASTA convergencia.

FIN ALGORITMO

Figura 8.8. Algoritmo de backfitting para estimar un modelo aditivo.

La Figura 8.9 muestra los resultados. Se ha representado la estimación de la función de regresión bivariante de `room` sobre `lstat` y `age` en el panel superior. La comparación de este gráfico con la Figura 8.7 revela que el modelo aditivo no puede recoger el máximo local (situado alrededor de `lstat=35`, `age=50`) que vimos antes. Esto es una muestra de que este modelo es más rígido que el modelo de regresión no paramétrico.

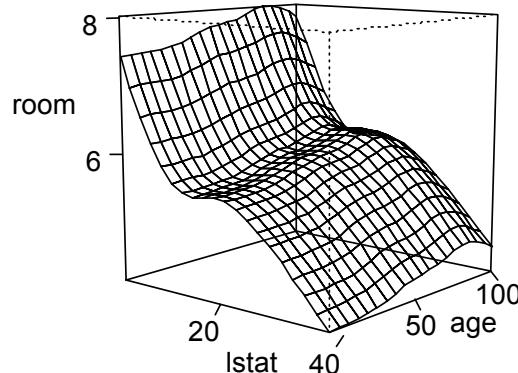


Figura 8.9. Modelo aditivo para el ajuste de `room` como función de `lstat` y `age`. Función de regresión estimada.

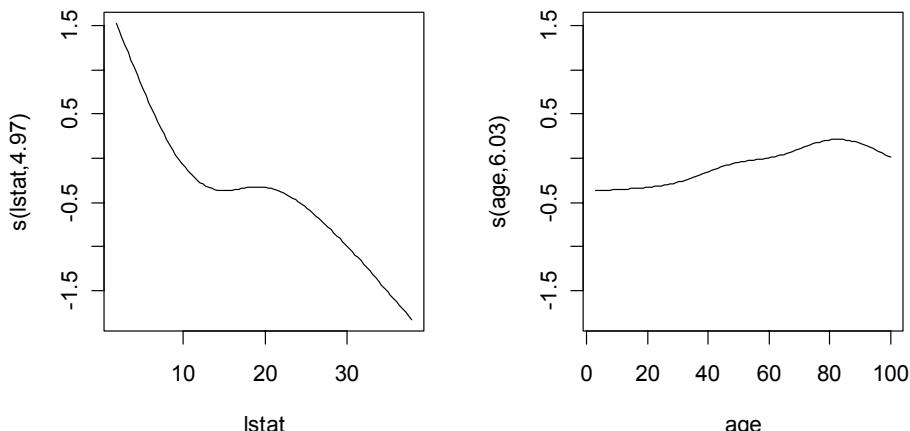


Figura 8.10. Estimaciones no paramétricas de las contribuciones aditivas $g_k(\cdot)$ de cada variable explicativa.

En la Figura 8.10 se han representado las estimaciones no paramétricas de las contribuciones aditivas de cada variable explicativa, $g_{lstat}(\cdot)$ y $g_{age}(\cdot)$. En las etiquetas que acompañan los ejes de ordenadas puede verse el número de parámetros equivalentes de ambas estimaciones. Se puede constatar que el gráfico de $g_{lstat}(\cdot)$ es (si se le suma la media global de $room$) muy similar a la estimación de la función de regresión univariante que se representó en la Figura 8.4 (panel derecho).

Obsérvese que si se dan cortes al gráfico tridimensional, paralelos al plano ($lstat$, $room$), los perfiles que se obtienen son copias de la función $g_{lstat}(\cdot)$. Análogo resultado se obtiene si los cortes son paralelos al plano (age , $room$). De hecho la superficie representada se puede obtener desplazando la función $g_{lstat}(\cdot)$ en el espacio, apoyada sobre la función $g_{age}(\cdot)$ (o viceversa) y sumando la media global de $room$.

8.3 Discriminación no paramétrica

En el capítulo anterior tratamos algunos métodos de discriminación *paramétrica*. En esta sección trataremos el problema de discriminación (clasificación) desde una perspectiva *no paramétrica*. Recordemos el planteamiento que ya se expuso en la Sección 7.8.1 del capítulo anterior. Se observan p características, $x = (x_1, \dots, x_p)$, en n individuos que pertenecen a una población dividida en q subpoblaciones (o clases), $\{C_1, \dots, C_q\}$. De cada individuo también se sabe la clase $y_i \in \{1, \dots, q\}$ a la que pertenece. Así, los datos de que se dispone son

$$(y_i; x_{i1}, \dots, x_{ip}), i = 1, \dots, n$$

El objetivo es buscar una regla discriminante que asigne un nuevo individuo (del que desconocemos su clase) a una de las q clases, a partir de sus valores x_{ij} .

En la Sección 7.8.2 se indicó que la regla óptima es la regla bayesiana, que consiste en asignar el individuo con observaciones x a la clase j que tiene máxima probabilidad a posteriori:

$$f(x | C_j)P(C_j) = \underset{k=1 \dots q}{\text{Max}} f(x | C_k)P(C_k)$$

Esta regla sólo es aplicable si se conocen las probabilidades a priori de cada clase, $P(C_k)$, y las funciones de densidad $f(x | C_k)$ del conjunto de variables x para los individuos de cada clase. Las probabilidades a priori pueden estimarse fácilmente como las frecuencias relativas observadas de cada clase: $P(C_k)$. La estimación de las funciones de densidad $f(x | C_k)$, más compleja, puede llevarse a cabo mediante distintas técnicas. En el capítulo anterior vimos las paramétricas y, en éste, las no paramétricas.

El estimador no paramétrico de la función de densidad más usado es el estimador núcleo (exceptuando quizás el histograma). Este estimador distribuye el peso $1/n$ de cada dato observado en un entorno suyo de forma continua, tal como se ilustra en la Figura 8.11, donde hay cinco observaciones, marcadas en la parte inferior.

La forma en que se realiza ese reparto viene determinada por una función núcleo K (ya se habló de este tipo de funciones en la Sección 8.2 al presentar los estimadores núcleo de la función de regresión). En general K es a su vez una función de densidad continua, unimodal y simétrica alrededor del 0. En el caso de que x sea una sola variable explicativa (univariante, dimensión 1), la expresión de este estimador es

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right),$$

donde h es el parámetro de suavizado. Por lo tanto, la estimación de la densidad es una mixtura de n densidades con la misma forma que el núcleo K , reescaladas según el parámetro h , y centradas cada una en una observación x_i , como se ve en la Figura 8.11. La densidad estimada es la “suma” de estos núcleos (desplazados y reescalados convenientemente).

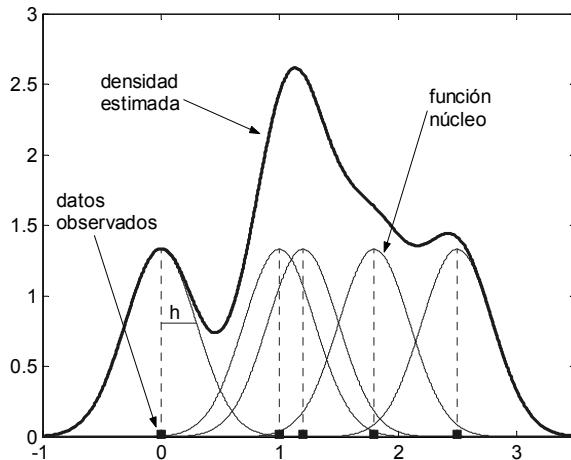


Figura 8.11. Estimación de la función de densidad a partir de cinco observaciones mediante un núcleo gaussiano.

La generalización al caso de x p -dimensional sigue las mismas pautas que en el caso de la regresión no paramétrica. Por ejemplo, si se usa un núcleo producto, el estimador núcleo de la densidad es de la forma

$$\hat{f}(x) = \frac{1}{n \prod_{j=1}^p h_j} \sum_{i=1}^n \prod_{j=1}^p K\left(\frac{x_{ij} - x_j}{h_j}\right)$$

La elección del parámetro de suavizado h es, también aquí, crucial. Existen versiones de los métodos de validación cruzada aplicables en el contexto de la estimación no paramétrica de la función de densidad. No obstante, el método de elección de h que da mejores resultados prácticos es el conocido como *plug-in*. Se basa en la determinación teórica del mejor valor de h , suponiendo que la función de densidad f verdadera es conocida. Así, h depende de características de f . En la práctica, estas cantidades se estiman a partir de los datos y se sustituyen en la expresión teórica de h para conseguir el estimador *plug-in* de h . Estos y otros métodos de elección de h pueden consultarse en [Wand & Jones 1995].

Como hemos dicho al principio, en análisis discriminante se necesitan estimaciones de $f(x | C_k)$, para $k=1, \dots, q$. Así que es necesario estimar no paramétricamente la densidad de x en cada una de las q clases en las que está dividida la población. Cada una de estas estimaciones usa exclusivamente las observaciones que pertenecen a la clase cuya densidad se está estimando. Finalmente, la regla discriminante consistirá en asignar el individuo con observaciones x a la clase j que tiene máxima probabilidad a posteriori estimada:

$$\arg \max_{k=1 \dots q} \hat{f}(x | C_k) \hat{P}(C_k)$$

Aplicaremos la técnica descrita al ejemplo que venimos usando en este capítulo: los datos sobre viviendas en los barrios de Boston. Dividimos los datos en dos clases, C_1 y C_2 , según si la variable `room` es menor o mayor que su mediana (que vale 6,2), respectivamente. Así las probabilidades a priori de cada clase serán iguales a 0,5. Como variables predictivas x tomaremos únicamente la variable `lstat` (univariante). Se estima la densidad de `lstat` en cada clase usando un núcleo gaussiano y ventanas elegidas por el método *plug-in*. La Figura 8.12 muestra las estimaciones obtenidas.

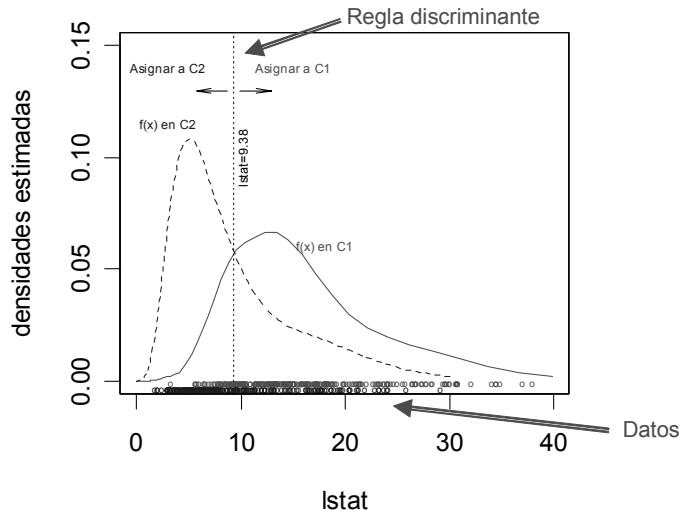


Figura 8.12. Estimaciones de las funciones de densidad de `lstat` en las clases C_1 y C_2 según la variable `room`.

Se puede observar que la estimación de la densidad correspondiente a la clase C_1 es mayor que la correspondiente a C_2 si y sólo si `lstat` es mayor que 9,38. Por lo tanto la regla discriminante no paramétrica asignará a C_1 todas las observaciones con valores de `lstat` mayores que 9,38 y asignará las restantes a C_2 .

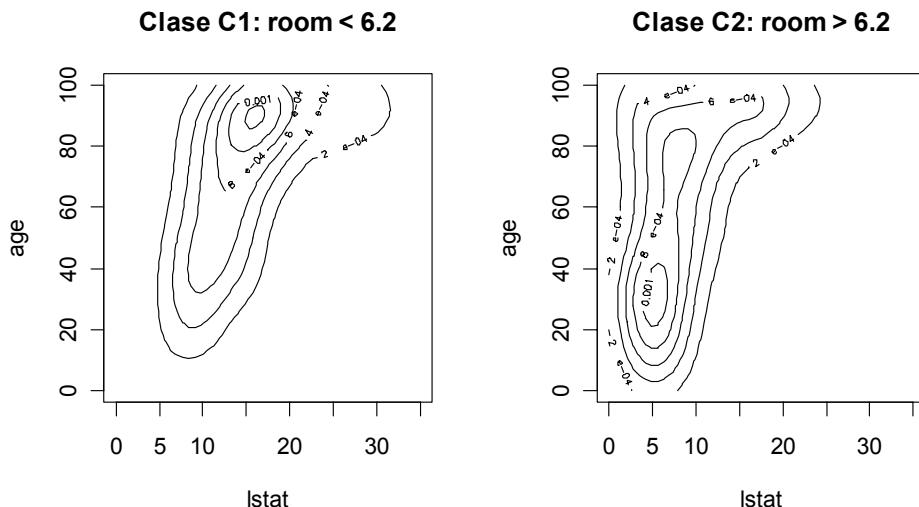


Figura 8.13. Estimación de la densidad de $(lstat, age)$ en las clases C_1 y C_2 .

En el ejemplo anterior, al sólo existir una sola variable explicativa, no se ve claramente la flexibilidad de la discriminación no paramétrica. Para ilustrar esta flexibilidad incluimos

age como variable explicativa adicional. Ahora son bivariantes las densidades que se han de estimar. La Figura 8.13 muestra la estimación de la densidad conjunta de (lstat, age) en cada una de las dos clases en las que se ha dividido la muestra. Se aprecia que la clase C_1 se concentra en valores relativamente altos de ambas variables (la moda está cerca del punto ($lstat=15$, $age=90$)), mientras que C_2 lo hace en valores bajos (la moda se encuentra en torno a ($lstat=5$, $age=30$)).

Para obtener la regla discriminante se toma la diferencia de la densidad estimada en la clase C_2 menos la estimada en C_1 , y se clasifican en C_2 las observaciones para las que esta diferencia sea positiva. En la Figura 8.14 se ha representado esta diferencia y se ha señalado en trazo grueso la frontera entre las zonas que se clasificarán en una u otra clase, que es donde la diferencia entre las densidades estimadas es igual a 0. Se han marcado con un círculo los puntos de C_2 y con una cruz los de C_1 . Se ve claramente que los discriminadores obtenidos mediante estimación no paramétrica de la densidad pueden realizar clasificaciones no lineales.

Densidad en C_2 menos densidad en C_1

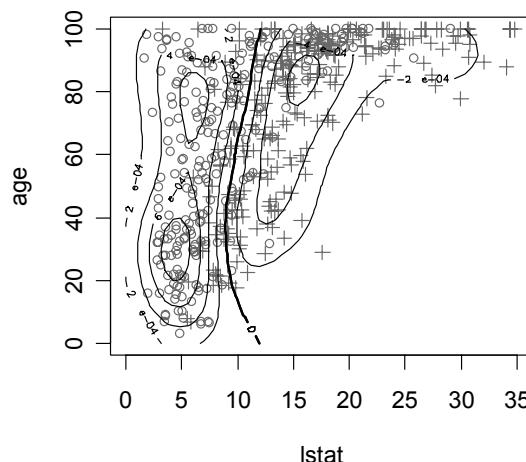


Figura 8.14. Diferencia de las densidades estimadas en las clases C_1 y C_2 .

La estimación no paramétrica de la función de densidad no está libre de la maldición de la dimensionalidad, que ya discutimos en el caso de la regresión no paramétrica. Su efecto es menor sobre las reglas discriminantes derivadas de los estimadores de las densidades (porque para discriminar no es necesario estimar bien las funciones de densidad completas, sino sólo los valores relativos de las densidades en las distintas subpoblaciones) pero aun así, no es recomendable usar el método de discriminación descrito si la dimensionalidad p es grande (digamos mayor o igual que 4). Una forma de solventar este problema es usar un estimador de la función de densidad construido bajo la hipótesis (poco verosímil en la mayor parte de los casos) de que las p componentes de las variables x son independientes. Así basta estimar no paramétricamente la densidad de cada variable explicativa y multiplicar éstas para obtener el estimador de la densidad conjunta. La regla discriminante obtenida a partir de ese estimador se conoce como regla *Bayes Naïve* y en la práctica da buenos resultados. Este clasificador Bayes se trata en detalle en el Capítulo 10 (se puede consultar también [Hastie et al. 2001], Sección 6.6.3). Obsérvese la analogía existente entre

el paso del modelo de regresión múltiple no paramétrico al modelo aditivo, y el paso de la estimación de la densidad multivariante al estimador basado en independencia de las marginales.

Existen otras formas de paliar la maldición de la dimensionalidad basadas en la reducción de la dimensión de la variable clasificadora x . Son técnicas que proyectan los datos en espacios de baja dimensión y buscan la proyección que permita distinguir mejor las clases en las que se dividen las observaciones (véase [Cook & Yin 2001; Hernández & Velilla 2001] y [Zhu & Hastie 2003]).

Regresión binaria local

La discriminación no paramétrica, basada en la estimación de las funciones de densidad en cada subpoblación, no es la única vía de plantearse el problema de clasificación desde una óptica no paramétrica. Este problema puede también modelarse como uno de regresión no paramétrica en el que la respuesta (el indicador de la clase a la que pertenece cada dato) es categórica. Aquí nos limitaremos a estudiar el caso en el que sólo hay dos clases (respuesta binaria). Estos modelos son la versión no paramétrica de los modelos lineales generalizados que se introdujeron en la Sección 7.7.

Consideremos el problema de análisis discriminante con dos clases, C_0 y C_1 , y observaciones asociadas

$$(y_i; x_{i1}, \dots, x_{ip}), i = 1, \dots, n,$$

con y_i igual a 0 o a 1, según si la observación i -ésima pertenece a una u otra clase. Modelizamos y_i como una variable aleatoria que toma los valores 1 y 0 con probabilidades respectivas p_i y $1-p_i$, donde p_i es función de las variables explicativas (x_{i1}, \dots, x_{ip}) :

$$E(y_i) = P(y_i = 1) = p_i = r(x_{i1}, \dots, x_{ip})$$

La distribución de probabilidad queda entonces totalmente determinada: y_i sigue una Bernoulli de parámetro $p_i = r(x_{i1}, \dots, x_{ip})$. Si la función $r(x_{i1}, \dots, x_{ip})$ fuese conocida, tendríamos una forma sencilla de clasificar una nueva observación de la que sólo conociésemos las variables explicativas $x = (x_1, \dots, x_p)$: se clasificaría en la población C_1 si y sólo si $p_x = r(x_1, \dots, x_p) > 0,5$. Dado que en general la función r no será conocida, lo que se propone es sustituirla en esa regla de clasificación por una estimación suya hecha de forma no paramétrica.

La estimación de $p_x = r(x_1, \dots, x_p)$ sería fácil si un modelo paramétrico (digamos el modelo logístico, visto en la página 196) se adaptase bien a todo el rango de valores de las variables explicativas. Esto no siempre es así y por eso precisamente buscamos un estimador no paramétrico. No obstante, aunque globalmente el modelo logístico no sea una buena aproximación de la función r , sí lo puede ser localmente, en un entorno del punto $x = (x_1, \dots, x_p)$. En realidad, es el mismo tipo de aproximación que hacíamos al estimar localmente mediante un polinomio la función de regresión no paramétrica.

Por lo tanto, suponemos que si $x_i = (x_{i1}, \dots, x_{ip})$ está en un entorno de $x = (x_1, \dots, x_p)$ entonces y_i sigue un modelo logístico:

$$p_i = \frac{1}{1 + e^{-\beta' x_i}}, \text{ o de forma equivalente } \log\left(\frac{p_i}{1 - p_i}\right) = \beta' x_i$$

Obsérvese que el vector de parámetros β es función del punto $x = (x_1, \dots, x_p)$, porque ese punto es el que define qué observaciones están en su entorno y cuáles no. Resulta pues que en un entorno de $x = (x_1, \dots, x_p)$ ajustamos un modelo paramétrico que podemos estimar, por ejemplo, por máxima verosimilitud. La contribución de cada observación a la función de log-verosimilitud es, por tratarse de un modelo logístico,

$$y_i \log\left(\frac{p_i}{1-p_i}\right) + \log(1-p_i).$$

Sumando sobre todas las observaciones y ponderando cada una por un peso $w_i = w(x, x_i)$ (decreciente en la distancia que separa x_i de x) se obtiene la llamada *función de log-verosimilitud local*:

$$l_x(\beta) = \sum_{i=1}^n w_i \left(y_i \log\left(\frac{p_i}{1-p_i}\right) + \log(1-p_i) \right)$$

Maximizando esta función se obtiene un estimador de β , $\hat{\beta} = \hat{\beta}(x)$, que permite obtener una estimación de $p_x = r(x_1, \dots, x_p)$:

$$\hat{r}(x_1, \dots, x_p) = \hat{p}_x = \frac{1}{1 + e^{-\hat{\beta}'x}}$$

Según si este valor es menor o mayor que 0,5, se clasificará la observación $x = (x_1, \dots, x_p)$ en C_0 o en C_1 , respectivamente.

En la práctica los pesos $w_i = w(x, x_i)$ se definen a partir de funciones núcleo en dimensión p , del mismo modo que se hace en regresión múltiple no paramétrica. El modelo logístico, elegido como aproximación paramétrica local, puede cambiarse por otro modelo paramétrico de respuesta binaria. La variación en los resultados es poco perceptible, porque la función de verosimilitud local usa únicamente pequeñas partes del modelo paramétrico elegido, y dos modelos paramétricos distintos pueden tener partes pequeñas semejantes, aunque globalmente sean distintos.

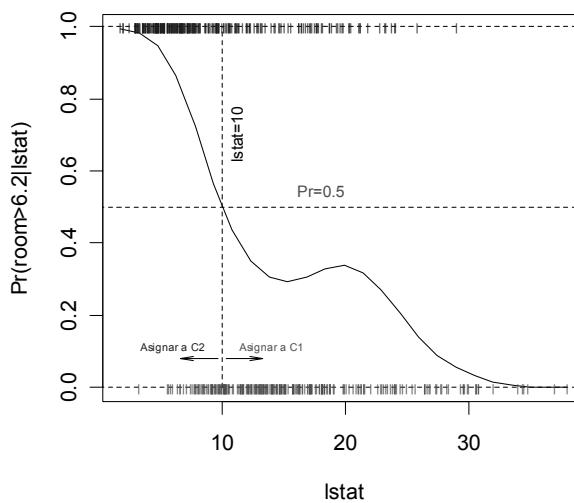


Figura 8.15. Regla discriminante basada en regresión logística local.

La Figura 8.15 muestra la puesta en práctica de la regresión binaria no paramétrica mediante el ajuste local del modelo logístico. En el conjunto de datos sobre características

de las viviendas en barrios de Boston, se desea recuperar la variable binaria creada a partir de `room` ($y_i=0$ si `room` < 6,2, $y_i=1$ en caso contrario) como función de la variable `lstat`. En cada ajuste local se han definido los pesos de cada observación según un núcleo gaussiano con ventana (desviación típica) igual a 3. El punto en el que la función de probabilidad estimada cruza el valor 0,5 es $lstat = 10$. Por lo tanto, la regla de regresión logística no paramétrica predice $y = 0$ cuando el valor observado de `lstat` es mayor que 10, y predice $y = 1$ en caso contrario. Este resultado es muy similar al obtenido con la regla discriminante vista en la sección anterior (9,38).

Completamos el ejemplo incluyendo una segunda variable explicativa (`age`) en el modelo de regresión logística local. La Figura 8.16 muestra las curvas de nivel de la estimación no paramétrica de la probabilidad de pertenecer a la clase C_2 en función de $(lstat, age)$. La línea de trazo grueso está formada por aquellos puntos en los que la estimación de esta probabilidad es igual a 0,5. Por tanto esta línea es la frontera de las zonas que se clasificarán en C_1 (a la izquierda de la línea) o en C_2 (a la derecha). Si se compara esta figura con la Figura 8.14 se aprecian grandes similitudes, aunque no hay una coincidencia total.

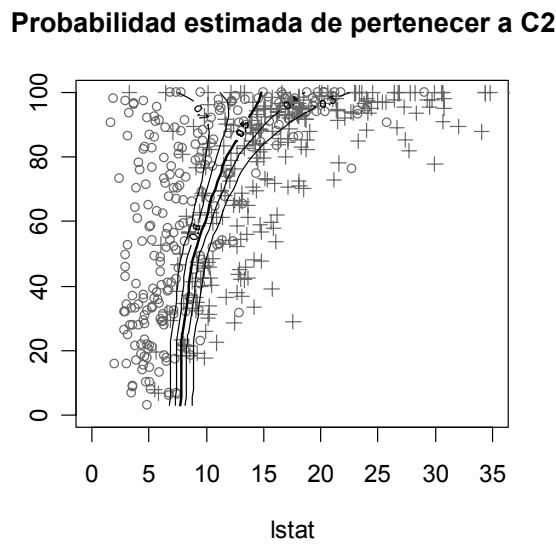


Figura 8.16. Regla discriminante basada en regresión logística local bivariante.

La estimación no paramétrica mediante verosimilitud local que acabamos de exponer también puede sufrir los efectos de la maldición de la dimensionalidad si hay muchas variables explicativas en x (dimensión alta). Los *modelos aditivos generalizados* (véase [Hastie & Tibshirani 1990] o el Capítulo 9 de [Hastie et al. 2001]) consiguen evitar el problema de forma análoga a como los modelos aditivos lo eluden en el modelo de regresión múltiple no paramétrico: se pierde flexibilidad del modelo para ganar en capacidad de estimación y de interpretación. Los modelos aditivos generalizados extienden los modelos aditivos al caso en el que la variable respuesta no es continua o, en caso de serlo, ésta no sigue una distribución normal (dado el valor de la variable explicativa). Es el mismo tipo de extensión que permite pasar del modelo de regresión lineal múltiple al modelo lineal generalizado.

8.4 Conclusiones, aplicabilidad y sistemas

Las técnicas de modelización no paramétrica, tal como han sido presentadas en este capítulo, implican un gran coste computacional si se trabaja con un gran número de datos. Existen formas de reducir considerablemente ese coste. Las técnicas más empleadas son la transformada rápida de Fourier, la discretización de los datos (*binning*) y la combinación de ambas. Véase la Sección 3.6 de [Fan & Gijbels 1996].

No obstante, aun sin estas optimizaciones, en muchos paquetes de estadística y en algunas suites de minería de datos se pueden encontrar estas técnicas para analizar volúmenes de datos que, si no tienen muy alta dimensionalidad, pueden tratarse usando técnicas de muestreo apropiadas.

Todos los cálculos y gráficos de este capítulo han sido realizados con el paquete R (véase [The R Development Core Team 2003]). R es un entorno de trabajo que admite un lenguaje de programación orientado a objetos. Está diseñado para la modelización estadística. Se distribuye bajo licencia GNU. Hay una gran cantidad de librerías que permiten realizar una gama amplísima de análisis estadísticos (usualmente son los autores de un método nuevo los que escriben la librería correspondiente en R). En este capítulo se han usado las librerías base, *locfit*, *KernSmooth*, *mgcv*, *modreg* y *sm*. El paquete comercial S-PLUS tiene características similares a R.

Por último, en este capítulo y el anterior, hemos tratado el problema de la regresión y la clasificación (discriminación). Existen técnicas paramétricas y no paramétricas para agrupamiento (*clustering*). En particular, muchas de ellas se basan en los mismos conceptos que los vistos para los discriminantes en estos dos capítulos.

Capítulo 9

REGLAS DE ASOCIACIÓN Y DEPENDENCIA

Este capítulo se centra en las técnicas *específicas* para el aprendizaje de reglas de asociación y dependencias. Estas reglas expresan patrones de comportamiento entre los datos en función de la aparición conjunta de valores de dos o más atributos. La característica principal de estas reglas es que tratan con atributos nominales, a diferencia de los estudios correlacionales que lo hacen con los numéricos. En concreto, estas reglas expresan las combinaciones de valores de los atributos (items) que suceden más frecuentemente. De hecho, este capítulo no se caracteriza exactamente por recoger todas las técnicas de la tarea de reglas de asociación, ya que en otros capítulos veremos otras técnicas, como las relacionales o evolutivas, que también pueden utilizarse para extraer reglas de asociación. El hilo integrador de este capítulo es que las técnicas que presentamos se basan en buscar conjuntos de items frecuentes, generalmente mediante técnicas de cobertura mínima.

Las reglas de asociación tienen importantes aplicaciones prácticas: análisis de la cesta de compra de un supermercado, estudio de textos, búsqueda de patrones en páginas web, etc. Estas aplicaciones normalmente llevan asociadas gran volumen de datos, por lo que la eficiencia es un factor clave en el aprendizaje de reglas de asociación. Las dependencias funcionales establecen relaciones entre varios atributos discretos, para todos sus valores. En este capítulo tratamos los aspectos más relevantes de las reglas de asociación y dependencias: conceptos, algoritmos y aplicaciones.

9.1 Introducción

Las reglas de asociación son una manera muy popular de expresar patrones de datos de una base de datos. Estos patrones pueden servir para conocer el comportamiento general del problema que genera la base de datos, y de esta manera, se tenga más información que

pueda asistir en la toma de decisiones. Por ejemplo, en un supermercado podemos conocer qué productos suelen comprarse conjuntamente, y así mejorar la distribución de los productos en estanterías. En un servidor web podemos conocer cuáles son los itinerarios más seguidos por los visitantes a las páginas web, y entonces, utilizar esta información para estructurar las páginas web en el servidor.

Las reglas de asociación surgieron inicialmente para afrontar el análisis de las cestas de la compra de los comercios. En este contexto, las diferentes cestas de la compra se pueden expresar formando una base de datos de una sola tabla. Las filas de esta tabla se refieren a una cesta de un supermercado, mientras que las columnas son cada uno de los productos en venta en el supermercado. La tabla sólo contiene valores binarios. Un 1 en la posición (i,j) indica que la cesta i incorpora el producto j , mientras que un 0 indica que el cliente no ha adquirido el producto. La siguiente tabla podría ser un ejemplo de una base de datos de este tipo.

	Vino “El cabezón”	Gaseosa “Chispa”	Vino “Tío Paco”	Horchata “Xufer”	Bizcochos “Goloso”	Galletas “Trigo”	Chocolate “La vaca”
T1	1	1	0	0	0	1	0
T2	0	1	1	0	0	0	0
T3	0	0	0	1	1	1	0
T4	1	1	0	1	1	1	1
T5	0	0	0	0	0	1	0
T6	1	0	0	0	0	1	1
T7	0	1	1	1	1	0	0
T8	0	0	0	1	1	1	1
T9	1	1	0	0	1	0	1
T10	0	1	0	0	1	0	0

Figura 9.1. Tabla de la cesta de la compra.

Una regla de asociación es una proposición probabilística sobre la ocurrencia de ciertos estados en una base de datos. A diferencia de las reglas de clasificación, en las reglas de asociación, en la parte derecha puede aparecer cualquier atributo, y además puede aparecer más de un atributo. Una típica regla de asociación sería:

SI bizcochos “Goloso” **Y** horchata “Xufer” **ENTONCES** galletas “Trigo”

Formalmente, $I = \{i_1, i_2, \dots, i_m\}$, es un conjunto de literales. Para la tabla de la Figura 9.1 $I = \{ \text{vino “El cabezón”}, \text{gaseosa “Chispa”}, \dots, \text{chocolate “La vaca”} \}$. X es un conjunto de ítems de I , si es un subconjunto de I . Por ejemplo, $\{ \text{vino “El cabezón”}, \text{chocolate “La vaca”} \}$ es un conjunto de ítems de tamaño 2 para la tabla de la Figura 9.1.

Por lo tanto, una regla de asociación puede ser vista como reglas de la forma **SI** α **ENTONCES** β , donde α y β son dos conjuntos de ítems disjuntos. Otra forma muy utilizada de expresar una regla de asociación es $\beta \Leftarrow \alpha$, o también $\alpha \Rightarrow \beta$. El conjunto α recibe el nombre de predecesor de la regla, y a β se le denomina sucesor o consecuente.

Dada una regla de asociación, se suele trabajar con dos medidas para conocer la calidad de la regla: **cobertura** (*support*) y **confianza** (*confidence*). La cobertura (también denominada soporte) de una regla se define como el número de instancias que la regla predice correctamente (también se utiliza el porcentaje). Por otra parte, la confianza (también llamada precisión) mide el porcentaje de veces que la regla se cumple cuando se puede

aplicar. Por ejemplo, la regla vista anteriormente extraída de la Figura 9.1 tiene una cobertura igual a tres y una precisión del 75 por ciento (se aplica correctamente tres veces de las cuatro veces que se puede aplicar).

Por desgracia, y en contraste con otras técnicas de minerías de datos, el área de estudio de las reglas de asociación no presenta una homogeneidad en los conceptos. De hecho, el área general suele recibir el nombre de “minería de conjuntos de ítems frecuentes”. Es frecuente, encontrar bibliografía que limita el aprendizaje de las reglas de asociación a casos parecidos al problema de la cesta de la compra. Sin embargo, como veremos, este no es más que un caso particular del área del aprendizaje de reglas de asociación. Existen, como veremos, y como se comentó brevemente en el Capítulo 6, gran variedad de tipos de reglas de asociación y de dependencias. Extendiendo la clasificación de [Han & Kamber 2001] presentamos una clasificación de familias de reglas de asociación basadas en los siguientes criterios:

- **Tipos de los valores utilizados en las reglas:** podemos tener reglas que trabajan con atributos binarios que indican la presencia o ausencia de un ítem, este el caso típico de la cesta de la compra. También hay sistemas que trabajan con atributos con más de dos valores, por ejemplo, **SI** país=Alemania **ENTONCES...** O bien, reglas que contemplan atributos numéricos, por ejemplo, **SI** 18<edad<20 **ENTONCES...**
- **Dimensiones de los datos:** la regla **SI** *Comprar(vino “El cabezón”)* **ENTONCES** *Comprar(gaseosa “Chispa”)*, se refiere a tan sólo una dimensión, que corresponde al hecho de aparecer o no en la cesta de la compra. Podemos incrementar las dimensiones de una regla incluyendo por ejemplo, la dimensión tiempo, o la dimensión cliente. Un ejemplo de regla multi-dimensional sería **SI** *Comprar(vino “El cabezón”), Cliente(Juan), Tiempo(Marzo)* **ENTONCES** *Comprar(gaseosa “Chispa”)*.
- **Niveles de abstracción:** algunos sistemas o algoritmos permiten incorporar a las reglas diferentes niveles de abstracción representados por conceptos que aglutinan otros conceptos o ítems. Este tipo de reglas se conocen como reglas multi-nivel. Un ejemplo sería: **SI** *Comprar(vino)* **ENTONCES** *Comprar(gaseosa)*.
- **Instantáneas o secuenciales:** depende de si se consideran relaciones en un instante de tiempo (por ejemplo una determinada compra) o en una secuencia o serie (varias compras o visitas a una página web).

Las reglas de asociación y dependencia se caracterizan precisamente por el hecho de que se expresan en forma de reglas de tipo “**SI ... ENTONCES ...**” y que los algoritmos tienen como objetivo primordial la eficiencia. Es interesante comparar las reglas de asociación con los métodos bayesianos, y en especial las redes bayesianas (véase el Capítulo 10), que también establecen dependencias (en este caso probabilísticas y estructurales) entre atributos nominales.

9.2 Reglas de asociación

Normalmente, el aprendizaje de reglas de asociación se basa en su confianza y cobertura. Los algoritmos de aprendizaje trabajan en la búsqueda de reglas que cumplan unos requisitos mínimos en estas medidas. Dado el gran volumen de datos de los problemas con el que los algoritmos de aprendizaje de reglas de asociación trabajan, la tarea de buscar

patrones que cumplan estos requisitos puede parecer muy costosa, ya que los conjuntos de ítems a ser analizados crece exponencialmente con respecto al número de variables de los datos. Sin embargo, por fortuna, en los casos reales existen realmente pocos conjuntos frecuentes y los métodos que exigen una confianza y/o cobertura mínimas se benefician de este hecho. Por ejemplo, la mayoría de los clientes de un supermercado suelen comprar un número bastante limitado de productos.

Un algoritmo de aprendizaje de reglas de asociación muy simple y popular es el algoritmo *Apriori*. El funcionamiento de este algoritmo se basa en la búsqueda de los conjuntos de ítems con determinada cobertura. Para ello, en primer lugar se construyen simplemente los conjuntos formados por sólo un ítem que superan la cobertura mínima. Este conjunto de conjuntos se utiliza para construir el conjunto de conjuntos de dos ítems, y así sucesivamente hasta que se llegue a un tamaño en el cual no existan conjuntos de ítems con la cobertura requerida. Tomemos la tabla de la Figura 9.1 y definamos la cobertura mínima igual a dos. En esta tabla existen siete conjuntos de sólo un ítem (siete atributos), y todos aparecen en la tabla al menos dos veces, por lo tanto tomaremos los siete. Si pasamos a los conjuntos formados por dos ítems, de los $C_2^7 = 7!/5! = 42$ posibles casos tendríamos 15 conjuntos. Por ejemplo, el ítem *bizcochos “Goloso”* Y *horchata “Xufer”*, tiene cobertura cuatro. Continuando, tendremos 11 conjuntos de tres ítems, y tan sólo dos conjuntos de cuatro ítems.

Una vez se han seleccionado los conjuntos de ítems que cumplen con la cobertura mínima, el siguiente paso consiste en extraer de estos conjuntos de reglas las que tengan un nivel de confianza mínimo. Por ejemplo del conjunto de ítems *horchata “Xufer”* Y *bizcochos “Goloso”* Y *galletas “Trigo”* podemos extraer las siguientes reglas de asociación.

SI <i>bizcochos “Goloso”</i> Y <i>horchata “Xufer”</i> ENTONCES <i>galletas “Trigo”</i>	$C_b=4$, $C_f=3/4$
SI <i>bizcochos “Goloso”</i> Y <i>galletas “Trigo”</i> ENTONCES <i>horchata “Xufer”</i>	$C_b=3$, $C_f=3/3$
SI <i>galletas “Trigo”</i> Y <i>horchata “Xufer”</i> ENTONCES <i>bizcochos “Goloso”</i>	$C_b=3$, $C_f=3/3$
SI <i>galletas “Trigo”</i> ENTONCES <i>bizcochos “Goloso”</i> Y <i>horchata “Xufer”</i>	$C_b=6$, $C_f=3/6$
SI <i>bizcochos “Goloso”</i> ENTONCES <i>horchata “Xufer”</i> Y <i>galletas “Trigo”</i>	$C_b=6$, $C_f=3/6$
SI <i>horchata “Xufer”</i> ENTONCES <i>bizcochos “Goloso”</i> Y <i>galletas “Trigo”</i>	$C_b=4$, $C_f=3/6$
SI \emptyset ENTONCES <i>bizcochos “Goloso”</i> Y <i>galletas “Trigo”</i> Y <i>horchata “Xufer”</i>	$C_b=10$, $C_f=3/10$

Para cada regla hemos incluido su cobertura y confianza de acuerdo con la Figura 9.1. Si fijásemos la confianza mínima en el 100 por 100, sólo seleccionaríamos la segunda y la tercera regla.

Como hemos visto, el aprendizaje de reglas de asociación se divide normalmente en dos fases: la extracción de los conjuntos de ítems que cumplen con la cobertura requerida desde los datos, y la generación de las reglas a partir de estos conjuntos.

Para la búsqueda de los conjuntos de ítems se emplea una propiedad que permite acelerar de manera considerable su búsqueda: un conjunto ítems formado por X ítems es frecuente si y sólo si cada uno de los X ítems es frecuente por sí solo. Esto permite generar los conjuntos de ítems frecuentes de una manera incremental. La Figura 9.2 muestra el algoritmo.

La selección de candidatos de este algoritmo consiste en formar, dado un conjunto de items de tamaño i , los posibles candidatos de tamaño $i+1$. Si se dispone de un grupo Z formado por conjuntos de items de tamaño i , para la selección de candidatos, se toma cada pareja $\{x,y\}$ que formen un conjunto de items de tamaño $i+1$, donde x y y son conjuntos de items de Z .

```

ALGORITMO Apriori( $D$ : datos,  $MinC$ : cobertura mínima)
   $i=0$ 
  Rellena_Item( $C_i$ ) //Incluye en  $C_0$  todos los ítems de tamaño 1
  MIENTRAS  $C_i \neq \emptyset$ 
    PARA CADA  $x=\text{elemento de } C_i$ 
      SI Cobertura( $x$ )  $\geq MinC$  ENTONCES  $L_i = L_i \cup x$ 
    FINPARA
     $C_{i+1}=\text{Selecciona_Candidatos }(L_i)$ 
     $i=i+1;$ 
  FIN MIENTRAS
  RETORNA  $C$ 
FIN ALGORITMO

```

Figura 9.2. Algoritmo de búsqueda de conjuntos de items (Apriori).

La siguiente fase consiste en la creación de reglas a partir de los conjuntos de items frecuentes. Si sólo buscamos reglas de asociación con un ítem en la parte derecha el proceso es sencillo: de un conjunto de items de tamaño i , se crean i reglas colocando siempre un único ítem diferente en la parte derecha. Por ejemplo, si tenemos el conjunto de items horchata "Xufer" **Y** bizcochos "Goloso" **Y** galletas "Trigo" se construyen las reglas:

SI bizcochos "Goloso" Y horchata "Xufer" ENTONCES galletas "Trigo"	$C_b=4, C_f=3/4$
SI bizcochos "Goloso" Y galletas "Trigo" ENTONCES horchata "Xufer"	$C_b=3, C_f=3/3$
SI galletas "Trigo" Y horchata "Xufer" ENTONCES bizcochos "Goloso"	$C_b=3, C_f=3/3$

El cálculo de la confianza de las reglas se puede hacer de manera eficiente, dividiendo la cobertura del conjunto items por la cobertura de la parte derecha. Este último dato lo hemos calculado en la construcción de los conjuntos de items, por lo que si almacenamos esta información en estructura de datos (se suele utilizar una tabla *hash*) nos evitamos volver a recorrer la base de datos para su cálculo.

Si deseamos utilizar campos con más de un ítem en la parte derecha podemos aprovechar la siguiente propiedad. Si tenemos una regla con varios items en la parte derecha, por ejemplo:

SI bizcochos "Goloso" Y horchata "Xufer" ENTONCES galletas "Trigo" Y vino "Tío Paco"
--

Esta regla puede dividirse en dos reglas con un ítem en la parte derecha:

SI bizcochos "Goloso" Y horchata "Xufer" ENTONCES vino "Tío Paco"
SI bizcochos "Goloso" Y horchata "Xufer" ENTONCES galletas "Trigo"

La cobertura de ambas reglas será idéntica a la regla anterior, por tener la misma parte izquierda. En cuanto a la confianza, se puede asegurar que será igual o mayor a la regla con dos items en la parte derecha, ya que el número de registros que cumplen la parte

izquierda en los que aparezca *vino* “Tío Paco” será mayor o igual al número de registros que cumplan la parte izquierda en los que aparezcan *vino* “Tío Paco” y *galletas* “Trigo”.

De la misma manera, podemos observar que para que una regla con dos ítems en la parte derecha cumpla los requisitos de precisión y confianza es necesario que las dos reglas formadas por la misma parte izquierda y por cada uno de los ítems en la parte derecha también cumplan los requisitos. Este hecho facilita considerablemente la generación de reglas con más de un ítem en la parte derecha, ya que podemos hacerlo de manera incremental: primero se derivan (como hemos comentado previamente) las reglas con un ítem en la parte derecha; a partir de este conjunto de reglas, se derivan las de dos ítems en la parte derecha; utilizando el último conjunto se derivan las de tres ítems en la parte derecha; y así sucesivamente hasta que ya no se encuentren reglas válidas.

En cada paso, debemos asegurar que la confianza se cumpla, ya que la confianza de una regla con dos ítems en la parte derecha (tal como hemos comentado previamente) será igual o inferior que la menor confianza entre las dos reglas con las que se deriva.

Cabe resaltar que el procedimiento utilizado para construir las reglas con $(i+1)$ ítems en la parte derecha a partir de las reglas con i ítems en la parte derecha es similar al procedimiento empleado para la generación de los conjuntos de ítems de tamaño $(i+1)$ a partir de los conjuntos de ítems de tamaño i .

9.2.1 Mejoras y extensiones

Con el objetivo de mejorar la búsqueda de reglas de asociación, así como ampliar el campo de aplicación de estas estrategias se han definido diferentes variantes del algoritmo básico de aprendizaje de reglas de asociación visto previamente. Estas variantes se han basado en el perfeccionamiento de los puntos clave que constituyen un mayor consumo de recursos, por ejemplo, minimizar el acceso a la base de datos para recabar información sobre ítems, minimizar el número de registros a ser inspeccionados para el aprendizaje de las reglas, etc.

Para reducir el número de accesos a la base de datos se suelen utilizar estructuras de datos complejas que permiten un mejor acceso a la información almacenada que se va recabando de los ítems de la base de datos. Una posible opción es la utilización de tablas *hash*. Otra opción propuesta es la utilización de estructuras de tipo árbol. Por ejemplo [Han et al. 2000] introducen el uso de la estructura árbol-FP “*Frequent Pattern Tree*” (*FP-tree*). Esta estructura permite mejorar la generación y prueba de los conjuntos de ítems candidatos.

Otra aproximación para la optimización del tiempo de aprendizaje de reglas de asociación consiste en la aplicación de técnicas de muestreo de registros (véase el Capítulo 5). Dado que las reglas de asociación buscan expresar los patrones más frecuentes, parece claro que trabajar sobre una muestra de registros representativos de la base de datos permitiría obtener resultados parecidos a trabajar sobre todos los datos, y al mismo tiempo se reduciría de manera considerable el tiempo de aprendizaje. Un desarrollo de esta técnica de muestreo para el aprendizaje de reglas de asociación se puede encontrar en [Toivonen 1996].

Otra técnica similar a la utilización de muestreo sobre los registros es la de realizar un filtro sobre los atributos para rechazar aquellos que se consideren poco relevantes. A esta técnica se le llama selección de atributos (*feature selection*), y en realidad, tal como vimos en la Sección 5.4.2, se puede emplear para reducir la complejidad de cualquier problema de

minería de datos. Ejemplos de trabajos sobre esta técnica para aprendizaje de reglas de asociación pueden encontrarse en [Kohavi & John 1997] y [Liu & Setiono 1998].

Una manera mejor de abordar bases de datos con gran volumen de datos es la utilización de técnicas de paralelización. Estas técnicas consisten en modificar los algoritmos de manera que puedan ser ejecutados de manera concurrente en máquinas con varios procesadores. La paralelización de varios algoritmos ha sido tratada en [Adano 2001]. La ejecución en este tipo de máquinas supone un avance bastante importante en la aplicabilidad del aprendizaje de reglas de asociación para bases de datos grandes. Sin embargo las máquinas multi-procesador son por lo general muy caras, por lo que su utilización está bastante más restringida que las computadoras con único procesador.

Finalmente, las reglas de asociación en su versión original sólo trabajan con atributos discretos, es decir, no soportan el uso de atributos numéricos. Con el fin de subsanar esta importante limitación se han desarrollado varios trabajos que resuelven este problema de maneras diversas. Por lo general, estas propuestas se basan en la conversión a tipo categórico de los atributos de tipo numérico (discretizar). Como vimos en el Capítulo 4, una propuesta bastante sencilla es discretizar los elementos de acuerdo con unos rangos predefinidos. Por ejemplo, si tenemos un atributo altura podríamos establecer tres grupos: bajo ($x < 1,50$), mediano ($1,50 \leq x \leq 1,80$), alto ($x > 1,80$). Una estrategia más elaborada es utilizar técnicas de agrupamiento para encontrar los posibles grupos y entonces asignar un valor discreto a cada grupo. Una discusión más detallada acerca de métodos de discretización puede encontrarse en la Sección 4.4.1.

9.3 Reglas de dependencias

El aprendizaje de reglas de asociación representa un potente marco para la búsqueda de patrones en una base de datos. Sin embargo, tiene algunas limitaciones importantes debidas, sobre todo, a que están demasiado orientadas al problema de la extracción de patrones en una cesta de un supermercado o problemas de características similares.

Una primera limitación del marco anterior es que no permite considerar reglas que contemplen la ausencia de un ítem:

SI gaseosa "Chispa" =1 **Y** vino "El cabezón"=1 **ENTONCES** vino "Tío Paco" =0

Esta regla tiene una cobertura igual a 3, y una confianza del 100 por 100, por lo que se podría considerar una regla bastante buena. En realidad, esta limitación se podría subsanar de manera sencilla si modificamos el marco de aprendizaje de reglas de asociación. La idea sería considerar como ítem cada posible valor diferente de un determinado atributo. Por ejemplo, *vino "Tío Paco" =1*, sería un ítem, pero *vino "Tío Paco" =0* también lo sería. De la misma manera, esta idea también se puede extender para atributos nominales que tengan más de dos valores, por ejemplo podríamos tener el ítem *país=Italia*, o el ítem *país=Portugal*, o el ítem *país=Alemania*, etc.

De esta manera las reglas de asociación son muy parecidas a las reglas de clasificación, pero con la salvedad que no hay una clase definida, ya que cualquier atributo puede estar en la parte derecha de una regla de asociación. Además, esta modificación no afectaría al funcionamiento del algoritmo de aprendizaje de reglas de asociación (excepto en la

eficiencia, ya que crece el número de items a considerar), por lo que también sería válido para el aprendizaje de reglas de este tipo.

Otro problema importante lo podemos apreciar en este ejemplo. Considérese que analizamos las ventas de un supermercado. Tenemos una base de datos formada por 100 ejemplos, con una estructura similar a la de la tabla Figura 9.1. En la siguiente tabla, llamada tabla de contingencia²⁹, x representa la presencia del producto x en la cesta de la compra, y $\neg(x)$ la ausencia:

	PAN INTEGRAL	$\neg(\text{PAN INTEGRAL})$	SUMA
Pan Blanco	20	60	80
$\neg(\text{Pan Blanco})$	10	10	20
Suma	30	70	100

Si observamos la regla **SI** “*Pan Integral*” **ENTONCES** “*Pan Blanco*”, esta regla tiene cobertura 20 por ciento y confianza del 66 por ciento. Por tanto, podríamos considerar esa regla como de calidad. Sin embargo, esta apreciación no es del todo correcta, ya que el 80 por ciento de los clientes compran pan blanco, por lo que incluso, comprar pan integral disminuye la probabilidad de que compren pan blanco.

Sea $p(X)$ la probabilidad de que un conjunto de items aparezca en una cesta de la compra. Una manera de calcular la dependencia entre los items “*Pan Integral*” y “*Pan Blanco*” es la siguiente [Silverstein et al. 1997]:

$$\frac{p(\text{"Pan Blanco"} \cap \text{"Pan Integral"})}{p(\text{"Pan Blanco"}) \cdot (\text{"Pan Integral"})} = \frac{0,2}{0,8 \cdot 0,3} = 0,833$$

Si el resultado de esta operación es 1, indica que la aparición en una cesta de cada uno de los items se puede considerar como eventos independientes. Si el resultado es mayor que 1, existe una dependencia positiva, es decir, la compra de un ítem aumenta la probabilidad de la compra del otro ítem. Sin embargo, si el resultado es menor que 1 (como en este caso), existe dependencia negativa, es decir, la compra de un ítem disminuye la probabilidad de la compra del otro ítem. A mayor diferencia entre el resultado y 1, mayor dependencia entre ambos eventos. Por lo tanto, se podría concluir que hay una dependencia negativa entre comprar pan blanco y pan integral.

Por otra parte, si hallamos la dependencia entre no comprar pan blanco y pan integral:

$$\frac{p(\neg(\text{"Pan Blanco"}) \cap \text{"Pan Integral"})}{p(\neg(\text{"Pan Blanco"})) \cdot (\text{"Pan Integral"})} = \frac{0,1}{0,2 \cdot 0,3} = 1,66$$

Este resultado indica una fuerte dependencia positiva entre el hecho de no comprar pan blanco, y comprar pan integral. Cabe destacar, sin embargo, que la regla **SI** “ $\neg(\text{Pan Blanco})$ ” **ENTONCES** “*Pan Integral*” tiene una confianza del 50 por ciento, menor a la regla **SI** “*Pan Integral*” **ENTONCES** “*Pan Blanco*”.

Por lo tanto, este ejemplo muestra los problemas para encontrar algunas dependencias importantes entre eventos por parte del aprendizaje de reglas de asociación basadas en cobertura y confianza.

²⁹ Este tipo de tabla (confusión o contingencia) también se vio en el Capítulo 2 con otros fines.

Motivados por estas limitaciones, en [Silverstein et al. 1997] se propone un marco basado en reglas de dependencias para variables binarias. Una regla de dependencia se define como cualquier conjunto de variables atributo que sea dependiente. La noción de dependencia es la establecida en el ejemplo anterior, es decir, dos eventos son independientes si $p(A \cap B) = p(A) \cdot p(B)$.

El método propuesto consiste en medir la significancia de las diferentes dependencias mediante el test ji-cuadrado usado en la estadística clásica para las correlaciones. Veamos cómo funciona. Sea $R = \{i_1, \neg(i_1)\} \times \dots \times \{i_k, \neg(i_k)\}$ el producto cartesiano de los eventos que representan la presencia o ausencia de un ítem en la cesta. Un elemento $r = r_1 \dots r_k$ representa una determinada cesta de la compra, donde el valor de r en una posición indica la presencia o no de ese elemento, también llamado el valor de una celda. Sea $O(r)$ el número de cestas que contienen una celda r_i . Para determinar si una determinada celda es dependiente debemos comparar si el número de elementos difiere del esperado.

El número estimado para un evento se calcula usando el método de ji-cuadrado asumiendo la independencia. Para un determinado evento, se emplea $E(i_j) = O_n(i_j)$ y $E(\neg(i_j)) = n - O_n(i_j)$. Para conjuntos de eventos, se asume independencia para calcular $E(r) = n \cdot E(r_1)/n \cdot E(r_2)/n \cdot \dots \cdot E(r_k)/n$. El valor ji-cuadrado se define como:

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E(r))^2}{E(r)}$$

Este valor determina si los k ítems son independientes entre sí. Para saberlo, se calcula el valor, y se compara con el definido en la estadística ji-cuadrado para los grados de libertad determinados (1 para variables binarias) y para el grado de significancia requerido. Si el valor calculado es menor que el dado por la estadística, no rechazamos la hipótesis de independencia con el grado de significancia requerido.

Veamos un ejemplo. Vamos a estudiar la dependencia de los ítems “Pan Integral” y “Pan Blanco” de acuerdo con la tabla de contingencia visto anteriormente. El valor ji-cuadrado es:

$$\frac{(20 - 80 \cdot 30/100)^2}{(80 \cdot 30/100)} + \frac{(60 - 80 \cdot 70/100)^2}{(80 \cdot 70/100)} + \frac{(10 - 20 \cdot 30/100)^2}{(20 \cdot 30/100)} + \frac{(10 - 20 \cdot 70/100)^2}{(20 \cdot 70/100)} = \\ 0,667 + 0,286 + 2,667 + 1,143 = 4,76$$

Si buscamos el valor de ji-cuadrado para el 95 por ciento de significancia y un grado de libertad, obtendremos 3,85. Dado que 4,76 es mayor que 3,85, podemos descartar la hipótesis de independencia con un 95 por ciento de certeza. Es decir, podemos asegurar con ese rango de certeza que los atributos son dependientes.

Además, podemos extraer más información del cómputo del valor ji-cuadrado ya que el valor que aporta cada celda tiene relación con el grado de dependencia entre los valores de esa celda. Por ejemplo, para el caso anterior la celda correspondiente “Pan Blanco” y $\neg(\text{“Pan Integral”})$ representa la dependencia más importante, ya que aporta la mayor cantidad (2,667) al valor final.

Por lo tanto, el test ji-cuadrado nos permite establecer estadísticamente si dos atributos son dependientes o no. Sin embargo, es importante no caer en la tentación de utilizar el valor calculado final por el test ji-cuadrado para conocer el rango de la dependencia, ya que este valor depende del tamaño de la muestra e incluso sucede que cuando una hipótesis de

independencia es falsa, el valor tiende al infinito cuando el tamaño de la muestra crece. Por lo tanto, es válido utilizar el valor ji-cuadrado para establecer comparaciones de dependencias en una misma base de datos (estableciendo los que se conoce como una matriz de distancias ji-cuadrado) pero, sin embargo, no debe utilizarse para comparar dependencias entre varias bases de datos.

9.3.1 Interés de dependencias

En la sección anterior, hemos establecido cómo comprobar estadísticamente si dos atributos son dependientes a través del test ji-cuadrado. Una manera de comprobar la dependencia interna entre cada uno de los valores de los atributos es mediante las probabilidades de cada uno de los valores.

El interés entre dos eventos x e y se define de manera similar al cálculo de la dependencia entre dos ítems:

$$I(xy) = \frac{p(xy)}{p(x)p(y)}$$

Dado un atributo, consideramos un evento como cada uno de los posibles valores (o ítems) que pueda tomar. Por lo tanto, para una dependencia entre dos atributos binarios, podemos construir una tabla 2×2 que contenga el interés de cada una de las combinaciones. Un valor de interés superior a 1 significa dependencia positiva, mientras un valor inferior a 1 indica dependencia negativa.

Por ejemplo, para el caso de los atributos “Pan Blanco” y (“Pan Integral”) tenemos la siguiente tabla de interés:

	Pan Integral	$\neg(\text{Pan Integral})$
Pan Blanco	0,833	1,071
$\neg(\text{Pan Blanco})$	1,667	0,714

De esta tabla se puede extraer, por ejemplo, que existe una fuerte dependencia positiva, entre comprar pan integral y no comprar pan blanco, y una ligera dependencia negativa entre no comprar pan blanco y no comprar pan integral.

Utilizando el marco de aprendizaje de reglas de asociación basado en cobertura/confianza, fijando una cobertura mínima del 25 por ciento y una confianza del 50 por ciento, veríamos que existen las siguientes 3 reglas:

SI “Pan Blanco” ENTONCES $\neg(\text{Pan Integral})$	Cb=80, Cf=0,75
SI “Pan Integral”) ENTONCES “Pan Blanco”	Cb=30, Cf=0,67
SI $\neg(\text{Pan Integral})$ ENTONCES “Pan Blanco”	Cb=70, Cf=0,86

En este caso, se observa que las mejores reglas en cobertura y confianza coinciden con la relación dominante, sin embargo esto no siempre se cumple.

Se puede demostrar [Silverstein et al. 1997] que existe una correspondencia directa entre el valor de una celda de la tabla de interés con el valor de esa celda utilizado para calcular el valor ji-cuadrado de la relación. Ésta es la causa de que para el caso del problema del pan integral y el pan blanco, la dependencia más significativa coincida con la celda que más aporta al valor ji-cuadrado.

La tabla de interés es útil para conocer el grado de relación entre cada uno de los posibles valores de dos atributos. Sin embargo, es importante recalcar de nuevo que no sirve para establecer comparaciones entre tablas (relaciones) diferentes.

9.3.2 Relación entre interés y correlación

Históricamente, se ha utilizado el coeficiente de correlación para medir la dependencia entre dos series de datos numéricos. La correlación es una medida entre 1 y -1 bastante sencilla que permite establecer el grado de similitud entre las dos series de valores. Podemos utilizar la correlación con atributos nominales, realizando una numerización “n-a-1” (véase la Sección 4.4.2) que será más fácil de interpretar si el atributo es binario.

Además de esta limitación, para el caso de reglas de dependencia, la correlación no es una medida apropiada ya que es la suma normalizada de la dependencia entre cada uno de los valores. Por lo tanto, si obtenemos una correlación cercana a 1, significa que la evolución de los dos atributos es similar, es decir que cuando tengamos a , probablemente también encontraremos b , y cuando tengamos $\neg(a)$ probablemente encontraremos $\neg(b)$. No obstante, no podemos entrar en detalle en la relación, y conocer el grado de dependencia entre, por ejemplo, a y b . Estas relaciones sí son posibles de conocer mediante el cálculo del interés de la relación.

Dado que la correlación unifica todas las dependencias en un único valor, es posible que pierda algún tipo de dependencia. Por lo general, la correlación es útil para capturar dependencias funcionales lineales, mientras que otros tipos de dependencias pueden no detectarse. A pesar de estos problemas, la correlación puede emplearse en una primera fase para detectar dependencias entre atributos, utilizándose a posteriori, las tablas de interés para conocer cuáles son los detalles de las relaciones capturadas.

9.4 Reglas de asociación multinivel

En muchos casos es bastante difícil encontrar relaciones interesantes debido a que los datos están muy dispersos, es decir, existe una gran cantidad de atributos con respecto al pequeño número de items presentes en cada registro. Una medida que permite subsanar este problema consiste en agrupar atributos en categorías. De esta manera, el aprendizaje de reglas se realiza utilizando estas categorías, y es más fácil encontrar reglas con niveles adecuados de confianza o cobertura.

Las reglas de asociación que utilizan varios niveles de conceptos para expresar las relaciones se denominan reglas multinivel. Cuando se desea utilizar reglas multinivel, se debe proporcionar, además de los datos, una jerarquía de conceptos que contiene un árbol de relaciones entre los atributos. Una jerarquía de conceptos, formalmente, define una secuencia de relaciones entre conceptos más específicos a conceptos más generales. Los niveles del árbol se numeran de arriba abajo, comenzando por el nivel superior 0, que representa a todos los conceptos, y finalizando en el nivel inferior donde se sitúan los items. En la Figura 9.3 podemos ver una posible jerarquía de conceptos de cuatro niveles correspondiente a la tabla de la Figura 9.1.

Lo interesante de esta jerarquía es el hecho de que pocos clientes comprarán una marca concreta de bizcochos con una marca concreta de chocolate. Sin embargo, sí será más

común que se compren bizcochos y chocolate independientemente de las marcas. Este comportamiento se puede detectar mediante la utilización de una jerarquía de conceptos como la de la Figura 9.3.

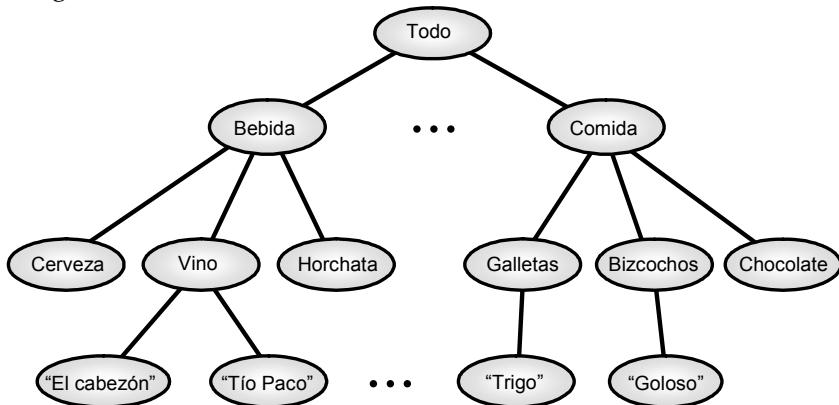


Figura 9.3. Jerarquía de conceptos.

El nivel de agrupamiento, o abstracción, debe ser flexible, de manera que permita al usuario seleccionar el grado de agrupamiento que considere adecuado. Un nivel excesivo de abstracción puede conducir a que se aprendan reglas poco informativas, por ejemplo que si se compra bebida es probable que se compre comida. Por el contrario, un nivel demasiado concreto puede provocar que no se encuentren reglas con suficiente cobertura o confianza.

El algoritmo de aprendizaje de reglas de asociación multi-nivel es bastante similar al de aprendizaje de reglas de asociación clásico. Se utiliza una aproximación basada en la búsqueda de reglas con un mínimo de cobertura/confianza realizando esta búsqueda desde conceptos más generales a conceptos más específicos, hasta que no se encuentran conjuntos de ítems o conceptos que no cumplan los requisitos mínimos. Es decir, primero se buscan los conjuntos de conceptos frecuentes en el nivel 1, a continuación, se prosigue con los conjuntos de conceptos frecuentes del nivel 2, así hasta llegar al nivel inferior. Para cada nivel, se puede utilizar cualquier algoritmo de búsqueda de conjuntos de ítems frecuentes, como Apriori, visto en la Sección 9.2. Existen para el caso de las reglas multi-nivel varias versiones que adecuan los requisitos de la regla de acuerdo al nivel del concepto [Han & Kamber 2001]:

- **Soporte uniforme:** se utiliza el mismo límite de cobertura en todos los niveles. Esta aproximación bastante simple permite, además, optimizar la búsqueda de los conjuntos de ítems debido a la propiedad de que un concepto de un nivel que no cumpla con el requisito de cobertura no contiene ningún concepto más específico que cumpla el requisito. Sin embargo, este criterio tiene el problema de que los conjuntos de ítems de niveles inferiores no son tan frecuentes como los de niveles superiores, por lo que no es del todo justo establecer un nivel idéntico en todos los niveles.
- **Soporte reducido:** motivado por la limitación anterior. En cada nivel se establece un límite de cobertura específico que normalmente se va incrementando conforme se desciende en la jerarquía de conceptos, es decir, a conceptos más específicos, límites más altos. El problema de esta aproximación es que la búsqueda de conjuntos de ítems no es tan sencilla como en la anterior ya que ampliando el límite de

cobertura de nivel a nivel provoca que no podamos utilizar la propiedad que minimiza la búsqueda. Para este caso, se han definido varias estrategias, véase por ejemplo [Han & Kamber 2001].

Hasta ahora, hemos contemplado sólo la posibilidad de que, dada una regla de asociación multinivel **SI** *a* **ENTONCES** *b*, *a* y *b* pertenezcan al mismo nivel. También es posible considerar reglas del tipo **SI** *comida* **ENTONCES** *vino*, es decir que combinen items o conjuntos de items de diferente nivel. A este tipo de reglas se le denominan reglas de asociación de niveles cruzados. El aprendizaje de este tipo de reglas es bastante complejo, ya que considerar diferentes niveles en una misma regla dispara el espacio de búsqueda en gran medida.

Otro aspecto que no trataremos, aunque también está siendo estudiado, es el establecimiento de reglas de dependencia multinivel.

9.5 Reglas de asociación secuenciales

Otro tipo de reglas de asociación son las llamadas reglas de asociación secuenciales. Este tipo de reglas expresan patrones de comportamiento secuenciales, es decir, que se dan en instantes distintos (pero cercanos) en el tiempo. Este tipo de información es de importancia crucial en áreas de aplicación tales como el análisis de navegación sobre páginas web. El objetivo es encontrar relaciones del tipo: el 40 por ciento de las personas que consultan la página web de información sobre la cartelera visitan en menos de dos días la página web de compras de entradas de cine. Este tipo de información permite mejorar la estructura de los servidores web de acuerdo con las secuencias más usuales. También es útil en distribución y en marketing ya que nos pueden interesar reglas del estilo: si un cliente compra un microondas y un frigorífico entonces en menos de seis meses compra una lavadora.

En este tipo de problemas se trabaja con una base datos cuyos registros están formados por transacciones con la siguiente información: identificador de usuario o cliente, identificador de tiempo, y un conjunto de items de la transacción. Por ejemplo, un registro en un entorno comercial sería: el usuario “Pepe García” compró el “27/03/2003” una botella de vino “El cabezón” y un paquete de galletas “Trigo”. En un entorno de páginas web: la máquina “123.23.45.127” visitó el día “12/03/2003” las páginas “inform/tiempo.html” e “inform/mapa.html”.

El aprendizaje de reglas de asociación secuenciales se basa en encontrar las secuencias más comunes. Una secuencia se define formalmente como una lista de conjuntos de items de un mismo cliente ordenada por el tiempo.

Por ejemplo, considérese la siguiente tabla:

ID-Cliente	ID-Tiempo	Items
1	15/03/2003	{23,56}
1	17/03/2003	{42,13}
1	18/03/2003	{45,33}
2	12/03/2003	{12,13}
2	18/03/2003	{23,34,5,8}

De esta tabla se podrían extraer las siguientes secuencias:

ID-Cliente	Secuencia
1	$\langle\{23\ 56\}\{42\ 13\}\{45\ 33\}\rangle$
2	$\langle\{12\ 13\}\{23\ 34\ 5\ 8\}\rangle$

Una subsecuencia se define como una secuencia $\langle a_1, a_2, a_3, \dots, a_n \rangle$ contenida en otra secuencia $\langle b_1, b_2, b_3, \dots, b_m \rangle$ tal que existe una serie de enteros $i_1 < i_2 < i_3 \dots < i_n$ y $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, a_3 \subseteq b_{i3}$.

Veamos algún ejemplo de subsecuencias válidas o no válidas:

Secuencia	Subsecuencia	Válido
$\langle\{23,56\}\{42,13\}\{45,33\}\rangle$	$\langle\{23\}\{42,13\}\rangle$	✓
$\langle\{23,56\}\{42,13\}\{45,33\}\rangle$	$\langle\{42\}\{45\}\rangle$	✓
$\langle\{23,56\}\{42,13\}\{45,33\}\rangle$	$\langle\{45\}\{42\}\rangle$	✗
$\langle\{23,56\}\{42,13\}\{45,33\}\rangle$	$\langle\{42,13\}\rangle$	✓
$\langle\{42,33\}\rangle$	$\langle\{42\}\{33\}\rangle$	✗

En este ámbito, la cobertura de una secuencia se define con respecto a secuencias de conjuntos de ítems de un cliente. Concretamente, la cobertura de una secuencia es la proporción de clientes que contiene esa determinada secuencia. Por lo tanto, el aprendizaje de reglas secuenciales tiene como objetivo hallar las secuencias de conjuntos de ítems que cumplan una mínima cobertura.

Uno de los algoritmos más populares para el aprendizaje de reglas de asociación secuenciales es el algoritmo *AprioriAll* [Agrawal & Srikant 1995].

Este algoritmo se divide en cinco fases:

1. **Ordenación:** se ordena la base de datos tomando el ID-Cliente como clave primaria, y el ID-Tiempo como clave secundaria. Se construye una secuencia de conjuntos de ítems por cada cliente.
2. **Selección de conjuntos de ítems:** en esta fase se seleccionan los conjuntos de ítems que cumplen con una mínima cobertura, calculando la cobertura con respecto a cada cliente, es decir, utilizando la tabla resultante del paso anterior.
3. **Transformación y renombramiento:** se le asigna a cada conjunto de ítems frecuente un identificador. Cada secuencia se transforma de manera que sólo contenga sus ítems frecuentes. A continuación se renombra cada conjunto por su identificador en cada una de las secuencias.
4. **Construcción de secuencias frecuentes:** a partir del conjunto de ítems frecuente se construye incrementalmente el conjunto de secuencias que cumplen con el criterio de cobertura.
5. **Selección de secuencias máximas:** filtra el conjunto de secuencias frecuentes de manera que no haya subsecuencias partiendo desde las secuencias de mayor tamaño.

Veamos un ejemplo de la aplicación de este algoritmo. Para ello vamos a utilizar la siguiente base de datos:

ID-Cliente	ID-Tiempo	Items
1	15/03/2003	{30}
1	17/03/2003	{90}
2	18/03/2003	{10,20}
2	12/03/2003	{30}
2	18/03/2003	{40,60,70}
3	18/03/2003	{30,50,70}
4	13/03/2003	{30}
4	15/03/2003	{40,70}
4	17/03/2003	{90}
5	14/03/2003	{90}

El primer paso consiste en ordenar la tabla por ID-cliente e ID-tiempo, y extraer las secuencias por cada cliente:

ID-Cliente	Secuencia
1	<{30}{90}>
2	<{10 20}{30}{40 60 70}>
3	<{30 50 70}>
4	<{30}{40 70}{90}>
5	<{90}>

La segunda fase consiste en determinar los conjuntos de items con una mínima cobertura con respecto a las secuencias por cliente. Si tomamos la cobertura mínima igual a dos tendremos los siguientes conjuntos de items y un identificador asignado a cada uno de ellos:

Conjuntos de items	Identificador
{30}	1
{40}	2
{70}	3
{40 70}	4
{90}	5

A continuación, en la fase 3, se transforman las secuencias dejando sólo los conjuntos de items frecuentes y renombrándolos de acuerdo con un identificador:

ID-Cliente	Secuencia	Secuencia transformada	Sec. transformada y renombrada
1	<{30}{90}>	<{{30}}{{90}}>	<{1}{5}>
2	<{10 20}{30}{40 60 70}>	<{{30}}{{40}, {70}, {40,70}}>	<{1}{2,3,4}>
3	<{30 50 70}>	<{{30}, {70}}>	<{1,3}>
4	<{30}{40 70}{90}>	<{{30}}{{40},{70},{40 70}}{{90}}>	<{1}{2,3,4}{5}>
5	<{90}>	<{{90}}>	<{5}>

En la fase 4, se generan incrementalmente las secuencias que cumplen con un mínimo de cobertura, entendiendo cobertura como el número de clientes que presentan esa secuencia. La siguiente tabla contiene las secuencias, ordenadas por tamaño, que superan una cobertura del 40 por ciento, es decir, dos clientes:

Tamaño 1		Tamaño 2		Tamaño 3		Tamaño 4	
Secuencia	Soporte	Secuencia	Soporte	Secuencia	Soporte	Secuencia	Soporte
<1>	4	<1 2>	2	<1 2 3>	2	<1 2 3 4>	2
<2>	2	<1 3>	3	<1 2 4>	2		
<3>	3	<1 4>	2	<1 3 4>	2		
<4>	2	<1 5>	2	<2 3 4>	2		
<5>	3	<2 3>	2				
		<2 4>	2				
		<3 4>	2				

Finalmente, el paso 5 reduce el conjunto de secuencias eliminando las subsecuencias existentes partiendo desde las secuencias con más elementos. El conjunto final es:

Secuencia	Secuencia Original	Cobertura
<1 2 3 4>	<{30} {40} {70} {40 70}>	2
<1 5>	<{30} {90}>	2

Existen otros algoritmos que permiten la búsqueda de secuencias frecuentes de manera más sofisticada, por ejemplo *Generalized Sequential Patterns* (GSP) [Agrawal & Srikant 1996], permite considerar restricciones temporales, es decir secuencias de una determinada duración, así como items con diferentes niveles de abstracción, o secuencias multi-nivel.

9.6 Aprendizaje de reglas de asociación con sistemas de minería de datos

En esta sección vamos a ver cómo utilizar algunos sistemas genéricos de minería de datos para aprender reglas de asociación. En concreto, vamos a examinar los paquetes Clementine y WEKA, que se describen en el Apéndice A.

Para mostrar las posibilidades de estos dos sistemas para inducir reglas de asociación, vamos a trabajar con la base de datos *Adult*. En esta base de datos cada registro representa información censal de un individuo extraída del censo de Estados Unidos. Incluye 48.842 individuos, cada uno con 15 atributos tales como: país de nacimiento, nivel de estudio, sexo... Se puede encontrar más información en el Apéndice B. Existen algunos registros con atributos desconocidos. Originalmente, esta base datos ha sido utilizada fundamentalmente para labores de clasificación, donde el objetivo es aprender el nivel de ingresos a partir del resto de atributos.

En este problema, existen algunos atributos continuos, por lo que es imposible la aplicación directa de los algoritmos de aprendizaje de reglas de asociación. Para evitar este problema, WEKA dispone de un útil filtro en el paquete "WEKA.filters.DiscretizeFilter" que permite discretizar automáticamente atributos numéricos. Como vimos en la Sección 4.4.1 (página 89), este paquete permite utilizar dos métodos: uno para problemas de aprendizaje supervisado, y otro para problemas de aprendizaje no supervisado. En este caso vamos a utilizar el método por defecto, pero indicando con el parámetro "-B" que cada atributo numérico se distribuya en diez valores. Con el parámetro "-R" indica los atributos a discretizar. El comando sería:

```
java WEKA.filters.DiscretizeFilter -B 10 -R 1,3,5,11,12,13 -i adult.arff -o adult-disc.arff
```

De esta manera se genera un archivo “adult-disc.arff” donde los atributos numéricos han sido transformados en atributos nominales.

Clementine permite también discretizar atributos utilizando las herramientas de manipulación de atributos. Esta manera es bastante más farragosa, ya que la discretización se realiza de modo más manual que en WEKA. Para trabajar con la misma base de datos en ambos sistemas (y así comparar resultados y prestaciones), utilizaremos en Clementine los datos ya discretizados por WEKA, es decir, el archivo “adult-disc.arff”.

Comencemos analizando WEKA. Este sistema de minería de datos provee el paquete “WEKA.associations.Apriori” que contiene la implementación del algoritmo de aprendizaje de reglas de asociación Apriori. Podemos configurar este algoritmo con varias opciones: con la opción “-U” indicamos el límite superior de cobertura requerido para aceptar un conjunto de items. Si no se encuentran conjuntos de items suficientes para generar las reglas requeridas se va disminuyendo el límite hasta llegar al límite inferior (opción “-M”). Con la opción “-C” indicamos la confianza mínima (u otras métricas dependiendo del criterio de ordenación) para mostrar una regla de asociación; y con la opción “-N” indicamos el número de reglas que deseamos que aparezcan en pantalla. La ordenación de estas reglas en pantalla puede configurarse mediante la opción “-T”, algunas opciones que se pueden utilizar son: confianza de la regla, *lift* (confianza dividida por el número de ejemplos cubiertos por la parte derecha de la regla), y otras más elaboradas.

Tomando los valores por defecto (límite superior de cobertura mínima = 90 por ciento, número de reglas = 10, ordenación por confianza y confianza mínima = 90 por ciento), WEKA obtiene las siguientes reglas:

1. income-level=<=50K 37155 ==> capital-gain='(-inf-9999.9]'	37137	conf:(1)
2. native-country=United-States 43832 ==> capital-gain='(-inf-9999.9]'	42800	conf:(0.98)
3. capital-loss='(-inf-435.6]' 46574 ==> capital-gain='(-inf-9999.9]'	45440	conf:(0.98)
4. race=White 41762 ==> capital-gain='(-inf-9999.9]'	40741	conf:(0.98)
5. capital-loss='(-inf-435.6]' native-country=United-States 41764 ==> capital-gain='(-inf-9999.9]'	40732	conf:(0.98)
6. race=White native-country=United-States 38493 ==> capital-gain='(-inf-9999.9]'	37534	conf:(0.98)
7. race=White capital-loss='(-inf-435.6]' 39754 ==> capital-gain='(-inf-9999.9]'	38733	conf:(0.97)
8. native-country=United-States 43832 ==> capital-loss='(-inf-435.6]'	41764	conf:(0.95)
9. capital-gain='(-inf-9999.9]' 47708 ==> capital-loss='(-inf-435.6]'	45440	conf:(0.95)
10. race=White 41762 ==> capital-loss='(-inf-435.6]'	39754	conf:(0.95)

En cada regla, tenemos la cobertura de la parte izquierda y de la regla, así como la confianza de la regla. Podemos observar que en todas las reglas aparecen los atributos “capital-gain” y “capital-loss”. Una de las razones de este comportamiento es que las variables están distribuidas de una manera no uniforme. Para “capital-gain” el valor “‘(-inf-9999.9]’” representa el 98 por ciento de los casos. En el caso de “capital-loss” el valor “‘(-inf-435.6]’” contiene el 95 por ciento de los casos. En realidad, si observamos los datos, apreciaremos que la mayoría de los registros tienen un cero en este campo, lo que provoca esta aglutinación en los rangos inferiores. Para evitar este problema, vamos a aprender de nuevo reglas de asociación pero sin tener en cuenta estos atributos. En WEKA, este proceso se puede realizar mediante el filtro “WEKA.filters.AttributeFilter”. También eliminamos el atributo 8 “relationship”, ya que es en cierta manera dependiente del atributo 6 “marital-status”, por lo que no aporta demasiada información, e incluso provoca que se encuentren

reglas poco útiles del tipo: si una persona es marido, entonces está casada. En este caso, tomando valores por defecto, las diez reglas con mayor confianza son:

1. education-num='(8.5-10]' race=White 22676 => native-country=United-States	21405	conf:(0.94)
2. education-num='(8.5-10]' 26662 => native-country=United-States	24664	conf:(0.93)
3. education-num='(8.5-10]' income-level=<=50K 22096 => native-country=United-States	20373	conf:(0.92)
4. race=White 41762 => native-country=United-States	38493	conf:(0.92)
5. race=White sex=Male 28735 => native-country=United-States	26450	conf:(0.92)
6. race=White income-level=<=50K 31155 => native-country=United-States	28501	conf:(0.91)
7. workclass=Private race=White 29024 => native-country=United-States	26540	conf:(0.91)
8. race=White hours-per-week='(30.4-40.2]' 21920 => native-country=United-States	19943	conf:(0.91)
9. workclass=Private race=White income-level=<=50K 22282 => native-country=United-States	20180	conf:(0.91)
10. sex=Male native-country=United-States 29223 => race=White	26450	conf:(0.91)

Estas reglas muestran algún patrón interesante como, por ejemplo, la regla 4. Esta regla indica que si la persona es de raza blanca, tiene un 92 por ciento de probabilidad de haber nacido en Estados Unidos según los datos analizados. No obstante, es importante tener en cuenta lo que hemos comentado anteriormente sobre las reglas de dependencia. Ya que la mayoría de elementos de la base de datos son de Estados Unidos, se encuentran muchas reglas con este valor en la parte derecha. Si ordenamos por otras medidas podríamos ver reglas más interesantes.

Veamos ahora las posibilidades del sistema Clementine para el aprendizaje de reglas de asociación. Este sistema también incluye una implementación del algoritmo Apriori. Para trabajar con este componente de aprendizaje, debemos definir todos los atributos que deseemos utilizar como atributos simbólicos de entrada y salida con un componente o nodo “tipo” del repositorio de operaciones con campos en la interfaz gráfica del Clementine.

Podemos configurar el componente de aprendizaje con varias opciones: cobertura mínima de reglas, precisión mínima de reglas y número máximo de condiciones en la parte izquierda de una regla. Hay una opción interesante que consiste en utilizar valores verdaderos para los campos definidos como marcas. Esta opción es útil en el caso de que trabajemos con atributos binarios y sólo queramos reglas con valores positivos (indicando la presencia del ítem). Esta es la situación del clásico análisis de la cesta de la compra. Si marcamos la opción experto, tenemos acceso a opciones avanzadas como por ejemplo variar la medida utilizada para la evaluación (confianza, diferencia de información, jícuadrado, etc.).

Al igual que antes, vamos a utilizar los datos discretizados por WEKA (“adult-disc.arff”) ignorando los atributos 8 (*relationship*), 11 (*capital-gain*) y 12 (*capital-loss*). Utilizamos los siguientes valores para la configuración del nodo Apriori: cobertura mínima 10 por ciento, precisión mínima 80 por ciento, número máximo de condiciones en la parte izquierda de la regla 5. Las diez primeras reglas ordenadas por confianza y cobertura (se toma el valor del producto de la cobertura por la confianza) son:

1. native_country == United-States <= race == White	(41760:85.5%, 0.922)
2. race == White <= native_country == United-States	(43811:89.7%, 0.878)
3. native_country == United-States <= income_level == <=50K	(37169:76.1%, 0.892)
4. race == White <= income_level == <=50K	(37169:76.1%, 0.839)
5. native_country == United-States <= workclass == Private	(33896:69.4%, 0.889)
6. native_country == United-States <= sex == Male	(32626:66.8%, 0.895)

7. race == White <= workclass == Private	(33896:69.4%, 0.856)
8. race == White <= sex == Male	(32626:66.8%, 0.88)
9. native_country == United-States <= race == White & income_level == <=50K	(31161:63.8%, 0.915)
10. race == White <= native_country == United-States & income_level == <=50K	(33115:67.8%, 0.86)

En cada regla tenemos su cobertura, el tanto por ciento que representa es la cobertura, y la confianza de la regla (tanto por uno). Como podemos observar en los resultados de los dos sistemas varias reglas se repiten, pero no todas, ya que ni la implementación, ni las opciones, ni el modo de ordenar las reglas son iguales.

El modelo generado para las reglas de decisión Clementine no lo representa como un modelo de clasificación, sino que cambia el icono indicando que es un “modelo sin refinar”, ya que no hay una clase definida. Es posible extraer un modelo de decisión desde este modelo indicando qué atributo queremos que sea la clase a predecir, así como los requisitos mínimos que han de cumplir las reglas (cobertura y confianza).

Finalmente, Clementine dispone de un gráfico que permite analizar las relaciones o enlaces entre los diferentes items de determinados atributos. A este tipo de gráficos los denomina mallas (*web* en la versión en inglés). Para utilizar este componente, indicamos los atributos a analizar. El gráfico permite mostrar tres tipos de enlaces entre los valores de los atributos: débiles, medios y fuertes. Además, se debe introducir un nivel mínimo para dibujar las relaciones. Estos niveles se pueden expresar en valores absolutos o relativos. Por ejemplo, la malla de la Figura 9.4 representa los enlaces entre los items de los atributos: “*sex*”, “*race*” e “*income-level*”. Los umbrales que establecemos son los siguientes: <1 por ciento (no mostrar); >1 por ciento y <5 por ciento (relación débil); >5 por ciento y <10 por ciento (relación media); >10 por ciento (relación fuerte). En la gráfica, los enlaces débiles se representan con líneas punteadas, los medios con líneas finas y los fuertes con líneas gruesas.

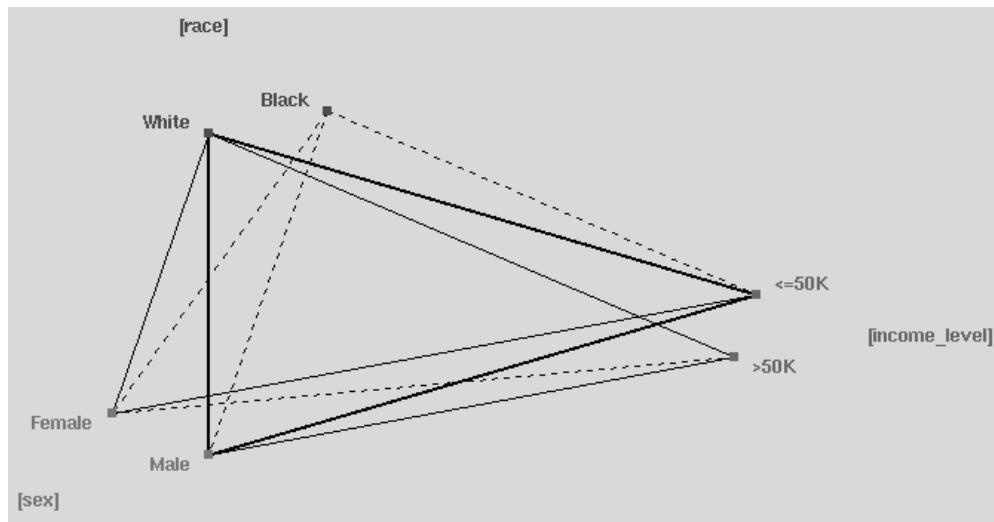


Figura 9.4. Gráfico de malla.

Como se puede observar, no todos los valores del atributo “*race*” aparecen. Esto se debe a que sólo con los valores “*White*” y “*Black*” se pueden establecer enlaces (o conjuntos de items) que superen una cobertura del 1 por ciento. Si queremos conocer con exactitud los

valores de los enlaces podemos utilizar la opción resumir que muestra la cobertura de los enlaces de la gráfica.

Este tipo de gráficas son muy útiles para mostrar visualmente las asociaciones entre dos atributos. No obstante, no permiten visualizar las asociaciones entre más de dos atributos, a diferencia de las reglas de asociación.

Para finalizar, hemos de decir que el número de herramientas específicas de reglas de asociación y dependencias que incluyan al menos los algoritmos Apriori y AprioriAll (o derivados) era, en el momento de escribir este libro, bastante limitado, especialmente las de carácter gratuito, por lo que hemos preferido describir dos herramientas generalistas. Sí existen, en cambio, ciertas implementaciones desde el ámbito científico, que pueden reutilizarse para aplicaciones prácticas con un poco más de esfuerzo. Prueba de ello se puede encontrar en el repositorio FIMI (*Frequent Itemset Mining Implementations*, <http://fimi.cs.helsinki.fi/>) que incluye implementaciones (algunas de ellas variantes muy eficientes) y datos de ejemplo.

Capítulo 10

MÉTODOS BAYESIANOS

José A. Gámez Martín, Ismael García Varea y José M. Puerta Callejón

Uno de los principales problemas a los que se enfrentan las técnicas de minería de datos es cómo trabajar con incertidumbre. Este hecho, que es un problema para muchas de las técnicas de minería de datos en general, no lo es en absoluto para los métodos y técnicas bayesianas, puesto que una de sus principales características es el uso explícito de la teoría de la probabilidad para cuantificar la incertidumbre.

En este capítulo nos centraremos básicamente en el estudio de las redes bayesianas y de su uso en la minería de datos. Veremos que se trata de un modelo que permite un doble uso: *descriptivo* y *predictivo*. En cuanto a su uso como modelo descriptivo, los algoritmos de aprendizaje de redes bayesianas se centran en el descubrimiento de relaciones de independencia y/o relevancia entre sus variables, por lo que el modelo resultante refleja de forma explícita numerosas relaciones de interés. En cierto modo, permiten establecer relaciones mucho más ricas que las reglas de asociación vistas en el capítulo anterior. Además, su explotación posterior mediante el uso de distintas técnicas de inferencia probabilística nos permitirá ampliar el conjunto de relaciones descubiertas y evaluar las distintas hipótesis que podamos formular.

En cuanto al uso predictivo, nos referimos fundamentalmente al uso de las redes bayesianas como clasificadores. Para esta tarea, veremos que podemos construir redes con distinto nivel de complejidad mediante la adición de restricciones al proceso de aprendizaje.

10.1 Introducción

La teoría de la probabilidad y los métodos bayesianos son una de las técnicas que más se han utilizado en problemas de inteligencia artificial y, por tanto, de aprendizaje automático y minería de datos. Como se indica en [Mitchell 1997] dos son las razones por las que los métodos bayesianos son relevantes al aprendizaje automático y a la minería de datos:

1. Son un método práctico para realizar inferencias a partir de los datos, induciendo modelos probabilísticos que después serán usados para razonar (formular hipótesis) sobre nuevos valores observados. Además, permiten calcular de forma explícita la probabilidad asociada a cada una de las hipótesis posibles, lo que constituye una gran ventaja sobre otras técnicas.
2. Facilitan un marco de trabajo útil para la comprensión y análisis de numerosas técnicas de aprendizaje y minería de datos que no trabajan explícitamente con probabilidades. En este capítulo no trataremos esta faceta, aunque en otros capítulos se dan algunas ideas sobre este punto³⁰.

Como ejemplo de la ventaja que supone poder dar la probabilidad asociada a la clasificación, piénsese, por ejemplo, en un sistema de recomendaciones para invertir en bolsa, en el que a partir de unos datos de entrada sobre un determinado producto, el sistema nos recomienda si invertir o no. Evidentemente, si no invertimos no perdemos nada (si acaso lo que nos haya costado la consulta), pero si invertimos podremos multiplicar nuestra inversión o por el contrario perderla parcial o totalmente. Supongamos que consultamos acerca de dos productos diferentes P1 y P2. Si la lógica del sistema recomendador no es capaz de tratar con incertidumbre, la salida podría ser (por ejemplo) afirmativa para ambos productos, con lo que nuestra decisión podría ser diversificar la inversión. Sin embargo, si el sistema estuviera fundamentado en un método bayesiano podríamos obtener la siguiente recomendación: P1 (SI con probabilidad 0,9, NO con probabilidad 0,1) y P2 (SI con probabilidad 0,52, NO con probabilidad 0,48). Sin duda alguna el encargado de tomar la decisión preferiría tener esta información a la proporcionada en el primer caso.

Frente a esta ventaja y al hecho indiscutible de constituir uno de los enfoques teóricamente más sólidos y elegantes, los métodos bayesianos se encuentran con el inconveniente del coste computacional que requiere llevar a la práctica dicho enfoque. De hecho, es habitual formular ciertas suposiciones que de alguna forma permiten restringir la complejidad de los modelos a usar. No faltan las opiniones que afirman que estas suposiciones simplifican tanto la capacidad expresiva de los modelos resultantes que los hacen poco menos que inútiles de cara a su aplicación práctica. Sin embargo, se ha demostrado que a pesar de las simplificaciones realizadas, algunos de estos modelos (como el clasificador Naïve Bayes (NB)) son realmente competitivos e incluso superan a otras técnicas más complejas en determinados casos o aplicaciones. Además, gran parte de estas críticas han perdido su sentido con la aparición de las Redes Bayesianas (RBs) [Pearl 1988], que permiten simplificar el coste computacional del modelo probabilístico, pero sin pérdida de expresividad por parte del mismo.

En este capítulo haremos un repaso por las técnicas bayesianas más empleadas en la minería de datos. Así, comenzaremos por introducir el uso de estos métodos en los problemas de clasificación. A continuación revisaremos el clasificador Naïve Bayes, al que trataremos de forma individual debido a su relevancia. El siguiente apartado se dedicará a las redes bayesianas y a su uso como modelos descriptivos. Posteriormente se estudiarán

³⁰ Por ejemplo, el Capítulo 11 trata la construcción de estimadores de probabilidad usando árboles de decisión. En los capítulos de evaluación y de combinación (17 y 18) también se trata la importancia de acompañar las predicciones con probabilidades.

diversos modelos de redes bayesianas orientadas a tareas de clasificación. La parte final del capítulo estará dedicada a revisar aspectos concretos como el tratamiento de valores desconocidos, y a presentar brevemente varios sistemas o paquetes *software* para trabajar con los métodos bayesianos.

10.2 Teorema de Bayes e hipótesis MAP

En teoría de la probabilidad, el teorema de Bayes es la regla básica para realizar inferencias. Así, el teorema de Bayes nos permite actualizar la creencia que tenemos en un suceso o conjunto de sucesos a la luz de nuevos datos u observaciones. Es decir, nos permite pasar de la probabilidad a priori $P(\text{suceso})$ a la probabilidad a posteriori $P(\text{suceso}|\text{observaciones})$. La *probabilidad a priori* puede verse como la probabilidad inicial, la que fijamos sin saber nada más. Por ejemplo, si estamos en invierno, la probabilidad de que una persona padezca gripe en un momento determinado podría ser del 0,05 ($P(\text{Gripe}=\text{si}) = 0,05$). La probabilidad a posteriori es la que obtendríamos tras conocer cierta información, por tanto, puede verse como un refinamiento de nuestro conocimiento. Por ejemplo, si sabemos que la persona en cuestión se ha vacunado contra la gripe, entonces la probabilidad a posteriori sería prácticamente cero ($P(\text{Gripe}=\text{si}|\text{Vacuna}=\text{si}) \approx 0$).

Teniendo en cuenta estos conceptos, el teorema de Bayes viene representado por la siguiente expresión:

$$P(h|O) = \frac{P(O|h) \cdot P(h)}{P(O)}$$

Donde, como podemos ver, lo que aparecen son la probabilidad a priori de la hipótesis (h) y de las observaciones (O) y las probabilidades condicionadas $P(h|O)$ y $P(O|h)$. A esta última se le conoce como la *verosimilitud* de que la hipótesis h haya producido el conjunto de observaciones O .

Centrándonos en el problema de la clasificación, con una variable clase (C) y un conjunto de variables predictoras o atributos $\{A_1, \dots, A_n\}$, el teorema de Bayes tendría la siguiente forma:

$$P(C|A_1 \dots A_n) = \frac{P(A_1 \dots A_n | C) \cdot P(C)}{P(A_1 \dots A_n)}$$

Evidentemente, si C tiene k posibles valores $\{c_1, \dots, c_k\}$, lo que nos interesa es identificar el más plausible y devolverlo como resultado de la clasificación. En el marco bayesiano, la hipótesis más plausible no es otra que aquella que tiene máxima probabilidad a posteriori dados los atributos, y es conocida como la hipótesis máxima a posteriori o hipótesis *MAP* (del inglés, *maximum a posteriori*). Así, la calse o valor a devolver será:

$$c_{MAP} = \arg \max_{c \in \Omega_C} p(c | A_1, \dots, A_n) = \arg \max_{c \in \Omega_C} \frac{p(A_1, \dots, A_n | c) p(c)}{p(A_1, \dots, A_n)} = \arg \max_{c \in \Omega_C} p(A_1, \dots, A_n | c) p(c)$$

Donde Ω_C representa el conjunto de valores que puede tomar la variable C . Nótese que en el último paso se ha eliminado la división debido a que el divisor sería el mismo para todas las categorías.

Por tanto, el teorema de Bayes nos facilita un método sencillo y con una semántica clara para resolver esta tarea. Sin embargo, este método tiene un problema, y es su altísima

complejidad computacional, debido a que necesitamos trabajar con distribuciones de probabilidad que involucran muchas variables, haciéndolas en la mayoría de los casos inmanejables. En las siguientes secciones veremos cómo este problema es paliado haciendo uso de las (a veces supuestas) independencias entre las variables.

10.3 Naïve Bayes

Sin duda alguna se trata del modelo más simple de clasificación con redes bayesianas (se describirán en la siguiente sección). En este caso, la estructura de la red es fija y sólo necesitamos aprender los parámetros (probabilidades). El fundamento principal del clasificador Naïve Bayes [Duda & Hart 1973; Langley et al. 1992] es la suposición de que todos los atributos son independientes conocido el valor de la variable clase. A pesar de que asumir esta suposición en el clasificador Naïve Bayes (NB) es sin duda bastante fuerte y poco realista en la mayoría de los casos, se trata de uno de los clasificadores más utilizados. Además, diversos estudios (p.e. [Michie et al. 1994]) demuestran que sus resultados son competitivos con otras técnicas (redes neuronales y árboles de decisión entre otras) en muchos problemas y que incluso las superan en algunos otros. Como un ejemplo de problema en el que el clasificador NB se está mostrando como una de las técnicas más eficaces, podemos citar la lucha contra el correo basura o *spam*. Muchos lectores de correo incorporan este clasificador para etiquetar el correo no solicitado.

La hipótesis de independencia asumida por el clasificador NB da lugar a un modelo gráfico probabilístico en el que existe un único nodo raíz (la clase), y en la que todos los atributos son nodos hoja que tienen como único parente a la variable clase. Gráficamente tendríamos la siguiente estructura:

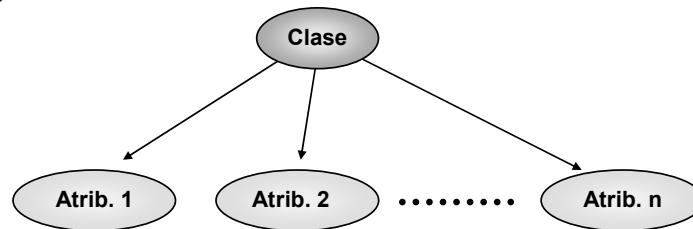


Figura 10.1. Topología de un Clasificador Naïve Bayes.

10.3.1 Estimación de los parámetros

Debido a la hipótesis de independencia usada en el Naïve Bayes, la expresión para obtener la hipótesis MAP queda como sigue:

$$c_{MAP} = \arg \max_{c \in \Omega_C} p(A_1, \dots, A_n | c) p(c) = \arg \max_{c \in \Omega_C} p(c) \prod_{i=1} p(A_i | c)$$

Es decir, la tabla de probabilidad $P(A_1, \dots, A_n | c)$ ha sido factorizada como el producto de n tablas que sólo involucran dos variables. Por tanto, los parámetros que tenemos que estimar son $P(A_i | c)$ para cada atributo y la probabilidad a priori de la variable clase $P(c)$. Veamos cómo hacerlo dependiendo de que el atributo A_i sea discreto o continuo.

- **Atributos discretos:** en este caso la estimación de la probabilidad condicional se basa en las frecuencias de aparición que obtendremos en la base de datos. Así, si llamamos $n(x_i, Pa(x_i))$ al número de registros de la base de datos en que la variable

X_i toma el valor x_i y los padres de X_i ($Pa(X_i)$) toman la configuración denotada por $Pa(x_i)$, entonces la forma más simple de estimar $P(x_i|Pa(x_i))$ es:

$$P(x_i | Pa(x_i)) = \frac{n(x_i, Pa(x_i))}{n(Pa(x_i))}$$

Es decir, el número de casos favorables dividido por el número de casos totales. Esta técnica se conoce como *estimación por máxima verosimilitud* y tiene como desventajas que necesita una muestra de gran tamaño y que sobreajusta a los datos. Existen otros estimadores más complejos que palían estos problemas, entre ellos citaremos el *estimador basado en la ley de la sucesión de Laplace*:

$$P(x_i | Pa(x_i)) = \frac{n(x_i, Pa(x_i)) + 1}{n(Pa(x_i)) + |\Omega_{X_i}|}$$

Es decir, el número de casos favorables más uno dividido por el número de casos totales más el número de valores posibles, como vimos en la Sección 5.4.1, junto a otros estimadores, como el m -estimado. Nótese que con pocos ejemplos, la probabilidad se corrige por la probabilidad uniforme a priori, es decir, uno dividido por el número de valores posibles. Con esta estimación lo que se pretende es que todas las configuraciones posibles tengan una mínima probabilidad, ya que con el estimador de máxima verosimilitud cualquier configuración que no esté presente en la base de datos tendría probabilidad cero.

- **Atributos continuos:** en este caso, el clasificador Naïve Bayes supone que el atributo en cuestión sigue una distribución normal; por tanto, lo único que tenemos que calcular (a partir de la base de datos) es la media μ y la desviación típica σ condicionadas a cada valor de la variable clase.

$$P(A_i | c) \propto \mathcal{N}(\mu, \sigma) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{(X - \mu)^2}{2\sigma^2}\right)$$

Evidentemente, esta estimación tiene el inconveniente de que los datos no siempre siguen una distribución normal.

10.3.2 Ejemplos

Veamos en primer lugar cómo se puede aplicar de una manera sencilla a un pequeño ejemplo ficticio. Posteriormente, veremos un ejemplo más complejo.

Ejemplo 1: supongamos un problema de clasificación con la variable clase C (tomando valores + y -) y dos atributos: D (discreto tomando valores a y b) y N (numérico). Sea el siguiente conjunto de tripletas $[D, N, C]$ nuestra base de datos: $\{[a, 5, +], [a, 2, 2, -], [a, 1, 8, -], [b, 4, +], [b, 2, +], [a, 3, -]\}$.

Para construir el clasificador NB tenemos que estimar $P(C)$, $P(D|C)$ y $P(N|C)$. Aplicando las técnicas anteriores obtenemos las siguientes estimaciones:

$P(C)$	$P(D C)$	+	-	$P(D C)$	+	-	$P(N C)$
+	0,5	a	0,33	1	a	0,4	0,8
-	0,5	b	0,67	0	b	0,6	0,2
		$P(D C)$	Máx. Ver.		$P(D C)$	Laplace	
							$C=+$
							$\mathcal{N}(\mu=3,67, \sigma=1,53)$
							$C=-$
							$\mathcal{N}(\mu=2,33, \sigma=0,61)$
							$P(N C)$

Podemos ver cómo en este caso la estimación basada en Laplace compensa el cero asignado a $P(b| -)$ dado que puede deberse a la pequeñez de la muestra. Usando esta estimación para $P(D|C)$, y usando la distribución normal para el atributo numérico, podríamos ahora clasificar un nuevo caso, p.e. ($D=b$, $N=3,5$):

$$P(+|b, 3,5) = P(+)\cdot P(b|+) \cdot \mathcal{N}(3,67, 1,53:3,5) = 0,5 \cdot 0,6 \cdot 0,26 = 0,078$$

$$P(-|b, 3,5) = P(-)\cdot P(b| -) \cdot \mathcal{N}(2,33, 0,61:3,5) = 0,5 \cdot 0,2 \cdot 0,1 = 0,01$$

Y tras normalizar obtenemos $P(+|b, 3,5) = 0,89$ y $P(-|b, 3,5) = 0,11$, por lo que devolveríamos + como clasificación para el caso tratado.

Ejemplo 2: para finalizar esta sección consideraremos la aplicación de este método sobre uno de la base de datos “Ingresos” (*adult* en su denominación original, véase Apéndice B) que puede usarse para intentar deducir si los ingresos de alguien serán o no superiores a 50.000 dólares. Esta base de datos consta, además de la variable clase, de 14 atributos, ocho discretos y seis numéricos. La base de datos está dividida en un conjunto de entrenamiento con 32.561 casos y en otro de test con 16.281 casos.

Debido a que en las siguientes secciones necesitaremos que todas las variables sean discretas, se ha obtenido una nueva base de datos “Ingresos.d” mediante el siguiente proceso de preprocesamiento:

1. Se han discretizado las variables numéricas usando el método de [Fayyad & Irani 1993], que ya comentamos en el Capítulo 4 (Sección 4.4.1). Tras este proceso las variables discretizadas tienen 12,5 valores en media. Las que ya eran discretas tienen 12,4 valores de media.
2. A consecuencia de los resultados obtenidos, la variable *fnlwgt* ha sido eliminada del conjunto de datos puesto que tras discretizarla sólo puede tomar un valor, siendo por tanto irrelevante para la clasificación.
3. La variable *país-de-nacimiento*, aunque es discreta, puede tomar un alto número de valores (41). Tras estudiar la distribución de casos entre los distintos valores se ha visto que el 90 por ciento pertenecen a un único valor (USA) mientras que el resto se distribuyen entre el resto de las categorías de esta variable. Por ello, se ha decidido hacer esta variable binaria tomando los valores *USA* y *noUSA*. En resumen, después de estos cambios, el número valores por variable es de 13,5 de media.

Los resultados obtenidos (tasa de acierto en el entrenamiento y el test) tras aplicar los algoritmos Naïve Bayes, ID3 (sólo a “Ingresos.d”) ya que sólo trata valores discretos y C4.5 han sido los siguientes³¹:

Algoritmo	Ingresos		Ingresos.d	
	Ent.	Test	Ent.	Test
C4.5	87,79	85,96	87,53	86,70
ID3	--	--	94,10	78,53
Naïve Bayes	83,37	83,00	84,13	84,35

Tabla 10.1. Resultados sobre las bases de datos “Ingresos” e “Ingresos.d”.

³¹ Los algoritmos ID3 y C4.5 se comentan en el Capítulo 11, aunque se consideran una referencia bastante usual a la hora de hacer comparaciones entre métodos. En este caso se han usado las versiones de estos algoritmos disponibles en la herramienta WEKA (Apéndice A): id3 y j48.

Como comentario a estos datos podemos decir que en general C4.5 es el algoritmo que mejor resultado obtiene, puesto que claramente podemos ver que en ID3 se produce un sobreajuste a los datos de aprendizaje. Por otra parte, en este caso C4.5 pierde parte de la interpretabilidad que los árboles (y reglas que de ellos se extraen) poseen, puesto que el modelo generado produciría 681 reglas (532 en *Ingresos.d*) lo que complica el poder realizar una interpretación global. Finalmente, respecto a los resultados del clasificador NB, indicar que son inferiores en este problema a los obtenidos por C4.5 pero, pese a la simplicidad del NB, son competitivos. Además, en otro parámetro a tener en cuenta, como es el tiempo necesario para inducir el modelo NB es claramente ganador puesto que en “Ingresos” necesita 1,66 segundos por 40 de C4.5 y en “Ingresos.d” 0,28 segundos por los seis que requiere C4.5. Esta ventaja se percibe todavía mucho más cuando el número de atributos es muy alto (del orden de cientos, miles o incluso más), haciendo NB mucho más apropiado para este tipo de problemas.

Es importante indicar también que ambos algoritmos se han beneficiado del preprocesamiento realizado, tanto en tiempo de computación como en bondad de la clasificación. Por último, decir que si bien C4.5 realiza una selección de variables inmersa en su proceso de construcción, esto no ocurre con NB, que usa todos los atributos presentes en el conjunto de datos, dándose el caso, además, de que este clasificador es muy sensible a la presencia de variables redundantes o irrelevantes. En la Sección 10.5 se harán más experimentos con esta misma base de datos.

10.4 Redes bayesianas

Las Redes Bayesianas (RBs) son un formalismo que en los últimos años ha demostrado su potencialidad como modelo de representación del conocimiento con incertidumbre. Este formalismo nació como una aportación de diferentes campos de investigación: teoría de toma de decisiones, estadística e inteligencia artificial. El éxito de numerosas aplicaciones en campos variados como la medicina, la recuperación de información, la visión artificial, la fusión de información, la agricultura, etc., avalan este formalismo.

Las RBs representan el conocimiento cualitativo del modelo mediante un grafo dirigido acíclico. Este conocimiento se articula en la definición de relaciones de independencia/dependencia entre las variables que componen el modelo. Estas relaciones abarcan desde una independencia completa hasta una dependencia funcional entre variables del modelo. El hecho de utilizar una representación gráfica para la especificación del modelo hace de las RBs una herramienta realmente muy atractiva en su uso como representación del conocimiento, aspecto muy importante de la minería de datos.

Las RBs no sólo modelan de forma cualitativa el conocimiento sino que además expresan de forma numérica la “fuerza” de las relaciones entre las variables. Esta parte cuantitativa del modelo suele especificarse mediante distribuciones de probabilidad como una medida de la creencia que tenemos sobre las relaciones entre variables de modelo.

Formalmente, una red bayesiana es una tupla $B=(G,\Theta)$, donde G es el grafo y Θ es el conjunto de distribuciones de probabilidad $P(X_i | Pa(X_i))$ para cada variable desde $i=1$ hasta n y $Pa(X_i)$ representa los padres de la variable X_i en el grafo G . Un estudio más detallado se puede encontrar en [Pearl 1988].

Veremos ahora un ejemplo ilustrativo de una RB. Una posible red que refleja las relaciones entre las variables de un pequeño dominio para determinar si un determinado cliente comprará o no una computadora personal puede ser el siguiente:

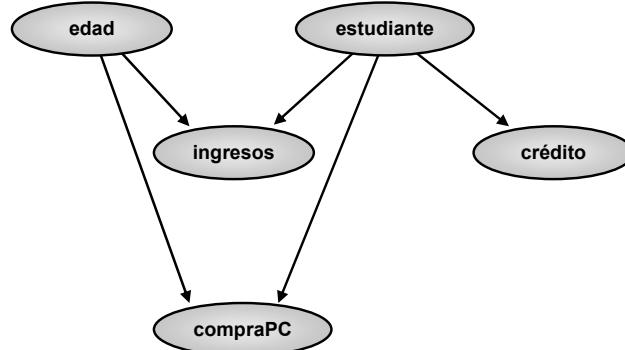


Figura 10.2. Red Bayesiana Compra PC.

En este ejemplo podemos observar que la presencia de un arco directo entre dos variables expresa una relación directa entre las variables que une; de hecho, si interpretamos esta relación como una relación causa-efecto, podremos expresar de forma explícita una manera de representar conocimiento muy habitual entre los expertos en sus determinados dominios. Pero además mediante una RB podremos extraer relaciones de (in)dependencia no directas entre las variables con el criterio conocido como *d-separación* o *separación dirigida*. En nuestro ejemplo, podemos observar que el nivel de ingresos depende directamente de la edad y de su condición de estudiante, la compra de una computadora personal depende directamente de la edad, la condición de estudiante y el nivel de crédito. Otra relación no directa que describe nuestro ejemplo es que la edad es (marginalmente) independiente del nivel de crédito y que los ingresos son independientes de comprar una computadora personal conocida la edad.

El formalismo que vamos a utilizar para representar la parte cuantitativa del modelo de red es la teoría de la probabilidad. En entornos probabilísticos, una distribución de probabilidad P puede ser considerada un modelo de dependencias utilizando la siguiente relación:

$$I(X, Y | Z) \Leftrightarrow P(X | YZ) = P(X | Z),$$

donde X, Y, Z son subconjuntos de variables del modelo y la sentencia $I(., . | .)$ se interpreta como una relación de independencia condicional. La expresión anterior se leería como " X es condicionalmente independiente de Y conocido Z ". En un entorno probabilístico el conocimiento cuantitativo viene expresado por una distribución de probabilidad conjunta definida sobre las variables del modelo. La codificación de las relaciones de independencia condicional expresadas en una RB hacen que la distribución de probabilidad conjunta se pueda almacenar de una manera mucho más eficiente y local a cada una de las variables del modelo, esto es:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | Pa(X_i))$$

Por ejemplo, si nosotros tuviésemos n variables aleatorias binarias (discretas con dos valores), necesitaríamos almacenar $O(2^n)$ parámetros para especificar la distribución de

probabilidad conjunta $P(X_1, X_2, \dots, X_n)$. Sin embargo, una red bayesiana puede necesitar un número exponencialmente menor de parámetros, dado que para cada variable tan sólo hay que almacenar una distribución de probabilidad condicionada a su conjunto de padres en el grafo previamente especificado.

Una vez que hemos definido las RBs, se pueden destacar dos problemas fundamentales para trabajar con este formalismo. Por un lado, se trata de utilizar éstas para realizar procesos eficientes de razonamiento una vez que tengamos especificado el modelo completo tanto cualitativa como cuantitativamente. Para este tipo de tareas han sido muchos los trabajos realizados [Jensen 2001] y veremos a continuación un ejemplo. La existencia de estos algoritmos cada vez más eficientes hace que aparezcan cada vez más aplicaciones prácticas muy interesantes que utilizan las redes bayesianas como motor de inferencia en sus sistemas. Por otro lado, el segundo problema es la tarea de su construcción que describiremos en la siguiente sección.

Por ejemplo, una vez que tenemos definido el modelo de la figura 1, podremos realizar procesos de inferencia. Existen numerosos paquetes *software* (gratuitos o no) que están dedicados al manejo de RBs (véase la Sección 10.8).

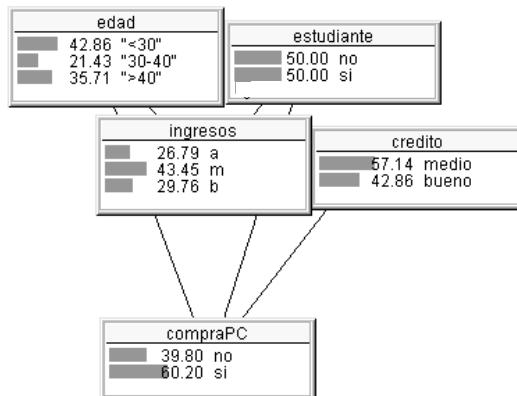


Figura 10.3. Ejemplo de Compra de un PC: probabilidades sin evidencia.

Utilizando cualquiera de ellos, podremos realizar inferencias a partir del modelo de la figura 1, como por ejemplo las probabilidades a priori (véase Figura 10.3), así vemos que en principio el 60 por ciento de la población compraría un PC.

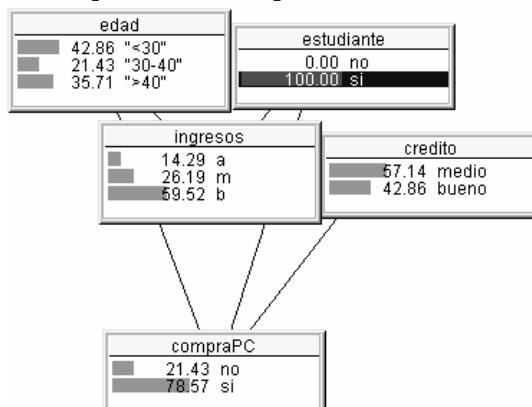


Figura 10.4. Ejemplo de Compra de un PC: probabilidades con evidencia estudiantil.

Sin embargo, nosotros podemos conocer alguna evidencia, como por ejemplo que la persona en cuestión es un estudiante, entonces la probabilidad de la variable estudiante cambiará al 100 por ciento, conocimiento totalmente cierto, y el resto de variables cambiarán en función de la observación que hemos realizado. En nuestro caso la probabilidad de comprar un PC sabiendo que la persona es un estudiante pasa a ser de un 79 por ciento, como se puede observar en la Figura 10.4. En esta figura podemos observar la modificación también del resto de variables.

10.5 Aprendizaje de redes bayesianas

Como hemos visto en la sección anterior, una red bayesiana, una vez construida, constituye un dispositivo potente para el razonamiento probabilístico. Sin embargo, nos queda la tarea de la construcción de tal modelo. Una posibilidad es que un experto, en el dominio que se quiere modelar, construya la red bayesiana a partir de su conocimiento en el problema. Debido al gran volumen de datos de los que habitualmente se dispone en dominios concretos, es de enorme interés proporcionarles a estos expertos herramientas que adquieran este tipo de conocimiento de forma automática a partir de datos de ejemplos del problema en cuestión, para que de esta forma tengan una herramienta de soporte para la decisión.

El problema del aprendizaje de redes bayesianas se puede definir como sigue: dado un conjunto de datos \mathbf{D} , encontrar el grafo dirigido acíclico G que mejor represente el conjunto de dependencias/independencias presentes en los datos. Para solucionar este problema, desde el punto de vista bayesiano, es necesario calcular la probabilidad a posteriori de una red bayesiana en concreto, dado el conjunto de datos conocidos, esto es, $P(G|\mathbf{D})$. Una vez que sabemos cómo calcular esta probabilidad, tendremos una medida de adecuación de cada grafo (red bayesiana) a los datos de partida y por consiguiente podremos comparar, para quedarnos con la mejor, entre distintas redes bayesianas.

Por otra parte, es conocido que el conjunto de redes bayesianas con n nodos es de orden super-exponencial, con lo que la enumeración exhaustiva del conjunto de redes para encontrar la mejor red candidata no es factible, además se ha demostrado que el problema del aprendizaje, tal como lo hemos definido, es un problema NP-duro [Chickering et al. 1996].

Como acabamos de comentar, no es posible, por tanto, la resolución exacta a nuestro problema, por consiguiente, es habitual utilizar algún mecanismo de búsqueda heurística para guiar el problema y así encontrar cada vez mejores redes bayesianas.

Para resolver el problema, en primer lugar vamos a definir algunas medidas o métricas bayesianas de la optimalidad de una red bayesiana respecto a unos datos.

10.5.1 Medidas bayesianas

Sea $G=(V,E)$ un grafo dirigido acíclico que representa una red bayesiana $B=(G,\Theta)$, donde Θ es el conjunto de parámetros definidos en la red, esto es, el conjunto de distribuciones de probabilidad condicional. Sean \mathbf{D} los N casos de datos con la siguiente forma:

$$\mathbf{D} = \begin{bmatrix} X_1[1] & X_2[1] & \dots & X_n[1] \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ X_1[N] & X_2[N] & \dots & X_n[N] \end{bmatrix}$$

El problema consiste en calcular la probabilidad $P(B|\mathbf{D})$, es decir, la probabilidad de una red dados los datos. Aplicando la regla de Bayes, obtenemos que:

$$P(G|\mathbf{D}) = \frac{P(\mathbf{D}|G)P(G)}{P(\mathbf{D})}$$

Como los datos son conocidos y constantes, entonces el término $P(\mathbf{D})$ puede eliminarse de la expresión anterior. Además, una red B posee dos componentes (parámetros y estructura o grafo) con lo que la expresión anterior se traduce en:

$$\begin{aligned} P(G|\mathbf{D}) &\propto P(G)L(\mathbf{D}|G) \\ L(\mathbf{D}|G) &= \int_{\Theta} P(\mathbf{D}|G, \Theta)P(\Theta|G)d\Theta \end{aligned}$$

Siendo $L(\mathbf{D}|G)$ la función denominada verosimilitud marginal. Para el caso discreto, asumiendo que las distribuciones son de la familia exponencial y en concreto para el caso multinomial y su conjugada, la fórmula anterior tiene una solución cerrada dando lugar a una medida bayesiana, conocida como BDe:

$$L(\mathbf{D}|G) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

siendo Γ la función gamma y α_{ijk} los hiperparámetros de la distribución a priori. Si estos hiperparámetros se suponen uniformes, entonces los α_{ijk} pueden igualarse a una constante, denominada tamaño muestral equivalente. Además, N_{ijk} es la frecuencia de aparición en los datos de la variable i -ésima con su k -ésimo valor (de los r_i posibles) y para la configuración j -ésima (de las q_i posibles) de sus padres en el grafo. Por último, tanto N_{ijk} como α_{ijk} es igual a su proyección sobre k , esto es: $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Para un estudio más profundo sobre las suposiciones realizadas y los pasos hasta la expresión anterior podemos consultar los siguientes trabajos [Castillo et al. 1998; Heckerman et al. 1995].

Hemos de destacar que la métrica es descomponible, en el sentido de que se puede calcular de forma local para cada familia (conjunto formado por un nodo y sus padres en G). Por tanto, la métrica podría expresarse, tomando la función logarítmica, como:

$$f(G:\mathbf{D}) = \log L(\mathbf{D}|G) = \sum_{i=1}^n f_i(X_i|Pa(X_i):\mathbf{D})$$

Siendo:

$$f_i(X_i|Pa(X_i):\mathbf{D}) = \sum_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} + \sum_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

Existen otros tipos de medidas de calidad para calcular la adecuación de una red bayesiana a un conjunto de datos. Éstas se basan en otros enfoques provenientes de la teoría de la información y del principio de longitud de descripción mínima, en donde se utilizan criterios de máxima verosimilitud para la estimación de los parámetros en una red

bayesiana, como por ejemplo la métrica denominada BIC (*Bayesian Information Criterion*), la cual tiene la siguiente expresión:

$$f_i(X_i|Pa(X_i):\mathbf{D}) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - 1/2 \text{Dim}(B) \log N$$

donde $\text{Dim}(B)$ representa una función para calcular el número de parámetros libres que se necesitan especificar en una red bayesiana, esto es, $\text{Dim}(B) = \sum_{i=1}^n (r_i - 1)q_i$, (número de entradas independientes en las tablas de probabilidad) y N representa el número total de casos en los datos.

10.5.2 Algoritmos de búsqueda

Una vez especificado el problema del aprendizaje podemos observar que éste se puede plantear como un problema de optimización, esto es, tenemos una métrica definida para calcular la adaptación de los datos a una red bayesiana y por tanto debemos encontrar la solución que maximice esta métrica. Como hemos comentado, este problema no es posible resolverlo eficientemente de forma exacta (incluso con las suposiciones anteriores). Por consiguiente, se han planteado en la literatura específica del tema algoritmos de búsqueda específicos o adaptación de algoritmos metaheurísticos para la resolución aproximada del aprendizaje estructural.

A continuación estudiaremos tres de los algoritmos más utilizados para el aprendizaje, basados en métodos voraces y locales: K2 [Cooper et al. 1992], B [Buntine 1994] y un algoritmo basado en vecindad local y ascensión de colinas que denominaremos HC.

Algoritmo K2

Se puede considerar el primer algoritmo basado en búsqueda u optimización de una métrica bayesiana y ha sido fuente de inspiración para posteriores trabajos sobre el tema. Este algoritmo utiliza un esquema voraz en su búsqueda de soluciones candidatas cada vez mejores y parte de que las variables de entrada están ordenadas, de forma que los posibles padres de una variable aparecen en el orden antes que ella misma. Esta restricción es bastante fuerte pero fue un estándar en el origen de los primeros trabajos sobre aprendizaje de redes bayesianas. El proporcionarle al algoritmo un orden entre las variables hace que éste tan “sólo” tenga que buscar el mejor conjunto de padres posible de entre las variables predecesoras en el orden. La búsqueda de este conjunto se hace de forma voraz.

El algoritmo parte de que el conjunto de padres para cada variable es el conjunto vacío. Posteriormente, y siguiendo el orden establecido, pasa a procesar cada variable X_i , calcula la ganancia que se produce en la medida utilizada al introducir una variable X_j como parente de X_i de entre todas sus predecesoras, esto es, para todo $j < i$ y se queda con la que produce la mejor ganancia. Desde un enfoque bayesiano, esta ganancia se calcula como la razón de la métrica vista en el punto anterior de una red bayesiana con la variable X_i sin padres y una red bayesiana donde X_i tenga como parente X_j . Una vez fijado el primer parente de X_i , entonces se prosigue de igual forma, pero esta vez midiendo la razón de una red bayesiana con X_i y el primer parente fijado y una red bayesiana donde se le introduce un segundo parente no insertado previamente. En la Figura 10.5 podemos encontrar el pseudocódigo de este algoritmo.

```

ALGORITMO K2( $X$ :nodos (variables ordenadas),  $D$ :Datos)
Fase de Inicialización:
PARA CADA  $X_i$  (i=0 hasta n)
   $Pa(X_i)$  = conjunto vacío
FIN PARA CADA
Fase Iterativa:
PARA CADA  $X_i$  (i=0 hasta n)
  ok := true
  MIENTRAS ok
    Sea  $X_j$  el nodo tal que  $j < i$  y  $X_j$  no pertenezca a  $Pa(X_i)$  que maximiza
     $f_i(X_i | Pa(X_i) \cup X_j; D)$ .
    SI  $f_i(X_i | Pa(X_i) \cup X_j; D) > f_i(X_i | Pa(X_i); D)$  ENTONCES  $Pa(X_i) := Pa(X_i) \cup X_j$ 
    EN CASO CONTRARIO ok := false
  FIN MIENTRAS
FIN PARA CADA
FIN ALGORITMO

```

Figura 10.5. Pseudocódigo del algoritmo K2.

Algoritmo B

Este algoritmo es también de los primeros en aparecer en la literatura especializada en el tema, y también está basado en la optimización de una métrica de calidad de redes bayesianas. Al igual que el algoritmo K2, éste se basa en un esquema voraz para la construcción de una solución aproximada a partir de la red vacía de enlaces (cada nodo posee inicialmente un conjunto vacío de padres). A diferencia del algoritmo previo, este algoritmo no impone la restricción de proporcionarle como entrada un orden específico entre las variables. El pseudocódigo del algoritmo B se puede observar en la Figura 10.6.

```

ALGORITMO B( $X$ :nodos (variables),  $D$ :Datos)
Fase de Inicialización:
PARA CADA  $X_i$  (i=0 hasta n)
   $Pa(X_i)$  = conjunto vacío
FIN PARA CADA
Fase Iterativa:
  ok := true
  MIENTRAS ok
    Sea  $X_j \rightarrow X_i$  el enlace que maximiza (de todos los enlaces que no formen un
    ciclo dirigido acíclico y previamente no incluido) la medida:
     $f_i(X_i | Pa(X_i) \cup X_j; D) - f_i(X_i | Pa(X_i); D)$ .
    SI  $f_i(X_i | Pa(X_i) \cup X_j; D) - f_i(X_i | Pa(X_i); D) > 0$  ENTONCES  $Pa(X_i) := Pa(X_i) \cup X_j$ 
    EN CASO CONTRARIO ok := false
  FIN MIENTRAS
FIN ALGORITMO

```

Figura 10.6. Pseudocódigo del algoritmo B.

Este algoritmo trata de introducir el arco que mayor ganancia representa con respecto a la red anterior y a la métrica utilizada. El algoritmo se detiene cuando la inclusión de un arco no representa ninguna ganancia.

Algoritmo HC

Es un algoritmo local de ascensión de colinas (*hill climbing*) por el máximo gradiente basado en la definición de una vecindad. El algoritmo parte de una solución inicial, como podría ser la red vacía de enlaces, u otra cualquiera (por ejemplo, en tareas de clasificación se podría inicializar con una estructura de NB). A partir de esta solución se calcula el nuevo valor de la métrica utilizada de todas las soluciones (grafos) vecinos a la solución actual y nos quedamos con el vecino que mejor valor de la métrica resulte. Estos algoritmos, al igual que los anteriores, se aprovechan de la descomponibilidad de las métricas para recalcular sólo las modificaciones que se realizan en los grafos vecinos definidos.

La vecindad clásica que se maneja en este algoritmo es la siguiente: dado un grafo G (solución actual) se denominan grafos vecinos G' a aquellos resultantes de incluir un solo arco a G o borrar un solo arco presente en G o invertir la dirección de un arco presente en G , todo ello sin incluir ciclos dirigidos en G' . El algoritmo parará cuando no exista ningún vecino que pueda mejorar la solución actual (óptimo local).

10.5.3 Ejemplo: Redes bayesianas y minería de datos

En este apartado vamos a utilizar la base de datos “*Ingresos.d*” y las redes bayesianas como modelo descriptivo. En este caso hemos utilizado el algoritmo HC, previamente descrito, usando como red inicial la red vacía de enlaces y como métrica se utilizará BIC. La estructura de red bayesiana obtenida por este algoritmo se puede observar en la Figura 10.7. Para la especificación de la probabilidad conjunta para este dominio se necesitarían $1,12 \times 10^{12}$ entradas, mientras que la red construida tan sólo requiere 1.193 entradas.

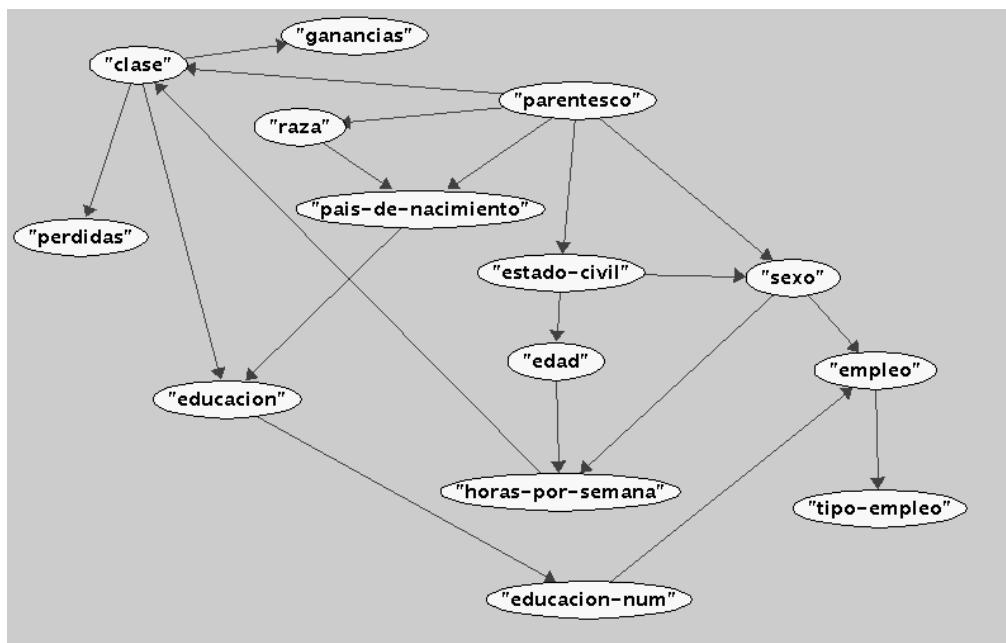


Figura 10.7. Red bayesiana aprendida en el dominio “*Ingresos.d*”.

En esta red bayesiana podemos observar las relaciones directas e indirectas entre las variables correspondientes al dominio utilizado. Entre ellas, por ejemplo, podemos

destacar las relaciones entre “Educación”, “Educación-num”, “Empleo” y “Tipo de empleo” que se observan conectadas en la red resultante; también podemos observar el camino existente entre las variables “Pérdidas”, “Ganancias” y la variable “Clase” (Ingresos). Además, utilizando métodos de inferencia para el cálculo de probabilidades a posteriori un experto podrá analizar los cambios de estas probabilidades conforme vaya introduciendo nuevas observaciones. Pondremos algún ejemplo: la variable “Clase” a priori tiene la distribución de: $\leq 50.000\$ = 0,76$ y $> 50.000\$ = 0,24$, igualmente la variable sexo tiene la distribución: Hombre = 0,67 y Mujer = 0,33. Si ahora fijamos la edad < 21 años, entonces la probabilidad a posteriori de ganar $> 50.000\$$ desciende al 0,05 y la probabilidad de ser mujer aumenta hasta el 0,44, indicando la mayor presencia de mujeres jóvenes en el mercado laboral. Otro caso puede ser, por ejemplo, fijar el nivel de estudios en doctorado, obteniéndose que el nivel de ingresos $> 50.000\$$ aumenta hasta el 0,76.

La aplicabilidad de este tipo de redes para el análisis y comprensión de los datos y las relaciones entre las variables es más que manifiesta. La capacidad descriptiva y explicativa las hacen muy adecuadas para la minería de datos en general y para analizar las relaciones entre atributos de una manera gráfica y mucho más sofisticada que las reglas de asociación o los estudios correlacionales.

10.6 Clasificadores basados en redes bayesianas

Ante el buen rendimiento ofrecido por el clasificador NB, a pesar de la fuerte suposición que realiza, cabe preguntarse si los resultados no serán mejores tras relajar o suprimir dicha suposición (independencia de los atributos dada la clase). En la Sección 10.4 hemos visto que las redes bayesianas permiten tratar modelos complejos gracias a la explotación de las independencias presentes en el modelo. De estas consideraciones surge la idea de usar redes bayesianas como clasificadores [Friedman et al. 1997; Larrañaga 1998].

Partiendo de que el clasificador NB constituye la red bayesiana más sencilla que podemos construir orientada a clasificación, en esta sección veremos otros modelos de redes bayesianas que han destacado por su aplicación a esta tarea. En primer lugar presentamos los métodos TAN y BAN, que consideran a la variable clase de manera especial. En segundo lugar, mostramos el uso de las redes bayesianas generales, como las vistas en el apartado anterior, para clasificación.

10.6.1 TAN

TAN (del inglés *Tree Augmented Naïve Bayes*) constituye una extensión del clasificador NB. La idea es construir una red bayesiana (algo) más compleja que el NB pero donde se da un tratamiento especial a la variable clase; por tanto, se encuadra en la filosofía de aprender redes bayesianas orientadas a clasificación. Con TAN [Friedman et al. 1997], los autores pretenden mantener la simplicidad computacional del clasificador NB pero intentando mejorar la tasa de acierto durante la clasificación. Para ello, en lugar de suponer todas las variables independientes (dada la clase), se admiten ciertas dependencias entre los atributos. En concreto, se supone que los atributos constituyen una red bayesiana con forma de árbol.

La ventaja de restringir la topología de la red (entre los atributos) a un árbol, es que esta estructura puede aprenderse eficientemente. Así, los autores de TAN proponen usar una

ligera modificación del algoritmo de Chow y Liu [Chow & Liu 1968] para realizar este aprendizaje. Este algoritmo está basado en el concepto de *información mutua*:

$$I_P(X, Y | C) = \sum_{x,y,c} P(x, y, c) \log \frac{P(x, y, c)}{P(x | c) \cdot P(y | c)}$$

que mide la cantidad de información que la variable Y nos proporciona sobre la variable X supuesto que el valor de la variable clase (C) es conocido.

Tras aprender la estructura de árbol entre los atributos, el algoritmo TAN añade la variable clase y la hace padre de todos ellos. Este algoritmo (descrito en la Figura 10.8) tiene dos claras ventajas: un bajo coste computacional $O(n^2 \cdot N)$, siendo N el número de instancias en el conjunto de entrenamiento y n el número de variables; y, en segundo lugar, asegura que la estructura de red obtenida es la de máxima verosimilitud del conjunto de todas las posibles estructuras TAN.

ALGORITMO TAN

1. Calcular $I_P(A_i, A_j | C)$ para cada par de atributos ($i \neq j$)
2. Crear un grafo no dirigido con todos los atributos como conjunto de nodos y añadir aristas entre cada par de nodos.
3. Asociar a cada arista (i, j) del grafo el peso $I_P(A_i, A_j | C)$.
4. Construir un árbol expandido de máximo peso a partir del grafo anterior.
5. Elegir un nodo cualquiera del árbol anterior como raíz y direccionar a partir de él el resto de aristas.
6. Añadir la variable clase C y el conjunto de aristas dirigidas $(C \rightarrow A_i)$ para todo atributo A_i .
7. Devolver el modelo TAN obtenido.

FIN ALGORITMO

Figura 10.8. Algoritmo TAN.

Como ejemplo podemos ver en la Figura 10.9 el modelo aprendido por este algoritmo para la base de datos Ingresos.d descrita en la Sección 10.3.

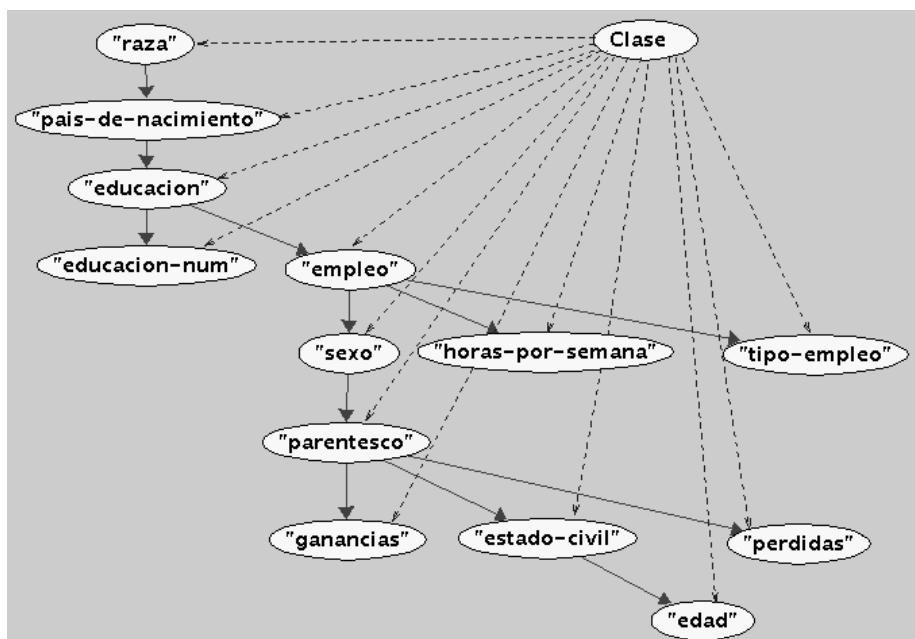


Figura 10.9. Estructura TAN aprendida para el archivo "Ingresos.d".

En trazo continuo se puede ver la estructura de árbol aprendida para los atributos y en discontinuo los arcos que se añaden al introducir la variable clase y la estructura NB.

Finalmente, indicar que existen distintas variaciones de este algoritmo; entre otras, podemos citar las siguientes:

- Métodos de aprendizaje de redes bayesianas que parten de la estructura del Naïve Bayes como estado inicial y, progresivamente añaden arcos hasta llegar a una estructura de TAN. Denotaremos esta variante como TAN_i-m , siendo m el método de aprendizaje utilizado.
- Métodos que simultáneamente al proceso de construcción realizan una selección de variables ($sTAN$).
- Métodos que distribuyen los atributos en un conjunto de árboles, un bosque, en lugar de en un único árbol (FAN , de *Forest Augmented Naive Bayes*).

10.6.2 BAN

El algoritmo BAN se encuadra en la filosofía de TAN, es decir, aprender redes bayesianas orientadas a clasificación. En BAN (de *Bayesian Network Augmented Naive Bayes*) se procede aprendiendo una red bayesiana para los atributos (excluyendo la clase) y posteriormente se aumenta el modelo añadiendo la variable clase C y aristas desde C hacia todos los atributos. Para aprender la red puede usarse cualquier algoritmo de aprendizaje de redes bayesianas (K2, B, ...).

Al igual que ocurre con TAN, existe la posibilidad de iniciar la estructura de red como un NB y a partir de ahí lanzar un algoritmo de aprendizaje que vaya añadiendo arcos. A estos modelos los notaremos como BAN_i-m , siendo m el algoritmo de aprendizaje de redes bayesianas utilizado. La Figura 10.10 muestra la estructura BAN aprendida usando un algoritmo de ascensión de colinas combinado con la métrica BIC como método de aprendizaje.

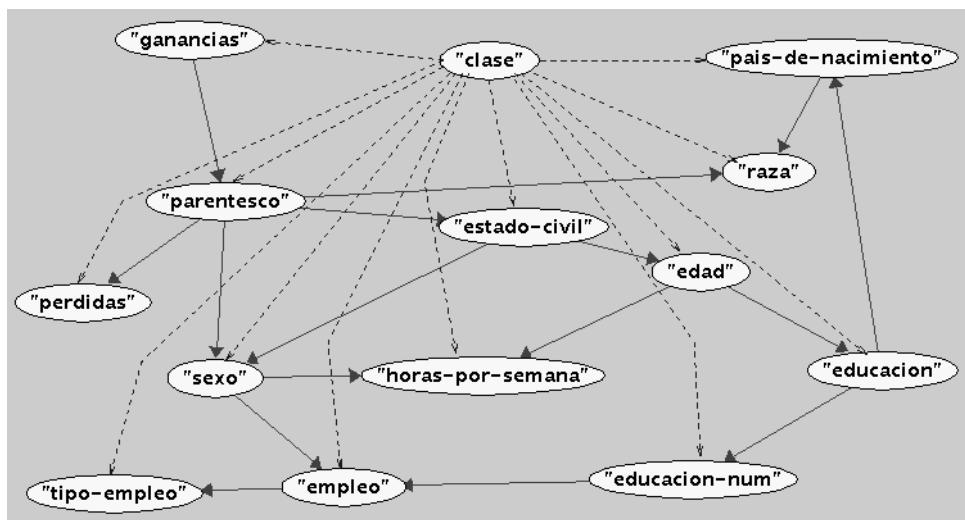


Figura 10.10. Estructura BAN aprendida para el archivo "Ingresos.d".

10.6.3 Redes bayesianas como clasificadores

Evidentemente, en los apartados anteriores también se aprenden redes bayesianas y se usan como clasificadores, la diferencia con este apartado es que se cambia de filosofía. Ahora no se da un tratamiento especial a la variable clase, aprendiéndose así un modelo orientado a clasificación, sino que se aprende una red incluyendo a todas las variables (clase y atributos) del problema, que posteriormente se usará para clasificar. A estos modelos los denotaremos como $RB-m$, siendo m el algoritmo de aprendizaje utilizado. Como ejemplo podemos ver la red de la Figura 10.7, que ha sido obtenida usando el algoritmo de ascensión de colinas combinado con la métrica BIC.

Puesto que en una red bayesiana se cumple que toda variable x es independiente del resto dado su *envolvente (o manto) de Markov* (en inglés, *Markov Blanket*) definido como $(padres(X) \cup hijos(X) \cup padres(hijos(X)))$, podemos seleccionar sólo a dichas variables como variables predictoras útiles para la clasificación. Por ejemplo, en la red de la Figura 10.7 y como se puede ver en la envolvente mostrada en la Figura 10.11, bastaría con considerar el subconjunto de variables $\{\text{parentesco, horas-por-semana, pérdidas, ganancias, educación, país-de-nacimiento}\}$ para realizar la clasificación, puesto que el valor de la clase es independiente de las demás variables conocido el valor de las incluidas en este conjunto.

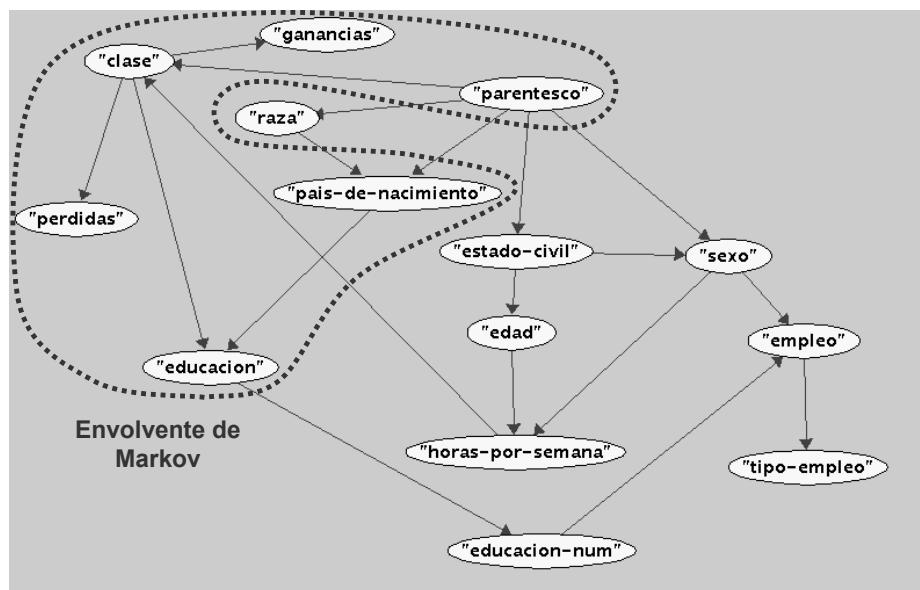


Figura 10.11. Envolvente de Markov sobre la red bayesiana de la Figura 10.7.

10.6.4 Ejemplos

Como ejemplo de los métodos introducidos en esta sección se ha experimentado con algunos de ellos sobre el archivo "Ingresos.d" descrito en la página 262 de la Sección 10.3 (cuando se ha usado el algoritmo K2, como ordenación de las variables se ha considerado el de la base de datos, excepto que la variable clase siempre se pone al final). Además, como se indicó en dicha sección, dado que el clasificador NB es muy sensible a la presencia de variables irrelevantes o redundantes, se ha realizado un proceso de selección (filtrado)

de variables (véase el Capítulo 5) que ha eliminado siete de los 13 atributos originales. Al archivo resultante de la selección lo hemos llamado “Ingresos.d.6”, dado que sólo contiene seis atributos (y, por supuesto, la clase). Los resultados³² obtenidos tanto para “Ingresos.d” como para “Ingresos.d.6” se muestran en la Tabla 10.2 (en negrita el mejor resultado de cada columna sin considerar a ID3 debido al gran sobreajuste que presenta).

Algoritmo	Ingresos.d		Ingresos.d.6	
	Ent.	Test	Ent.	Test
C4.5	87,53	86,70	86,68	86,26
ID3	94,10	78,53	88,24	83,85
Naïve Bayes	84,13	84,35	86,46	86,34
TAN	86,52	85,90	87,00	87,09
TAN-K2	86,55	86,89	86,66	86,55
TAN-B	86,34	85,92	86,96	87,16
FAN	86,46	85,86	86,96	87,16
BAN-K2	86,60	85,90	86,66	86,55
BAN-B	87,60	86,30	86,97	87,16
RB-K2	87,16	86,65	86,66	86,55
RB-B	86,63	86,63	86,46	86,55

Tabla 10.2. Resultados al aplicar los clasificadores bayesianos estudiados.

De estos resultados podemos sacar las siguientes conclusiones:

- Al considerar la posibilidad de dependencias entre los atributos, los resultados mejoran considerablemente a los alcanzados por el NB.
- Al realizar una selección de variables, los resultados del NB mejoran sensiblemente, y son similares o superiores a los obtenidos por C4.5. El resto de los modelos bayesianos también se han visto beneficiados de esta selección.
- Los modelos obtenidos tienen un gran poder de generalización, obteniendo resultados parecidos en los conjuntos de entrenamiento y test. De hecho, en bastantes ocasiones el acierto en el test es mejor que sobre el conjunto de entrenamiento.
- A medida que se admiten modelos más generales, los resultados suelen mejorar, aunque también lo hace el coste computacional de su obtención. Sin embargo, podemos ver que los modelos basados en TAN obtienen unos resultados muy buenos y mantienen un bajo coste computacional.

10.7 Tratamiento de datos desconocidos

En este apartado vamos a repasar brevemente el caso en el que en una base de datos existen registros para los que (por distintas razones) el valor de algunas variables es desconocido (datos faltantes o perdidos) y, por tanto, queda indeterminado en la base de datos. En estos casos no podremos utilizar las métricas definidas en este capítulo, ya que no podremos calcular de forma exacta los estadísticos necesarios, esto es, las frecuencias de una variable con sus padres (denotadas como N_{ijk} en la definición de las métricas). Para

³² Las implementaciones utilizadas (véase la Sección 10.8) son: ID3 y j48 disponibles en WEKA para los algoritmos ID3 y C4.5 respectivamente. Las versiones de Naïve Bayes y TAN disponibles en ELVIRA. La versión de FAN disponible en jBNC. Para el resto de algoritmos se han usado los algoritmos B y K2 disponibles en WEKA, modificándose los parámetros de acuerdo al tipo de clasificador a inducir.

resolver este problema podemos intentar eliminar o sustituir estos datos desconocidos o faltantes, mediante técnicas genéricas como se vio en la Sección 4.2.3, o bien se pueden intentar soluciones más específicas para el caso de las redes bayesianas.

Vamos a distinguir dos casos: a) cuando tenemos un modelo ya especificado y en el conjunto de test o en los nuevos casos existen valores desconocidos, b) cuando en la base de datos de entrenamiento existen valores desconocidos.

El primer caso es cuando encontramos datos desconocidos en alguna de las instancias a clasificar, una vez que tenemos ya especificado el modelo. Este caso se reduce a un problema de propagación de probabilidades en redes bayesianas, introduciendo como evidencia observada los valores del caso que no son desconocidos y como variable objetivo la clase. Como ya se ha comentado en la Sección 10.4, son muchos los métodos (exactos y aproximados) que se han desarrollado para resolver este problema.

El segundo caso es mucho más habitual, pero más difícil de tratar. Por ello, le dedicaremos más atención viendo la utilización del algoritmo EM para esta situación.

10.7.1 Algoritmo EM

Supongamos en este caso que la estructura G de una red bayesiana está perfectamente definida y que por tanto tan sólo tendremos que estimar las tablas de distribución condicional asociadas a cada nodo o variable, como ocurre por ejemplo en el clasificador NB. En este caso se puede utilizar el algoritmo EM (*Expectation Maximization*) [Dempster et al. 1977; McLachlan & Kirshnan 1996]. El algoritmo consta de dos etapas iterativas: en la primera de ellas se calculan las esperanzas de los estadísticos necesarios (frecuencias esperadas N_{ijk}) a partir del modelo de la etapa actual; y el segundo paso es la fase de maximización consistente en reestimar los parámetros a partir de las frecuencias esperadas calculadas en el paso previo. El proceso se reitera hasta su convergencia, la cual está teóricamente garantizada. En la Figura 10.12 se puede observar el algoritmo EM (particularizado para la estimación de parámetros en redes bayesianas).

ALGORITMO EM ($B=(G, \Theta)$: Red Bayesiana, D :Datos)
Fase de Inicialización:
Inicializar el conjunto de parámetros $\Theta^{(0)}$
Inicializar contador de etapas $e=0$
Fase Iterativa:
MIENTRAS no convergencia
Paso E: Etapa de cálculo de Esperanzas $E[N_{ijk} \Theta^{(e)}]$
Paso M: Etapa de maximización
$\Theta_{ijk}^{(e+1)} = E[N_{ijk} \Theta^{(e)}] / E[N_{ij} \Theta^{(e)}]$
FIN MIENTRAS
FIN ALGORITMO

Figura 10.12. Pseudocódigo del algoritmo EM.

Podemos ver el funcionamiento iterativo del algoritmo EM mediante el ejemplo descrito en las tablas siguientes. En este ejemplo tenemos una estructura definida, un clasificador NB, en la parte izquierda de la Figura 10.13 (a) tenemos los datos de partida en la parte superior izquierda, donde las interrogaciones significan datos perdidos o faltantes; en la parte derecha de la Figura 10.13 (b) tenemos la misma base de datos con las posibles imputaciones en las casillas donde tenemos datos perdidos.

Base de datos (a)

Base de datos completada. Proceso de aprendizaje (b)

Figura 10.13. Base de datos y su tratamiento con EM.

En primer lugar hemos de notar que si en un registro no hay ningún dato perdido, entonces este registro tiene una ponderación de 1,0 (esta ponderación aparece en la siguiente columna para cada uno de los pasos del algoritmo en la base de datos completada); si, por el contrario, el registro en cuestión tiene un dato/s perdido/s, entonces es como “si se repartiera” este caso entre los posibles valores con los que podemos completar el registro y por tanto le asignamos la probabilidad de este registro particular imputado como ponderación. Esta probabilidad se calcula con los parámetros del paso actual de la red. Por ejemplo, si asumimos que inicialmente todas las distribuciones son uniformes (situación reflejada en la Tabla 10.3), entonces la ponderación realizada en nuestro ejemplo para cada registro puede verse en la columna Paso 0 de la parte derecha de la Figura 10.13 (b).

P(C)	P(X C)		+	-	P(Y C)		+	-
	+	-			+	-		
+	0,5	a	0,5	0,5	a	0,5	0,5	0,5
-	0,5	n	0,5	0,5	n	0,5	0,5	0,5

Tabla 10.3. Probabilidades en el paso 0 de estimación (uniformes).

Una vez hecho esto, podemos calcular las nuevas frecuencias esperadas a partir de esta nueva base de datos completada (Paso E del algoritmo EM). En la Tabla 10.4 las frecuencias (entre paréntesis) muestran el paso de estimación, y posteriormente recalcular los parámetros de la red, esto es, sus probabilidades para cada nodo o variable (Paso M del algoritmo EM, probabilidades mostradas en la Tabla 10.4).

P(C)	P(X C)		+	-	P(Y C)		+	-
	(2,5) 0,5	a	(2,0) 0,8	(0,5) 0,2	a	(1,5) 0,6	(1,5) 0,6	
-	(2,5) 0,5	n	(0,5) 0,2	(2,0) 0,8	n	(1,0) 0,4	(1,0) 0,4	

Tabla 10.4. Probabilidades en el paso 1 de estimación-maximización.

Las nuevas ponderaciones para cada registro pueden verse en la columna Paso 1 de la Figura 10.13 (b). La Tabla 10.5 y la columna Paso 2 de la figura anterior muestran la siguiente iteración.

P(C)	P(X C)		+	-	P(Y C)		+	-
	(2,5) 0,5	a	(2,3) 0,92	(0,5) 0,2	a	(1,5) 0,6	(1,5) 0,6	
-	(2,5) 0,5	n	(0,2) 0,08	(2,0) 0,8	n	(1,0) 0,4	(1,0) 0,4	

Tabla 10.5. Probabilidades en el paso 2 de estimación-maximización.

Este proceso se reiteraría hasta que los parámetros no cambiasesen de valor de un paso a otro del algoritmo.

En el caso de que la estructura de la red bayesiana no sea conocida, como hemos supuesto en los párrafos anteriores, entonces el proceso se complica, porque no existe una forma cerrada de poder calcular una métrica de adecuación. Sin embargo, existe el algoritmo SEM (*Structural EM*) [Friedman 1998], que también garantiza una convergencia a un óptimo local y que aborda de forma simultánea la búsqueda de la estructura (G) de la red y la estimación de los parámetros.

10.8 Sistemas

Con la reciente aparición de una serie de programas y entornos para abordar el problema del aprendizaje automático con redes bayesianas, podemos decir que hoy en día existe una amplia variedad de opciones³³. En esta sección comentaremos algunos paquetes *software* que por distintos motivos (experiencia en su uso, flexibilidad, etc.) nos son más familiares. Indicaremos con I los que disponen de entorno gráfico de uso, con A los que facilitan un API (una interfaz de programación), con F los que facilitan el código fuente y con G los que son gratuitos.

- **WEKA** (IAFG): se trata de un entorno genérico de minería de datos (véase Apéndice A) que implementa una gran variedad de técnicas. En cuanto a los métodos estudiados en este capítulo, en la versión 3-3-6 podemos encontrar los siguientes:
 - Naïve Bayes. Implementa el NB clásico y además permite usar *kernels* para la estimación en las variables numéricas.
 - Redes bayesianas. Implementa los algoritmos K2 y B. Estos mismos algoritmos pueden ser usados para aprender modelos TANi y BANi. Para ello seleccionaremos la opción inicializar como un NB y limitaremos el número de padres por nodo a 2 en el caso de TAN.

Es de destacar que en todos ellos se puede realizar una combinación con los métodos de selección de variables.

- **ELVIRA** [Elvira Consortium 2002] (IAFG): es un *software* para manipular y crear redes bayesianas construido con dos proyectos de la Comisión Interministerial de Ciencia y Tecnología (CICYT) y que involucra directamente a cinco universidades españolas. Puede descargarse de la dirección <http://leo.ugr.es/~elvira>. En relación con los métodos descritos en este capítulo podemos destacar:
 - Aprendizaje de redes bayesianas. Accesibles desde la interfaz podemos encontrar algoritmos clásicos como el K2 o el PC³⁴, así como algoritmos basados en ascensión de colinas o búsqueda en entorno variable. No obstante, y dada la naturaleza de este proyecto, pueden usarse otros algoritmos desde la línea de

³³ Véase <http://www.ai.mit.edu/~murphyk/Bayes/bnsoft.html> para una lista bastante completa sobre *software* relacionado con redes bayesianas.

³⁴ PC es un algoritmo clásico de aprendizaje de redes bayesianas basado en la detección de independencias condicionales [Spirtes et al. 1991].

comandos y progresivamente se van añadiendo nuevos algoritmos desarrollados por el equipo investigador.

- Naïve Bayes y TAN. Actualmente se encuentran disponibles las implementaciones clásicas de estos algoritmos. No obstante, ésta es una de las líneas del proyecto que más está creciendo, por lo que en breve dispondrá de un conjunto más amplio de algoritmos.

Por último, hay que indicar que también existen algoritmos de preprocesamiento como discretización de variables, imputación de valores perdidos o selección de atributos; y que todos los modelos generados pueden usarse directamente para realizar inferencia sobre ellos.

- **HUGIN** [Madsen et al. 2003] (IA). Se trata del entorno comercial (<http://www.hugin.com>) de creación y uso de redes bayesianas probablemente más usado. En cuanto a las capacidades relacionadas con la minería de datos, podemos destacar:
 - Redes bayesianas. Dispone del algoritmo PC y de una extensión del mismo (NPC) desarrollada por el propio equipo de Hugin. Permite introducir conocimiento a priori sobre la estructura de la red.
 - Estimación paramétrica usando el algoritmo EM, por lo que soporta trabajar con bases de datos incompletas.
- **jBNC** (AFG). Es un paquete de libre disposición (<http://jbnc.sourceforge.net>) dedicado explícitamente a clasificadores bayesianos. Incluye Naïve Bayes, TAN y FAN, así como las versiones de estos modelos en que se hace selección de variables (sTAN y sFAN). Permite además elegir distintas medidas para evaluar los modelos durante la búsqueda.

Recientemente se ha introducido un *plug-in* que permite usar los algoritmos de jBNC desde WEKA.

- **B-course** (IG). Es una herramienta *software* diferente, por cuanto es una página web que da soporte a un servicio de aprendizaje de redes bayesianas vía Internet (<http://b-course.hjit.fi>). Encontramos en él las siguientes capacidades:
 - Redes bayesianas. Tras subir nuestros datos a la página mediante un formulario, éstos son preprocesados (imputación de valores perdidos y discretización). Posteriormente una red bayesiana es inducida usando métodos de búsqueda y evaluación con una métrica bayesiana. Finalmente, el usuario puede analizar las dependencias presentes en la red.
 - Clasificación con RBs. Algoritmos basados en el Naïve Bayes.

Como se ha indicado antes, ésta no es una lista ni mucho menos exhaustiva. Otros paquetes comerciales genéricos (*suites*) también están empezando a incorporar métodos bayesianos.

Finalmente, en este capítulo se han tratado tanto tareas predictivas y descriptivas. Dentro de las primeras se ha tratado exclusivamente el problema de clasificación y dentro de las segundas se ha tratado el establecimiento de relaciones probabilísticas entre variables. No obstante, existen métodos bayesianos aplicables a otras tareas, como la regresión o el agrupamiento. Por ejemplo, existen variantes del algoritmo EM para agrupamiento (como la incluida en el WEKA). Un sistema muy popular de agrupamiento

es el AUTOCLASS [Cheeseman & Stutz 1996], basado en técnicas bayesianas. Este sistema se comenta en el Apéndice A.

Capítulo 11

ÁRBOLES DE DECISIÓN Y

SISTEMAS DE REGLAS

Este capítulo se centra en las técnicas y métodos para el aprendizaje de *modelos comprensibles y proposicionales*: árboles de decisión y sistemas de reglas. El término “modelo” indica que estas técnicas construyen un “modelo”, “hipótesis” o “representación” de la regularidad existente en los datos y, en este sentido, son diferentes de algunos de los métodos vistos anteriormente (métodos basados en similitud, basados en casos o bayesianos). El término “comprensibles” hace referencia al hecho de que estos modelos se pueden expresar de una manera simbólica, en forma de conjunto de condiciones (a diferencia de otros métodos, como las redes neuronales o las máquinas de vectores de soporte) y, por tanto, pueden tener como resultado modelos inteligibles para los seres humanos (y también para sistemas semi-automáticos que procesen reglas). El término “proposicionales” hace referencia a que los métodos que se presentan en este capítulo se restringen a algoritmos que aprenden modelos sobre una única tabla de datos y que no establecen relaciones entre más de una fila de la tabla a la vez ni sobre más de un atributo a la vez. En vez del término proposicional también se suele utilizar frecuentemente el término reglas “atributo-valor”, dado que las condiciones se expresan sobre el valor de un único atributo. En el capítulo siguiente se abordarán métodos de aprendizaje de *modelos comprensibles y relacionales*, que utilizan la lógica de primer orden o de orden superior para representar los modelos y que permiten establecer reglas que relacionan varios atributos y/o tablas.

11.1 Introducción

De todos los métodos de aprendizaje, los sistemas de aprendizaje basados en árboles de decisión son quizás el método más fácil de utilizar y de entender. Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la

decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas. Los árboles de decisión se utilizan desde hace siglos, y son especialmente apropiados para expresar procedimientos médicos, legales, comerciales, estratégicos, matemáticos, lógicos, etc.

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

Consideremos, por ejemplo, un hospital público en el que se realizan operaciones de cirugía refractiva (LASIK) a aquellas personas miopes que lo soliciten. Evidentemente, dichas operaciones no están indicadas en muchos casos, y algunos casos podrían ser excluidos en una primera fase, con el objetivo de evitar riesgos potenciales o efectos secundarios. Aunque la indicación o no de dicha cirugía requiere un examen minucioso por parte del servicio de oftalmología del hospital, existen algunas condiciones claras por las cuales se puede determinar si, en principio, una persona está indicada para el estudio detallado (tensión ocular y paquimetría) y, finalmente, para la operación. En la Figura 11.1, se muestra un ejemplo del árbol de decisión que se utiliza para admitir las solicitudes.

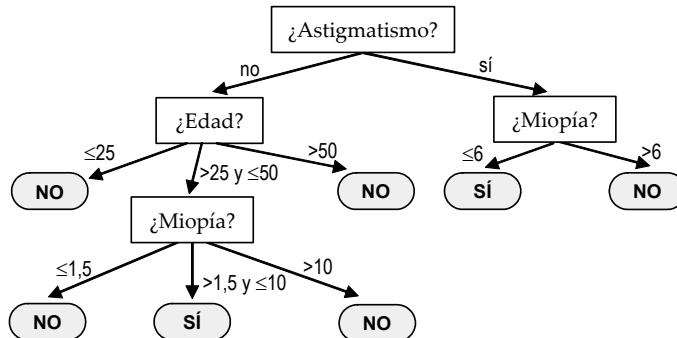


Figura 11.1. Árbol de decisión para determinar recomendación o no de cirugía ocular.

Como se puede observar en la figura es sencillo aplicar el árbol de decisión a un nuevo paciente para saber si se le ha de recomendar o no para dicha operación. Basta realizar las preguntas y seguir las respuestas hasta alguna de las hojas del árbol, catalogadas con un “no” o un “sí”. Este árbol de decisión en concreto funciona como un “clasificador”, es decir, dado un nuevo individuo nos lo clasifica en una de las dos clases posibles: “no” o “sí”.

Lo que nos concierne en este capítulo no es utilizar y entender un árbol de decisión construido (cosa, por cierto, bastante sencilla, como acabamos de comprobar), sino construirlos a partir de datos. Es decir, vamos a concentrarnos en examinar los métodos y algoritmos de *aprendizaje* de árboles de decisión, cómo funcionan y, fundamentalmente, cómo utilizarlos convenientemente para obtener modelos precisos, fiables y, en última medida, inteligibles a partir de datos.

Por otro lado, los sistemas de reglas son una generalización de los árboles de decisión (de hecho, veremos cómo un árbol de decisión se puede expresar como un conjunto de reglas) en el que no se exige exclusión ni exhaustividad en las condiciones de las reglas (es decir, podría aplicarse más de una regla o ninguna).

Respecto al ejemplo de la cirugía refractiva, en la siguiente Figura 11.2 se muestra un ejemplo del árbol de decisión anterior expresado en forma de reglas.

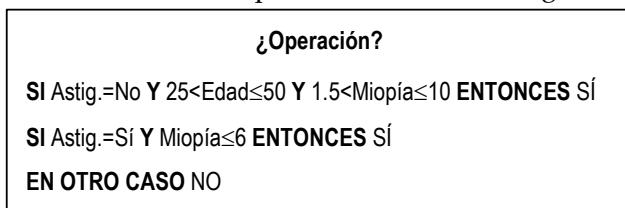


Figura 11.2. Reglas correspondientes al árbol de la Figura 11.1.

La representación en forma de reglas suele ser, en general, más sucinta que la de los árboles, ya que permite englobar condiciones y permite el uso de reglas por defecto, como la que comienza por "EN OTRO CASO" en el ejemplo anterior.

En general, y como veremos a continuación, la diferencia más importante entre los sistemas de aprendizaje de árboles de decisión y los sistemas de inducción de reglas proposicionales es la filosofía del algoritmo que utilizan: partición o cobertura.

11.2 Sistemas por partición: árboles de decisión para clasificación

La tarea de aprendizaje para la cual los árboles de decisión se adecuan mejor es la clasificación. De hecho, clasificar es determinar de entre varias clases a qué clase pertenece un objeto; la estructura de condición y ramificación de un árbol de decisión es idónea para este problema. La característica más importante del problema de la clasificación es que se asume que las clases son *disjuntas*, es decir, una instancia es de la clase *a* o de la clase *b*, pero no puede ser al mismo tiempo de las clases *a* y *b*. Un ejemplo de clasificación sería determinar si una luz en el firmamento es estrella o es planeta: o es una cosa o es la otra. En este sentido, la clasificación se diferencia de la categorización, donde se permiten más de una clase, etiqueta o categoría para cada instancia. Por ejemplo, asignar temáticas a libros es un problema de categorización, ya que un libro puede versar de muchas temáticas.

Debido al hecho de que la clasificación trata con clases o etiquetas disjuntas, un árbol de decisión conducirá un ejemplo hasta una y sólo una hoja, asignando, por tanto, una única clase al ejemplo. Para ello, las particiones existentes en el árbol deben ser también disjuntas. Es decir, cada instancia cumple o no cumple una condición. En el ejemplo de la Figura 11.1, una persona (en realidad un ojo) o bien tiene miopía > 6 , o bien tiene miopía ≤ 6 , pero no las dos cosas a la vez. Además, dicha propiedad es exhaustiva, es decir, una de las dos condiciones se debe cumplir.

Esta propiedad dio lugar al esquema básico de los primeros algoritmos de aprendizaje de árboles de decisión; el espacio de instancias se iba *partiendo* de arriba a abajo, utilizando cada vez una partición, es decir, un conjunto de condiciones excluyentes y exhaustivas. Estos algoritmos se llaman algoritmos de *partición* o algoritmos de "divide y vencerás". Otra característica importante de los primeros algoritmos de aprendizaje de árboles de decisión es que una vez elegida la partición dicha partición no se podía cambiar, aunque más tarde se pensara que había sido una mala elección. Por tanto, uno de los aspectos más importantes en los sistemas de aprendizaje de árboles de decisión es el denominado *criterio*

de partición, ya que una mala elección de la partición (especialmente en las partes superiores del árbol) generará un peor árbol.

En la Figura 11.3 se puede observar un algoritmo básico para generar un árbol de decisión a partir de un conjunto de ejemplos, utilizando la **técnica de “partición”**.

```

ALGORITMO Partición(N:nodo, E:conjunto de ejemplos)
SI todos los ejemplos E son de la misma clase c ENTONCES
    Asignar la clase c al nodo N.
    SALIR; // Esta rama es pura, ya no hay que seguir partiendo. N es hoja.
SI NO:
    Particiones := generar posibles particiones.
    MejorPartición:= seleccionar la mejor partición según el criterio de partición.
    PARA CADA condición i de la partición elegida.
        Añadir un nodo hijo i a N y asignar los ejemplos consistentes a cada hijo (Ei).
        Partición(i, Ei). // Realizar el mismo procedimiento global con cada hijo.
    FIN PARA
    FIN SI
FIN ALGORITMO

```

Para clasificar un conjunto de ejemplos *E*, se invoca con la llamada Partición(*R,E*), donde *R* es un nodo raíz de un árbol por empezar.

Figura 11.3. Algoritmo de aprendizaje de árboles de decisión por “Partición” (*Divide y Vencerás*).

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

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

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

Esto es lo que diferencia fundamentalmente los distintos algoritmos de “partición” existentes hasta la fecha, como CART [Breiman et al. 1984], ID3 [Quinlan 1983][Quinlan 1986], C4.5 [Quinlan 1993], ASSISTANT [Cestnik et al. 1987], etc.

Pasemos en primer lugar a estudiar las particiones a considerar.

11.2.1 Particiones posibles

Las particiones son, como hemos dicho, un conjunto de condiciones exhaustivas y excluyentes. Lógicamente, cuantos más tipos de condiciones permitamos, más posibilidades tendremos de encontrar los patrones que hay detrás de los datos. Por ejemplo, un algoritmo que permita incluir la partición ($x_i \leq a \cdot x_j^2$, $x_i > a \cdot x_j^2$) donde x_i y x_j son atributos del problema y a es una constante, va a poder encontrar patrones cuadráticos, mientras otro algoritmo que no permita dicha partición no va a poder encontrarlos. Cuantas más particiones permitamos más expresivos podrán ser los árboles de decisión generados y,

probablemente, más precisos. No obstante, cuantas más particiones elijamos, la complejidad del algoritmo será mayor. Por tanto, el *quid* de la cuestión en un buen algoritmo de aprendizaje de árboles de decisión es encontrar un buen compromiso entre expresividad y eficiencia.

Debido a esto, la mayoría de algoritmos de aprendizaje de árboles de decisión sólo permiten un juego muy limitado de particiones. Por ejemplo, el algoritmo más popular, denominado C4.5, y desarrollado por Ross Quinlan en la década de 1980 y principios de los 1990 [Quinlan 1993] a partir de un algoritmo anterior denominado ID3 [Quinlan 1983][Quinlan 1986], contiene un solo tipo de partición para los atributos nominales y un solo tipo de partición para los atributos numéricos. Pasemos a describir dichas particiones:

- Particiones nominales: si un atributo x_i es nominal, y tiene posibles valores $\{v_1, v_2, \dots, v_k\}$, sólo existirá una partición posible para dicho atributo y dicha partición será $(x_i = v_1, x_i = v_2, \dots, x_i = v_k)$, es decir, una condición con la igualdad entre el atributo y cada posible valor. Muchos algoritmos siguen esta partición, mientras otros exigen que los árboles sean binarios (sólo dos hijos por nodo) y, por tanto, que las particiones sean binarias (sólo dos condiciones). Para ello, consideran k particiones del estilo $(x_i = v_1, x_i \neq v_1)$, $(x_i = v_2, x_i \neq v_2)$, etc. Nótese que las dos variantes permiten obtener prácticamente los mismos árboles de decisión (equivalentes, al fin y al cabo).
- Particiones numéricas: si un atributo x_i es numérico y continuo, puede haber tomado muchos valores diferentes en los ejemplos y puede tomar infinitos posibles valores en general. Por esta razón, se intentan obtener particiones que separen los ejemplos en intervalos. Para ello, las particiones numéricas admitidas son de la forma $(x_i \leq a, x_i > a)$ donde a es una constante numérica elegida entre un conjunto finito de constantes que discriminan los ejemplos vistos. Por ejemplo, si tenemos diez ejemplos y en ellos aparecen los siguientes valores para el atributo x_i : $\{0,2, 0,3, 0,7, 0,1, 0,8, 0,45, 0,33, 0,1, 0,8, 0\}$, muchos algoritmos realizan el siguiente procedimiento: ordenan los valores (eliminando repetidos), es decir, $\{0, 0,1, 0,2, 0,3, 0,33, 0,45, 0,7, 0,8\}$ y después obtienen el valor intermedio entre cada par de valores. Para el ejemplo anterior, tendríamos que nuestro conjunto finito de constantes sería: $\{0,05, 0,15, 0,25, 0,315, 0,39, 0,575, 0,75\}$. Con estos siete valores, tenemos, por tanto, siete particiones posibles: $(x_i \leq 0,05, x_i > 0,05)$, $(x_i \leq 0,15, x_i > 0,15)$, $(x_i \leq 0,25, x_i > 0,25)$, etc. Generalmente, si existen muchos ejemplos se suele seleccionar un subconjunto de los valores anteriores (mediante análisis de densidad u otras técnicas, véase por ejemplo [Quinlan 1993]), con el objetivo de reducir el número de particiones posibles.

La expresividad resultante de las particiones anteriores se conoce como expresividad proposicional cuadricular. El término proposicional se refiere a que son particiones que sólo afectan a un atributo de un ejemplo a la vez, es decir, ni relacionan dos atributos del mismo ejemplo, ni dos atributos de distintos ejemplos. El término cuadricular hace referencia al tipo de particiones que realizan, especialmente cuando atendemos a los atributos numéricos (véase la Figura 11.4).

Aunque las particiones descritas anteriormente son bastantes sencillas, permiten obtener árboles de decisión bastante precisos y muy comprensibles, ya que las condiciones

$(x_i = v_1, x_i = v_2, \dots, x_i = v_k)$ y $(x_i \leq a, x_i > a)$ se pueden ajustar a muchos patrones y son fácilmente interpretables por los seres humanos.

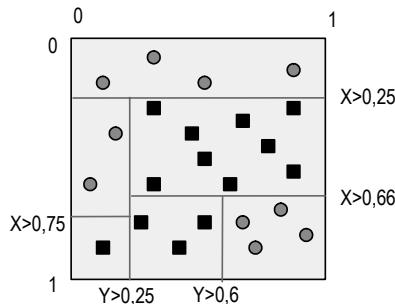


Figura 11.4. Clasificación cuadrangular realizada por un árbol de decisión con dos atributos (X e Y) numéricos.

En el primer ejemplo del capítulo mostrado en la Figura 11.1, se puede ver un ejemplo que utiliza las particiones nominales y numéricas descritas para obtener un árbol de decisión comprensible³⁵.

11.2.2 Criterio de selección de particiones

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

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

Basándose en la idea de buscar particiones que discriminen o que consigan nodos más puros, se han presentado en las últimas dos décadas numerosos criterios de partición, tales como el criterio del error esperado, el criterio Gini [Breiman et al. 1984], los criterios Gain,

³⁵ En realidad, en el ejemplo hay particiones numéricas con tres hijos. Aunque la partición numérica descrita no la contempla directamente, se puede obtener de manera indirecta mediante dos particiones consecutivas sobre el mismo atributo (la superior separando uno en dos intervalos y la inferior refinando en dos uno de los intervalos superiores).

Gain Ratio y la modificación del C4.5 [Quinlan 1993] y el DKM [Kearns & Mansour 1996]. Estos criterios de partición buscan la partición s con el menor $I(s)$, definido de la siguiente forma:

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

donde n es el número de nodos hijos de la partición (número de condiciones de la partición), p_j es la probabilidad de “caer” en el nodo j , p_j^1 es la proporción de elementos de la clase 1 en el nodo j , p_j^2 es la proporción de elementos de la clase 2 en el nodo j , y así para las c clases. Bajo esta fórmula general, cada criterio de partición implementa una función f distinta, como se muestra en la siguiente tabla:

Criterio	$f(p^1, p^2, \dots, p^c)$
Error Esperado	$\min(p^1, p^2, \dots, p^c)$
GINI (CART)	$1 - \sum(p_i)^2$
Entropía (gain)	$\sum p_i \log(p_i)$
DKM	$2(\prod p_i)^{1/2}$

Estas funciones $f(.)$ se denominan funciones de impureza y, por tanto, la función $I(s)$ calcula la media ponderada (dependiendo de la cardinalidad de cada hijo) de la impureza de los hijos en una partición.

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

11.3 Sistemas de aprendizaje de reglas por cobertura

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

¿Operación?
1. SI Astig.=Sí Y Miopía>6 ENTONCES NO
2. SI 25<Edad≤50 Y Miopía≤6 ENTONCES Sí
3. SI Edad>50 ENTONCES NO
4. SI Edad≤25 ENTONCES NO
5. SI Miopía>10 ENTONCES NO
6. EN OTRO CASO Operación=Sí

Figura 11.5. Conjunto de reglas para el problema de la cirugía refractaria.

Por ejemplo, considérese el conjunto de reglas para el problema de la cirugía refractaria de la Figura 11.5. La gran diferencia respecto a las reglas que podrían obtenerse de un árbol de decisión (véase Figura 11.2) es que, según el conjunto de reglas de la Figura 11.5, varias reglas podrían ser aplicables para el mismo ejemplo. Por ejemplo, las reglas 1, 3 y 5 podrían ser aplicables a la vez. En este caso, las tres darían la misma predicción. De hecho, aunque en este conjunto de reglas no hay posibilidad de contradicción, en general, existen métodos que generan conjuntos de reglas que podrían ser, en algunos casos, contradictorias para algunos ejemplos³⁶. La manera de resolver esto es dar un orden a las reglas (denominándose entonces listas de decisión) o ponderando las predicciones diversas.

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

```

ALGORITMO Cobertura(Epos, Eneg: conjunto de ejemplos)
  Reglas :=  $\emptyset$ 
  MIENTRAS Epos  $\neq \emptyset$  Y NO ParadaReglas HACER // Aprender una nueva regla
    NuevaRegla :=  $\emptyset$ ;
    Eneg_Act := Eneg;
    MIENTRAS Eneg_Act  $\neq \emptyset$  Y NO ParadaCondiciones HACER // Aprender una nueva condición.
      Cond := Seleccionar una condición según criterio (elimina muchos negativos)
      NuevaRegla := NuevaRegla  $\cup \{ \text{Cond} \}$  // Añadimos la nueva condición a la regla
      Eneg_Act := Ejemplos negativos consistentes con NuevaRegla;
    FIN MIENTRAS; // La regla está especializada suficientemente. No quedan ejemplos negativos.
    Reglas := Reglas  $\cup \{ \text{NuevaRegla} \}$ ;
    Epos := Epos – Ejemplos cubiertos por NuevaRegla
  FIN MIENTRAS; // El conjunto de reglas ya cubre todos los ejemplos positivos.
  RETORNA Reglas;
FIN ALGORITMO

```

Para clasificar un conjunto de ejemplos positivos y negativos *Epos, Eneg*, se invoca con la llamada Cobertura(*Epos, Eneg*).

Figura 11.6. Algoritmo de aprendizaje de reglas por “Cobertura” (Separa y Vencerás).

En la Figura 11.6 se muestra el algoritmo básico para generar un conjunto de reglas a partir de un conjunto de ejemplos, utilizando la técnica de “cobertura”. Dicho algoritmo tiene como característica fundamental que añade condiciones (pero no conjuntos de condiciones alternativas) a las reglas. Ya que el algoritmo es también “voraz” (no deshace o rectifica las elecciones hechas), el criterio para seleccionar las condiciones (ya sea el número de

³⁶ En cierto modo, el conjunto de reglas obtenidas es algo similar a los conjuntos de reglas de asociación del Capítulo 9 si eligiéramos como atributo de la parte derecha los dos valores de la clase, aunque aquí el objetivo es obtener un conjunto de reglas que permita clasificar todos los ejemplos vistos y los ejemplos futuros (gracias a la regla “EN OTRO CASO”). También, como veremos, el criterio para mantener o seleccionar o no las reglas es diferente del basado en precisión y soporte.

ejemplos negativos excluidos, una medida de impuridad o cualquier otro) es crucial y debe intentar evaluar muy bien y a priori qué particiones son mejores. Este criterio, así como otros aspectos no detallados en el algoritmo de la Figura 11.6, como son la creación de reglas por defecto, criterios de parada o de evaluación que permitan un margen de error en las reglas, los tipos de condiciones (similares a las de los árboles de decisión vistas en la sección anterior o más sofisticadas), el criterio de parada de reglas, el criterio de parada de condiciones, etc., determinan el algoritmo concreto. Además, el algoritmo de la Figura 11.6 se muestra para dos clases pero sólo genera reglas para la clase positiva. En realidad, una modificación sencilla para más de dos clases consiste en seleccionar en cada iteración una regla para la clase mayoritaria en ese momento, considerando el resto de clases como clase negativa, y así sucesivamente. Esta técnica se puede usar también para dos clases y el resultado es que ya aparecen reglas para la clase positiva y para la clase negativa.

Respecto a los criterios *ParadaReglas* y *ParadaCondiciones*, si los eliminamos tenemos un algoritmo que será consistente y completo para la evidencia. Obviamente esto puede no ser bueno si la evidencia tiene ruido.

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

Algunos ejemplos de algoritmos basados en el algoritmo de “cobertura” son AQ [Michalski & Larson 1983; Michalski et al. 1986] y CN2 [Clark & Niblett 1989; Clark & Boswell 1991]. AQ está basado prácticamente en el algoritmo de cobertura de la Figura 11.6 y es además consistente y completo. Para poder tratar con evidencias con ruido, utiliza ciertos pre- y pos-procesos para generalizar las reglas. CN2 es en realidad un rediseño del AQ en el que estos procesos se incluyen en el algoritmo. Además, CN2 tiene en realidad varias versiones. En las primeras versiones [Clark & Niblett 1989] aprende listas de decisión ordenadas, mientras que en las últimas versiones ya se trata de un conjunto de reglas desordenadas [Clark & Boswell 1991]. Otros algoritmos basados en el algoritmo de cobertura son el FOIL [Quinlan 1990; Quinlan & Cameron-Jones 1993] y el FFOIL [Quinlan 1996a]. Estos algoritmos serán estudiados en el Capítulo 12 de aprendizaje relacional.

Para terminar con esta sección vamos a comparar las ventajas e inconvenientes de los algoritmos basados en partición y los algoritmos basados en cobertura:

- Partición: suelen ser más eficientes debido a que cada partición en realidad genera dos o más reglas. La poda, que veremos a continuación, es más sencilla.
- Cobertura: permite hacer coberturas no exhaustivas, que es interesante cuando un atributo tiene muchos valores y sólo algunos son significativos. Es menos voraz y las últimas ramas se ajustan más a los ejemplos. Esto también es una desventaja, porque las últimas ramas suelen hacer sobreajuste (*overfitting*), es decir, intentan capturar aparentes regularidades del conjunto de entrenamiento que resultan ser ruido y el modelo es demasiado específico y *ad-hoc* para los datos de entrenamiento (véase la Sección 6.4.1).

La Figura 11.7 muestra que la capacidad de representación de ambas aproximaciones es muy similar, aunque el algoritmo de “cobertura” puede dejar zonas del espacio que no sean de ninguna clase mientras que el algoritmo de “partición” es exhaustivo.



Figura 11.7. Ejemplos de clasificación cuadricular realizada por el método de partición y el de cobertura.

Hasta ahora no hemos tratado cómo aprender sistemas de reglas que se comporten bien con conjuntos de datos con ruido o que no cometan sobreajuste (*overfitting*). A continuación, se describen una serie de métodos para modificar los algoritmos anteriores en este sentido.

11.4 Poda y reestructuración

Los algoritmos de aprendizaje de árboles de decisión y conjuntos de reglas vistos en las secciones anteriores obtienen un modelo que es completo y consistente con respecto a la evidencia. Es decir, el modelo cubre todos los ejemplos vistos y los cubre todos de manera correcta. Esto puede parecer óptimo a primera vista, pero se vuelve demasiado ingenuo en la realidad. En primer lugar, ajustarse demasiado a la evidencia suele tener como consecuencia que el modelo se comporte mal para nuevos ejemplos, ya que, en la mayoría de los casos, el modelo es solamente una aproximación del concepto objetivo del aprendizaje. Por tanto, intentar aproximar demasiado hace que el modelo sea demasiado específico, poco general y, por tanto, malo con otros datos no vistos. En segundo lugar, esto es especialmente patente cuando la evidencia puede contener ruido (errores en los atributos o incluso en las clases), ya que el modelo intentará ajustarse a los errores y esto perjudicará el comportamiento global del modelo aprendido. Esto, de nuevo, es lo que se conoce como sobreajuste (*overfitting*) (véase la Sección 6.4.1).

La manera más frecuente de limitar este problema es modificar los algoritmos de aprendizaje de tal manera que obtengan modelos más generales. En el contexto de los árboles de decisión y conjuntos de reglas, generalizar significa eliminar condiciones de las ramas del árbol o de algunas reglas. En el caso de los árboles de decisión dicho procedimiento se puede ver gráficamente como un proceso de “poda”, como se ilustra en la Figura 11.8:

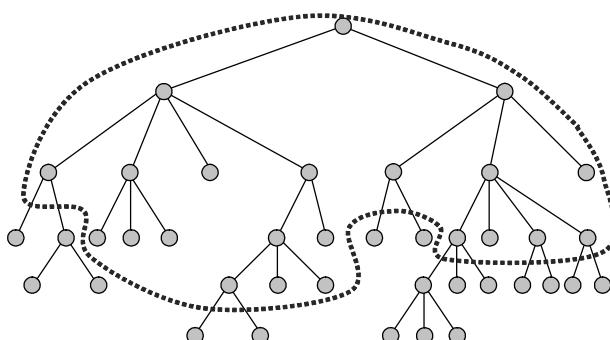


Figura 11.8. Ejemplo de poda. Algunos nodos inferiores son eliminados.

Los nodos que están por debajo del límite de poda se eliminan, ya que se consideran demasiado específicos o, dicho de una manera más informal, se consideran demasiado *ad-hoc*. Aunque utilicemos el término de poda en toda esta sección, las técnicas de generalización también pueden aplicarse a los algoritmos de cobertura (véase, por ejemplo [Cohen 1993]).

En primer lugar cabe distinguir entre métodos de prepoda y pospoda:

- Prepoda: el proceso se realiza durante la construcción del árbol o el conjunto de reglas. Se trata en realidad de determinar el criterio de parada a la hora de seguir especializando una rama o una regla. En general, los criterios de prepoda pueden estar basados en el número de ejemplos por nodo, en el número de excepciones respecto a la clase mayoritaria (error esperado) o técnicas más sofisticadas, como el criterio MDL.
- Pospoda: el proceso se realiza después de la construcción del árbol o el conjunto de reglas. En el caso de los árboles de decisión se trata de eliminar nodos de abajo a arriba hasta un cierto límite. En el caso de los sistemas de reglas se trata de eliminar condiciones, con el objetivo de hacer las reglas más generales. Aunque los criterios están basados en las mismas medidas que la prepoda, dado que la pospoda se realiza con una visión completa del modelo, la pospoda suele obtener mejores resultados. Obviamente, la pospoda es menos eficiente que la prepoda, ya que ésta no genera nada que luego deba eliminarse.

Evidentemente, se pueden combinar técnicas de prepoda y de pospoda. De hecho, algunos de los algoritmos más populares, como el C4.5, utilizan una prepoda por cardinalidad (los nodos con una cardinalidad inferior a una cierta constante no se abren) y una pospoda basada en un criterio más sofisticado. Una consecuencia de utilizar prepoda o pospoda (o los dos) es que los nodos hoja ya no van a ser puros, es decir, es posible que tengan ejemplos de varias clases. Normalmente se elegirá la clase con más ejemplos para etiquetar el nodo hoja y hacer, por tanto, la predicción.

Existen numerosos métodos para podar árboles de decisión o para podar reglas de decisión. Para una comparación de métodos se puede consultar [Esposito et al. 1997] o [Cohen 1993], respectivamente.

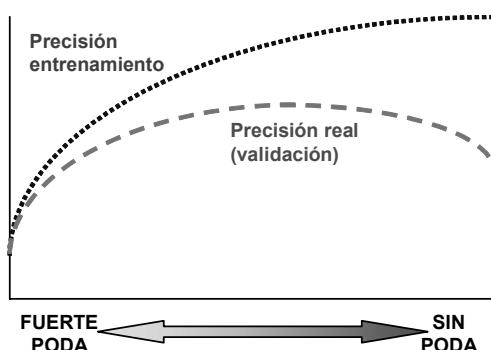


Figura 11.9. El efecto beneficioso de la poda sólo es visible con datos con ruido y con los datos de validación.

Lo que sí que nos interesa determinar es el nivel de poda óptimo. La mayoría de métodos de poda tienen uno o más parámetros que permiten decidir el grado de poda. Aunque los

algoritmos suelen tener un valor por defecto, es imposible encontrar un grado que funcione bien para todos los problemas. Existirán problemas con más ruido, con más ejemplos, o con características especiales que hagan necesario extremar o suavizar la poda. Una manera de poder ajustar el nivel de poda a un determinado problema es observar el comportamiento con respecto a los datos de validación. Por ejemplo, en la Figura 11.9 se muestra la precisión de un árbol de decisión con respecto a los datos de entrenamiento y los datos de validación. Lógicamente, la poda no es beneficiosa para los datos de entrenamiento, pero los datos de entrenamiento no son precisamente una referencia objetiva. En cambio, el comportamiento respecto a los datos de validación es esclarecedor; para cada problema en concreto existe un grado de poda que es óptimo y que se puede estimar mediante el uso de estos datos de validación, si se dispone de ellos.

Esta técnica de examinar el comportamiento respecto a unos datos de validación es precisamente la que se utilizaba en una versión anterior del C4.5 [Quinlan 1987a] denominada “poda de error reducido”, aunque el mismo trabajo incluye otras aproximaciones más sofisticadas, como la poda pesimista o la poda de complejidad de coste. Finalmente, Quinlan se decantó por un método de pospoda de la representación en forma de reglas del árbol generado, que fue finalmente incorporado en el C4.5 [Quinlan 1993].

La poda es una de las primeras y más simples modificaciones que se han ideado para mejorar el comportamiento de los árboles de decisión. Con posterioridad se han definido otra serie de operadores y modificaciones, generalmente denominados operadores de “reestructuración”. Por ejemplo, el C4.5 realiza lo que se conoce como “colapsamiento” (*collapsing*) [Quinlan 1993]. Otros operadores de modificación de la topología del árbol son la “transposición”, la “transposición recursiva” y la “poda virtual” [Utgoff et al. 1997].

El objetivo de estos operadores es, en cierto modo, más ambicioso que la poda. Debido al carácter voraz de los algoritmos de construcción de árboles de decisión es posible que algunas condiciones sean evaluadas peor por los criterios y se pospongan hasta más abajo. Cuando se realiza una visión global del árbol, se puede observar mucho mejor que ciertas partes se pueden reestructurar con el objetivo de simplificar la representación y/o conseguir mayor predictividad. También este tipo de operadores es esencial en el aprendizaje de árboles de decisión de una manera incremental, es decir, cuando los datos se presentan de una manera secuencial y escalonada (no se tienen todos inicialmente).

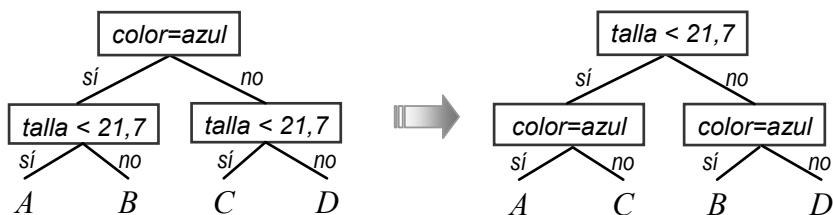


Figura 11.10. Ejemplo del proceso de “transposición”.

Por ejemplo, en la Figura 11.10, se muestra la aplicación del operador de “transposición”. El resultado es un árbol diferente pero equivalente, que, además, podría provocar el desencadenamiento de otros operadores y convertir el árbol en un árbol más simple.

11.5 Árboles de decisión para regresión, agrupamiento o estimación de probabilidades

Aunque los árboles de decisión parecen idóneos para clasificación, se han adaptado para otras tareas, como son la regresión, el agrupamiento o la estimación de probabilidades. De hecho, uno de los primeros algoritmos de aprendizaje de árboles de decisión, el CART [Breiman et al. 1984], es tanto un clasificador como un árbol de regresión.

En realidad un árbol de regresión se construye de manera similar a un árbol de decisión para clasificación, pero con las siguientes diferencias:

- La función aprendida tiene dominio real y no discreto, como en los clasificadores.
- Los nodos hoja del árbol se etiquetan con valores reales, de tal manera que una cierta medida de calidad se maximice, por ejemplo la varianza de los ejemplos que caen en ese nodo respecto al valor asignado.

La implementación más sencilla de esta idea es el propio algoritmo CART, que hace particiones binarias sobre los atributos de igual manera que lo visto para los árboles de decisión diseñados para clasificación, pero que va asignando una media y una varianza a cada nodo, intentando seleccionar las particiones que reduzcan las varianzas de los nodos hijos. Un sistema similar es CHAID [Kass 1980] (derivado de AID y THAID [Morgan & Sonquist 1963; Morgan & Messenger 1973]), que, en realidad, realiza particiones no binarias y usa test-*f* para determinar la partición óptima en el caso de regresión y test ji-cuadrado para clasificación.

Una variación muy popular de los árboles de regresión es considerar una función lineal en los nodos en vez de una media y una desviación típica. En este caso, en primer lugar, para evaluar las particiones se puede utilizar, por ejemplo, el error cuadrático medio de la regresión lineal de los ejemplos que hayan caído en cada nodo. En segundo lugar, para los nodos hoja, la predicción se realiza utilizando el modelo lineal. Nótese que un modelo lineal se puede obtener de una forma relativamente sencilla y eficiente para un conjunto de ejemplos, como vimos en el Capítulo 7. También hay que destacar que esta modificación es directa si todos los atributos son numéricos. En caso de que existan también atributos nominales, se debería utilizar algún tipo de regresión lineal que trate con estos atributos nominales.

Además de su uso para regresión, los árboles de decisión han sido modificados para utilizarse en agrupamiento. La primera idea es modificar el criterio de partición y de evaluación para que considere particiones que separen entre zonas densas y poco densas. Esto se sigue haciendo hasta que se llega a zonas muy densas o zonas muy poco densas, constituyendo entonces los nodos del árbol. Los grupos formados corresponden a los nodos de las zonas densas. Un refinamiento de esta idea es el método presentado por [Liu et al. 2000], en el que se consideran todos los ejemplos de una clase “E” (por existentes) mientras que se añaden ejemplos ficticios “N” (por no existentes) uniformemente distribuidos en el espacio. El siguiente paso es simple, se utiliza un método de aprendizaje de árboles de decisión para clasificación (preferiblemente con poda). El resultado es que las reglas obtenidas para la clase “E” serán los *clusters* o grupos formados.

Evidentemente esta idea se puede refinar más y no es necesario crear realmente los puntos “N”, sino que se puede rediseñar el algoritmo, en particular, el criterio de partición,

para que los considere, como si estuvieran allí. No obstante, si no disponemos de un algoritmo de agrupamiento mediante árboles de decisión, podríamos generar los ejemplos ficticios nosotros mismos y utilizar algoritmos de clasificación clásicos, tales como CART o C4.5 para poder hacer el agrupamiento.

Finalmente, los árboles de decisión se pueden utilizar también para la estimación de probabilidades. En este caso, la presentación del problema es similar a la de un problema de clasificación; los ejemplos tienen una etiqueta discreta, denominada clase. La diferencia es que el objetivo de los estimadores de probabilidades es más ambicioso. Como vimos en el Capítulo 6, no se trata de determinar para cada nuevo ejemplo de qué clase es, sino determinar para cada nuevo ejemplo cuál es la probabilidad de que pertenezca a cada una de las clases. Un estimador de probabilidades es especialmente útil cuando se quieren acompañar las predicciones con cierta fiabilidad, o se quieren combinar predicciones de varios clasificadores, o bien se quiere hacer un *ranking* de predicciones (por ejemplo, ordenar los clientes según su rendimiento).

La modificación de un clasificador árbol de decisión para que sea un estimador de probabilidades es bien sencilla. Supongamos que tenemos tres clases: *a*, *b* y *c*. Para cada nodo hoja con una cardinalidad *n* (número total de ejemplos de entrenamiento que caen en ese nodo) tendremos un determinado número de ejemplos de cada clase: n_a , n_b y n_c . Si dividimos cada uno de estos valores por la cardinalidad total, tendremos una estimación de las probabilidades de las clases en ese nodo, es decir: $p_a = n_a/n$, $p_b = n_b/n$, y $p_c = n_c/n$. Este tipo de árboles de decisión modificados de esta manera se denominan **PETs (Probability Estimation Trees)**.

Aunque la conversión es sencilla, las estimaciones de probabilidad obtenidas por los PETs de esta manera no son muy buenas comparadas con otros métodos. No obstante,

- El suavizado de frecuencias de las estimaciones de probabilidad de las hojas, como por ejemplo la corrección de Laplace o el *m*-estimado (véase la Sección 5.4.1), mejoran significativamente las estimaciones, especialmente si se utilizan para hacer *rankings*.
- La poda (o técnicas relacionadas de transformación o de colapsamiento) no es beneficiosa para la estimación de probabilidades. Los árboles sin podar dan los mejores resultados.

Teniendo en cuenta estas consideraciones, los árboles de decisión constituyen también una buena herramienta para la estimación de probabilidades.

Finalmente, los árboles de decisión se han utilizado frecuentemente como métodos de envolvente (*wrapper*) para la selección de atributos (algunos de estos métodos se vieron en la Sección 5.4.2). En realidad, los criterios de partición eligen el atributo que sea más discriminante, es decir, más significativo. Utilizando los valores obtenidos por cualquiera de los criterios de partición podemos determinar un orden de significatividad de los atributos. Esto se puede realizar considerando sólo la partición al nivel superior del árbol (en la raíz) o estudiando el uso que se hace de los atributos en todas las particiones del árbol, ponderándolas convenientemente. Por ejemplo, el algoritmo CHAID, comentado anteriormente, se ha utilizado más para tareas de selección de atributos y detección de interacciones [Kass 1975] entre los atributos que para tareas de clasificación o regresión. De hecho, el nombre de CHAID proviene de *CHi-squared Automatic Interaction Detector*.

11.6 Aprendizaje de árboles de decisión híbridos

Debido a la popularidad de los árboles de decisión, su estructura básica de aprendizaje se ha intentado combinar con otras técnicas de aprendizaje, dando lugar a métodos híbridos. Existen hibridaciones tanto de los árboles de decisión para clasificación como para regresión.

Un ejemplo de modificación de árboles de decisión para clasificación es lo que se conoce como árboles de decisión perceptrón [Utgoff 1989] o los árboles de decisión multivariantes basados en máquinas lineales (LMDT) [Brodley & Utgoff 1995] que en vez de utilizar particiones que involucran un único atributo de una manera sencilla, al estilo de $(x_i \leq a, x_i > a)$, permiten particiones que se basan en perceptrones (discriminantes lineales) o en conjuntos de perceptrones (denominados *máquinas lineales*). Esto permite extender la representación para tratar conceptos que no sean cuadriculares y así capturar mejor otro tipo de patrones basados en cualquier combinación de fronteras lineales. El número de nodos de las particiones depende del número de regiones obtenidas por el modelo utilizado en la partición. Los árboles de decisión oblicuos (OC1) [Murthy et al. 1994] son otra variación de los métodos anteriores, con heurísticas para evitar mínimos locales. En realidad todos estos métodos son una evolución de una variación del algoritmo CART [Breiman et al. 1984, Capítulo 4] para contemplar particiones lineales, denominado frecuentemente CART-LC.

Para acabar, una extensión importante de los árboles de decisión es lo que se conoce como “grafos de decisión” [Oliver 1993], que son, en realidad, grafos acíclicos de decisión que posibilitan que distintas ramas de un árbol de decisión converjan. Los sistemas de aprendizaje de grafos permiten, en general, obtener modelos más compactos y compartir ejemplos por distintas vías, lo que puede tener efectos beneficiosos para evitar el sobreajuste y para ahorrar memoria. Los criterios de partición y de construcción deben redefinirse (por ejemplo [Oliver 1993] utiliza el principio MML (*Minimum Message Length*)).

La hibridación y combinación de características anteriores (incluyendo también los sistemas de reglas) ha sido creciente en la última década y sería imposible dar cuenta de ello en un reducido espacio. Baste como ejemplo que ya en 1992 [Buntine 1992] presentó el sistema IND que incorpora características del C4.5, del CART, de métodos bayesianos y codificación mínima, así como posibilidad de construir árboles o grafos.

11.7 Adaptación para grandes volúmenes de datos

Los algoritmos de aprendizaje de decisión, debido a su carácter voraz y a su estructura de divide y vencerás, se comportan especialmente bien con grandes volúmenes de datos, ya sean de gran dimensionalidad (muchos atributos) o de gran cardinalidad (muchos ejemplos). Este buen comportamiento (demostrado teórica y experimentalmente) muchas veces se basa en asumir que los datos caben en memoria, cosa que en muchas aplicaciones de minería de datos no es cierta. Si los datos no caben en memoria, el rendimiento de los algoritmos se degradará por el constante trasvase de información de disco a memoria.

En la última década se han dedicado esfuerzos a modificar los algoritmos para que sean más eficientes en el contexto de grandes tablas, intentando minimizar los barridos sobre las tablas y el uso de accesos a disco. En general, los algoritmos, denominados

“algoritmos de aprendizaje de árboles de decisión escalables”, se rediseñan teniendo en cuenta varios requisitos. En primer lugar, no han de requerir que los datos quepan en memoria. En segundo lugar, las comprobaciones de consistencia de las condiciones se deben hacer eficientemente, utilizando índices, con el objetivo de agilizar los barridos de los datos.

Por ejemplo, las condiciones sobre índices son preferibles sobre aquellas que no permiten indización. Por ejemplo la partición $\{ x \leq 3, x > 3 \}$ es mucho más eficiente de comprobar que la partición $\{ x \leq y, x > y \}$ para la cual no se pueden definir índices.

Una aproximación más ambiciosa es la basada en “cuentas de correlación”. Las cuentas de correlación son una especie de *precómpagos* que ayudan a que la evaluación posterior de las particiones sea más eficiente. Un ejemplo de cuenta de correlación se muestra en la Figura 11.11.

Variable a predecir

Attr-Val	Class ₁	Class ₂	...	Class _K
A _i =a _{i1}	Count _{i1,1}	Count _{i1,2}	...	Count _{i1,k}
A _i =a _{i2}	Count _{i2,1}	Count _{i2,2}	...	Count _{i2,k}
...
A _i =a _{ir}	Count _{ir,1}	Count _{ir,2}	...	Count _{ir,k}

Una tabla para cada atributo

Figura 11.11. Cuenta de correlación de una tabla.

La ventaja es que estas tablas suelen caber en memoria y se utilizan para decidir en cada nodo cuál es la mejor partición. Tal y como se ilustra en la Figura 11.11 sólo sería válida para atributos nominales y particiones del estilo $\{ x = a, x = b, \dots \}$, aunque se pueden contemplar extensiones o aproximaciones para los valores numéricos. El mayor problema es que para crearlas se deben utilizar consultas SQL costosas y empieza a ser beneficioso para árboles grandes o cuando se quieren realizar varios árboles de decisión con distintos atributos sobre la misma tabla.

Una idea más ingeniosa y que puede utilizarse en algunos casos es si se divide verticalmente la tabla. Se dividirá en tantas “subtablas” como atributos predictores haya en la tabla original. Cada subtabla contendrá el identificador del ejemplo, un atributo y la clase. Esta operación se muestra en la Figura 11.12.

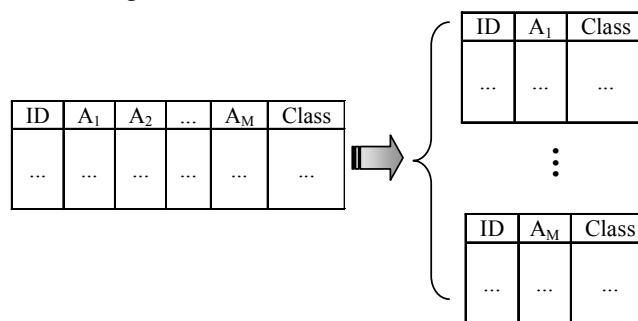


Figura 11.12. Dividiendo una tabla de forma vertical para acelerar las particiones.

Con esto, aunque el espacio en disco aumentará sensiblemente, se consigue que las evaluaciones de las particiones sobre un atributo se realicen más eficientemente, ya que cada una de las subtablas es más pequeña.

Existen muchísimas otras variaciones y modificaciones sobre árboles de decisión considerando grandes volúmenes de datos y gestión de memoria secundaria: paralelización, cambios de filas por columnas, etc. Ejemplos de algoritmos de aprendizaje de árboles de decisión que utilizan estas y otras técnicas son SLIQ [Mehta et al. 1996] y SPRINT [Shafer et al. 1996].

11.8 Sistemas, aplicabilidad y recomendaciones de uso

Basándose en diferentes particiones, en un criterio de partición y otras extensiones, han aparecido numerosos algoritmos o sistemas de aprendizaje de árboles de decisión. Enumeremos algunos de los más populares:

- CART [Breiman et al. 1984] y derivados: son métodos “divide y vencerás” que construyen árboles binarios y se basan en el criterio de partición GINI y que sirve tanto para clasificación como para regresión. La poda se basa en una estimación de la complejidad del error (“error-complexity”). Generalmente se pueden encontrar en paquetes de minería de datos con el nombre C&RT.
- ID3 [Quinlan 1983][Quinlan 1986], C4.5 [Quinlan 1993] y derivados (Assistant [Cestnik et al. 1987]): son métodos “divide y vencerás” y están basados en criterios de partición derivados de la ganancia (GainRatio). Tienen poda basada en reglas u otros mecanismos más sofisticados. Contiene métodos de colapsado de ramas y muchas otras mejoras. Existe un libro que describe concienzudamente la implementación y características del C4.5 [Quinlan 1993], la implementación se puede descargar gratuitamente o se encuentra en numerosas librerías. Por ejemplo, una versión más avanzada, la J4.8, se distribuye con la librería WEKA. Existe, además, una versión comercial presumiblemente mejor que las anteriores denominada C5 (See5) y comercializada por el propio Quinlan directamente (RuleQuest) o a través de paquetes de minería de datos (Clementine).
- IND [Buntine 1992], LMDT [Brodley & Utgoff 1995] y otros sistemas híbridos: incorporan características de varios sistemas o añaden otras técnicas de aprendizaje en la construcción de árboles de decisión: regresión lineal, perceptrones, etc.
- AQ [Michalski & Larson 1983; Michalski et al. 1986], CN2 [Clark & Niblett 1989; Clark & Boswell 1991] y derivados: son métodos por “cobertura” y se basan en métodos de búsqueda más elaborados (por ejemplo *beam search*). Se pueden encontrar en paquetes de minería de datos con el nombre de “RULES” (por ejemplo en el Clementine).
- SLIQ [Mehta et al. 1996] y SPRINT [Shafer et al. 1996]: modificaciones de árboles de decisión clásicos para conseguir escalabilidad para grandes volúmenes de datos, paralelización, etc.

Como dijimos en la introducción de este capítulo, toda esta variabilidad es consecuencia del éxito de los árboles de decisión y los sistemas de reglas en el aprendizaje automático y muy especialmente en la minería de datos.

Las ventajas principales de las técnicas vistas en este capítulo son las siguientes:

- Son aplicables a varias tareas de minería de datos: clasificación, regresión, agrupamiento y estimación de probabilidades.

- Tratan con atributos numéricos (continuos) y nominales (discretos).
- Muchos de ellos son eficientes y existen variantes escalables a grandes volúmenes de datos (tanto para muchos atributos como para muchos ejemplos).
- Son fáciles de usar.
- Son tolerantes al ruido, a atributos no significativos y a valores faltantes.
- Tienen una representación simbólica y se pueden desplegar de mayor a menor detalle (hasta un nivel de profundidad), con lo que generalmente aportan alta inteligibilidad.
- Existen, como hemos visto, multitud de sistemas, muchos de ellos gratuitos.

Lógicamente, también tienen algunas desventajas. En general, no son tan precisos como otros métodos, como pueden ser las redes neuronales o las máquinas de vectores de soporte. Además, son *débiles* (se conocen como *weak learners*), en el sentido de que, debido a su carácter voraz, son bastante dependientes de la muestra de ejemplos; dos muestras distintas sobre la misma distribución pueden dar dos árboles bastante diferentes. No obstante, es precisamente esta “debilidad” o “variabilidad” la que permite que los árboles de decisión sean frecuentemente utilizados con técnicas de combinación, como el *boosting* y el *bagging*. Este aspecto será tratado en profundidad en el Capítulo 18.

Por último, vamos a ilustrar el uso sencillo de un algoritmo de aprendizaje de árboles de decisión, el sistema C5.0. Para ello, vamos a utilizar su implementación en el sistema SPSS Clementine. El problema que vamos a tratar es el “*Wisconsin Diagnostic Breast Cancer*”, que se encuentra descrito en mayor detalle en el Apéndice B. Este conjunto de datos recoge 569 mediciones de las características de los núcleos de células mamarias (biopsia) según las imágenes tomadas de las muestras de pacientes susceptibles de padecer cáncer de mama. Todas las instancias están etiquetadas con los valores de diagnóstico “M” o “B”, indicando que finalmente el tumor fue “Maligno” o “Benigno”. El objetivo es realizar un modelo que nos permita predecir para futuras pacientes si el tumor es maligno o benigno según este conjunto de ejemplos anteriores etiquetados manualmente.

A parte del identificador de la paciente (que ignoraremos) existen 30 atributos (además de la clase). Estos 30 atributos representan tres estadísticas (la media, el error esperado y el peor valor) sobre diez características. Las primeras diez características son *RadioM*, *TexturaM*, *PerímetroM*, *ÁreaM*, *SuavidadM*, *CompacidadM*, *ConcavidadM*, *NumPuntosConcM*, *SimetríaM*, *DimFractalM* para las medias. Los otros 20 atributos corresponden a los valores del error esperado y el peor valor y cambian la “M” por una “EE” y por una “P” respectivamente.

Utilizando la herramienta Clementine cargamos los datos y los separamos en 400 datos para entrenamiento y 169 para prueba (test). Si damos tipo a los atributos (30 de entrada, el diagnóstico de salida y el identificador se ignora) para indicar cuál es la clase y los atributos predictores, podemos ya utilizar un algoritmo, como por ejemplo el C5.0. Vamos a probar el algoritmo con algunas variaciones y, por tanto, generaremos varios modelos variando los valores por defecto. La estructura del proceso realizado para ello en Clementine se muestra en la Figura 11.13.

Como se puede observar en la figura, los modelos se construyen con los datos de entrenamiento (parte superior) y luego se evalúan (“análisis”) con los datos de test (parte inferior).

El primer modelo generado (con el nombre “Diagnosis”) se ha generado utilizando los valores por defecto del C5.0 según la distribución del Clementine. El segundo modelo generado (con el nombre “DiagnosisRuleSet”) se ha construido utilizando la opción “generar reglas” del C5.0.

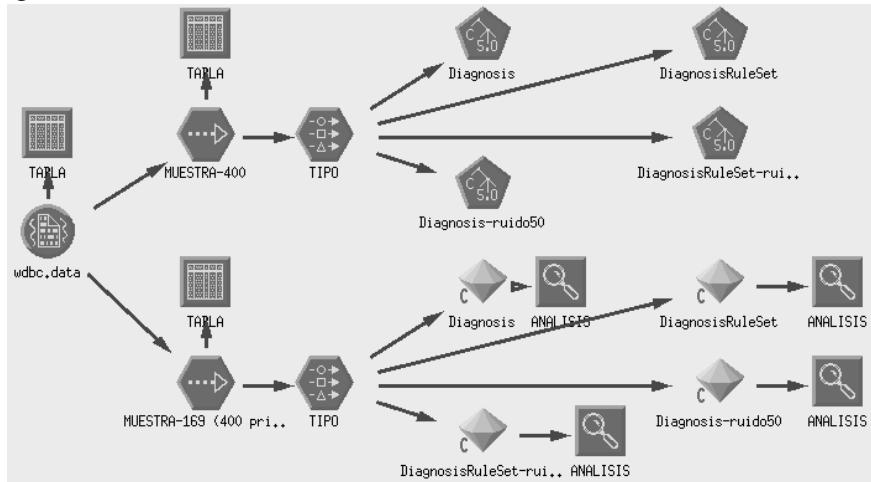


Figura 11.13. Aprendiendo y evaluando árboles de decisión en Clementine.

En la tabla siguiente se puede observar la diferencia de los modelos generados con ambas variantes.

	Árbol	Reglas
Modelo	PerímetroP <= 105 NumPuntosConcP <= 0.133 (209.0, 0.981) -> B NumPuntosConcP > 0.133 TexturaP <= 23.41 (5.0, 1.0) -> B TexturaP > 23.41 (11.0, 0.909) -> M PerímetroP > 105 PerímetroP <= 114.3 TexturaM <= 19.67 RadioM <= 14.11 (3.0, 1.0) -> M RadioM > 14.11 (14.0, 1.0) -> B TexturaM > 19.67 (18.0, 1.0) -> M PerímetroP > 114.3 ConcavidadM <= 0.062 CompacidadM <= 0.071 (6.0, 1.0) -> M CompacidadM > 0.071 (2.0, 1.0) -> B ConcavidadM > 0.062 (132.0, 1.0) -> M	Rules for M: #1 for M: if TexturaM > 19.67 and PerímetroP > 105 then -> M (109, 0.991) #2 for M: if TexturaP > 23.41 and NumPuntosConcP > 0.133 then -> M (131, 0.97) #3 for M: if PerímetroP > 105 then -> M (175, 0.904) Rules for B: #1 for B: if TexturaP <= 23.41 and PerímetroP <= 105 then -> B (136, 0.986) #2 for B: if PerímetroP <= 105 and NumPuntosConcP <= 0.133 then -> B (209, 0.976) #3 for B: if CompacidadM > 0.071 and ConcavidadM <= 0.062 then -> B (69, 0.958) #4 for B: if RadioM > 14.11 and TexturaM <= 19.67 and PerímetroP <= 114.3 then -> B (19, 0.952) Default : -> B
Precisión	92,31% (156 aciertos, 13 fallos)	92,90% (157 aciertos, 12 fallos)

Los resultados son bastante buenos en ambos casos (92,31 por ciento y 92,90 por ciento). La representación y tamaño en uno y otro caso son similares (alrededor de ocho reglas). Son pocas reglas y nos pueden dar bastante información (el perímetro peor y las texturas medias y peores parecen ser los atributos más relevantes). Los resultados con el algoritmo CART (C&R según la distribución del Clementine) son ligeramente peores y no se muestran aquí.

No obstante, podemos intentar mejorar los resultados variando las opciones de “ruido esperado” del C5.0. Si aumentamos el ruido esperado al 50 por ciento, generamos dos nuevos modelos, con nombres “Diagnosis-ruido50” y “DiagnosisRuleSet-ruido50”.

Los resultados de estos nuevos modelos (que resultan ser equivalentes) se muestran en la siguiente tabla:

	Árbol	Reglas
Modelo	PerímetroP <= 105 (225.0, 0.938) -> B PerímetroP > 105 PerímetroP <= 120.3 TexturaM <= 19.67 (25.0, 0.64) -> B TexturaM > 19.67 (26.0, 1.0) -> M PerímetroP > 120.3 (124.0, 1.0) -> M	Rules for M: #1 for M: if PerímetroP > 120.3 then -> M (124, 0.992) #2 for M: if TexturaM > 19.67 and PerímetroP > 105 then -> M (109, 0.991) Rules for B: #1 for B: if PerímetroP <= 105 then -> B (225, 0.934) #2 for B: if TexturaM <= 19.67 and PerímetroP <= 120.3 then -> B (200, 0.921) Default : -> B
Precisión	94,08% (159 aciertos, 10 fallos)	94,08% (159 aciertos, 10 fallos)

Los resultados son ahora ligeramente más precisos pero, lo que es más importante, son más simples y, por tanto, más comprensibles. Ésta es una de las razones fundamentales para utilizar árboles de decisión o sistemas de reglas.

Capítulo 12

MÉTODOS RELACIONALES Y ESTRUCTURALES

En este capítulo presentamos los métodos de minería de datos relacionales. La idea principal que subyace en los mismos es usar un lenguaje de representación *relacional*, mucho más potente expresivamente hablando que los tradicionales lenguajes de representación típicos de la mayoría de los métodos de minería de datos. Los métodos relacionales nos permiten descubrir patrones más complejos, haciendo uso de la estructura de los propios datos y las relaciones entre ellos, algo por ejemplo muy importante en aplicaciones como la bioinformática o la farmacología. De hecho, otras aproximaciones no declarativas, como la minería de grafos, tienen como objetivo analizar datos donde cada ejemplo tiene una estructura compleja (como, por ejemplo, una molécula). Estas aproximaciones las denominamos *estructurales*.

Daremos un tratamiento especial a la Programación Lógica Inductiva (ILP), a la que dedicaremos buena parte del capítulo, por ser esta área del aprendizaje automático la que tradicionalmente se ha ocupado al aprendizaje en notación relacional. Como otros métodos de minería de datos, los métodos ILP se orientaron inicialmente a resolver problemas de pequeño tamaño y sólo recientemente se han empleado para su aplicación a la minería de datos, dando lugar al término Minería de Datos Relacional.

12.1 Introducción

Tal y como se ha comentado en los primeros capítulos del libro, el objetivo de los métodos de minería de datos es poder extraer conocimiento de grandes volúmenes de datos almacenados, generalmente, en bases de datos. Dado que muchas técnicas de aprendizaje automático funcionan relativamente bien para tratar problemas de gran tamaño, hemos visto que muchas herramientas de minería de datos se basan en ellas. La mayoría de estos algoritmos (árboles de decisión, redes neuronales, redes bayesianas, reglas de asociación...)

trabajan con los datos en formato atributo-valor y, por lo tanto, son capaces de encontrar patrones en bases de datos cuya información se almacena en una única tabla (o relación que llamamos “vista minable”). En estos casos, cada objeto se describe a través de un conjunto fijo de atributos cada uno de los cuales puede solamente tener un valor simple (no estructurado).

Ésta no es una restricción de las bases de datos relacionales, sino que es una limitación de los algoritmos de aprendizaje automático. Para representar objetos más complejos y con estructura (estructurales) se pueden emplear bases de datos relacionales que contengan más de una tabla, de tal forma que un objeto se describe en ellas mediante varios registros de varias tablas (por ejemplo, los datos de los empleados de una empresa se suelen almacenar en distintas tablas: la de información personal, la del departamento/sección, la de proyectos, la de nóminas, etc.). Pero las técnicas atributo-valor no son capaces de descubrir patrones complejos que involucren objetos de varias tablas o que usen la información sobre la estructura de los datos.

Veamos la diferencia entre ambos tipos de representaciones con un ejemplo. Consideremos una universidad que mantiene una base de datos con las siguientes relaciones:

- *curso(Id_curso, profesor, inicio, duración, aula)*
- *matrícula(Id_estudiante, Id_curso)*

En la Figura 12.1 se muestran algunos registros de esta base de datos. Además, el propio esquema contiene metainformación, es decir, definiciones de claves primarias y claves ajena, como la que podría existir entre *Id_curso* de la tabla *matrícula* e *Id_curso* de la tabla *curso*.

curso					
Id_curso	profesor	tipo	inicio	duración	aula
c 01	“J. Campos”	tutorial	octubre	16	1.0
c 02	“M. Vidal”	tutorial	noviembre	20	2.1
c 03	“E. Ruiz”	seminario	febrero	12	1.1
...

matrícula	
Id_estudiante	Id_curso
2345	c 01
2345	c 03
1845	c 01
3102	c 02
...	...

Figura 12.1. Tablas (en formato atributo-valor) de las relaciones *curso* y *matrícula*.

Si usamos un sistema de aprendizaje atributo-valor podríamos descubrir algunos patrones, como por ejemplo qué cursos son seminarios

SI duración(Curso)<15 **ENTONCES** seminario(Curso)

pero no podríamos obtener otros patrones más complejos que involucren registros de las dos tablas, como por ejemplo cuándo un estudiante es alumno de un profesor. Para poder dar respuesta a esta cuestión necesitaríamos hacer uso de ambas relaciones.

La causa de esta limitación es que los lenguajes usados para representar los ejemplos obligan a expresarlos en una única tabla, como filas de una estructura fija y rígida. Se trata, por lo tanto, de una restricción bastante fuerte ya que excluye la posibilidad de expresar conocimiento estructural o relacional. Una forma de extender el aprendizaje atributo-valor es cambiar el lenguaje de representación y usar lógica de primer orden (o alguna variante de la misma). Ésta es la idea que subyace en la minería de datos relacional (*Relational Data Mining*, RDM [Džeroski & Lavrač 2001]), a veces también llamada multi-relacional para enfatizar el hecho de que puede tratar con varias tablas o relaciones.

Los métodos de la RDM se han aplicado con éxito en numerosas áreas de aplicación, tales como los negocios, la bioinformática (incluyendo el análisis del genoma), la farmacología y la minería web.

En la minería de datos relacional podemos expresar tanto condiciones sobre los valores de los atributos como sentencias sobre la estructura de la información o que relacionen objetos de varias tablas. En el ejemplo anterior, la relación ser alumno de un profesor se expresaría mediante la regla:

SI matricula(Estudiante,Curso) **Y** curso(Curso,Profesor,Tipo,Inicio,Duración,Aula)
ENTONCES alumno(Estudiante,Profesor)

Aunque los métodos más estudiados para descubrir patrones relacionales se encuadran dentro del área de la programación lógica inductiva (*Inductive Logic Programming*, ILP), también podemos encontrar otras típicas técnicas de minería de datos que trabajaban con una representación de una única tabla y que se han extendido para trabajar con varias tablas, como los árboles de decisión relacionales, las reglas de asociación relacionales, los modelos probabilísticos relacionales o los métodos para la predicción y el agrupamiento basados en distancias (definiendo medidas de distancias entre ejemplos/instancias representados en lógica relacional).

Cuando se usan bases de datos relacionales que contienen múltiples relaciones, no siempre es estrictamente necesario usar un algoritmo de RDM. Existen formas de amoldar una base de datos en un formato de una única tabla, de tal forma que se pueden aplicar algoritmos tradicionales atributo-valor [Džeroski 1996]. Una forma de hacer esto es crear una relación universal juntando (concatenando) todas las tablas en una tabla, usando las claves ajenas, por tanto desnormalizando completamente el esquema. La relación universal resultante de este proceso puede ser extremadamente grande y muy difícil de manejar en la práctica. Siguiendo con el ejemplo anterior podríamos añadir a la relación *matrícula* los atributos *profesor*, *tipo*, *inicio*, *duración* y *aula*, obteniendo así una nueva relación que llamamos *matrícula_universal*.

matrícula_universal						
Id_estudiante	Id_curso	profesor	tipo	inicio	duración	aula
2345	c 01	“J. Campos”	tutorial	octubre	16	1.0
2345	c 03	“E. Ruiz”	seminario	febrero	12	1.1
1845	c 01	“J. Campos”	tutorial	octubre	16	1.0
3102	c 02	“M. Vidal”	tutorial	noviembre	20	2.1
...

Tabla 12.1. Tabla universal obtenida juntando las tablas de la Figura 12.1.

La segunda forma de transformar una base de datos a una tabla consiste en crear nuevos atributos en la tabla final que sean agregaciones o resúmenes de información presente en las otras tablas. Este método se usa en el sistema LINUS [Lavrač & Džeroski 1994] entre otros. Siguiendo con el mismo ejemplo, podríamos añadir a la tabla *curso* un atributo con el número de alumnos matriculados.

curso					
profesor	tipo	inicio	duración	aula	n_alumnos
“J. Campos”	tutorial	octubre	16	1.0	10
“M. Vidal”	tutorial	noviembre	20	2.1	21
“E. Ruiz”	seminario	febrero	12	1.1	16
...

Tabla 12.2. Tabla de la relación *curso* ampliada con información agregada de la tabla *matrícula*.

Estas dos aproximaciones, así como aproximaciones híbridas, se comentaron en los capítulos 3 y 5, que parten de una base de datos y obtienen una vista minable. Los principales inconvenientes de estas aproximaciones son los siguientes: la tabla final tiene muchos atributos que contienen datos repetidos, se pierde la información sobre cómo estaban los datos inicialmente estructurados (la información del esquema original, restricciones de integridad, etc.), y la creación de nuevos atributos informativos, útiles y comprensibles puede requerir una cantidad sustancial de conocimiento sobre el dominio. En definitiva, ninguna de estas aproximaciones es muy atractiva desde el punto de vista de la eficiencia, lo que evidencia las ventajas de trabajar directamente con varias relaciones.

Dentro de los métodos de RDM destacamos la Programación Lógica Inductiva (ILP) por ser esta área la que tradicionalmente se ha ocupado al aprendizaje en notación relacional. La ILP, término acuñado por Muggleton [Muggleton 1991], se ocupa principalmente del problema de aprender programas lógicos desde ejemplos. Por lo tanto, sus fundamentos son por un lado la lógica y por otro el aprendizaje. En lo que sigue revisaremos primero las nociones básicas sobre la programación lógica (LP) y su relación con las bases de datos, para luego pasar a describir los principales conceptos de la ILP (el paradigma de aprendizaje relacional más importante) y su aplicación a la minería de datos. Asimismo, también presentaremos brevemente otros métodos relacionales y declarativos. Al final del capítulo se incluye una breve reseña de los principales sistemas ILP y otras referencias que consideramos de interés.

12.2 Programación lógica y bases de datos

En esta sección presentamos brevemente los fundamentos de la programación lógica y su conexión con las bases de datos. Para una revisión más profunda de este paradigma de programación recomendamos al lector consultar [Lloyd 1987] y [Apt 1997].

La programación lógica (LP) es un paradigma de programación que surgió en los años 70 sobre la idea de Colmerauer y Kowalski de que la lógica de primer orden, o un subconjunto de ella, podía usarse como lenguaje de programación. La principal característica de este paradigma es ser **declarativo**, es decir, en él están completamente separados y diferenciados *qué* es lo que se debe resolver (la componente lógica) del *cómo* debe hacerlo (la componente de control). Esto hace que el programador sólo se tenga que

preocupar de la parte lógica, dejando que sea el propio sistema de programación lógica el que se ocupe del control³⁷.

La programación lógica tiene como fundamento teórico la lógica de las cláusulas de Horn, un subconjunto del cálculo de predicados de primer orden.

Una cláusula es una expresión de la forma

$$p(X_1, \dots, X_n) \leftarrow L_1, \dots, L_m$$

donde p es un símbolo de predicado de aridad n , los L_i son literales, es decir expresiones positivas de la forma $q_i(Y_1, \dots, Y_m)$ (también llamadas átomos) o expresiones negativas de la forma $not\ q_i(Y_1, \dots, Y_m)$, y el símbolo “,” representa la conjunción lógica “ \wedge ”. Una cláusula puede verse como una regla que expresa la implicación lógica en la que $p(X_1, \dots, X_n)$ es el consecuente de la regla, lo que se denomina cabeza de la cláusula, y L_1, \dots, L_m es el antecedente de la regla, también llamado cuerpo de la cláusula. Cuando la cláusula tiene un único átomo en la cabeza se llama definida. Las cláusulas definidas reciben el nombre de hechos, cuando el cuerpo es vacío, o reglas, cuando el cuerpo es no vacío. Si la cláusula no contiene variables se dice que es básica (en inglés, *ground*).

Un programa lógico es un conjunto de cláusulas definidas. Un conjunto de cláusulas con el mismo símbolo de predicado p en la cabeza forman la definición de p . Un predicado puede definirse extensionalmente como un conjunto de hechos básicos o intensionalmente como un conjunto de reglas. Por ejemplo, el siguiente conjunto de cláusulas (Figura 12.2) define los predicados *mujer*, *hombre* y *padre* extensionalmente y el predicado *hermanas* intensionalmente, y en donde hemos usado la notación de Prolog (el lenguaje más representativo del paradigma de programación lógica): el símbolo “ $:$ ” representa la implicación lógica (“ \leftarrow ”), las variables comienzan por mayúscula, los términos entre comillas dobles, los nombres de los predicados comienzan por minúscula y todas las cláusulas terminan con el carácter “ $.$ ”.

```

mujer ("Ana") .
mujer ("Elena") .
hombre ("Juan") .
padre ("Luis", "Ana") .
padre ("Luis", "Elena") .
padre ("Luis", "Juan") .

hermanas (X, Y) :- padre (Z, X) , padre (Z, Y) , mujer (X) , mujer (Y) .

```

Figura 12.2. Programa lógico sobre relaciones familiares.

La regla de la figura anterior establece que son hermanas porque ambas tienen el mismo padre y son mujeres. El mecanismo de razonamiento usado en la programación lógica es la deducción, usando como regla de inferencia la resolución SLD, que permite inferir fórmulas aplicando las reglas del programa a casos particulares. Por ejemplo, del programa de la Figura 12.2 podríamos deducir *hermanas*("Ana", "Elena") a partir de la regla

³⁷ Esto era, en realidad, un ideal expresado por Kowalski, ya que la mayoría de sistemas lógicos permiten al programador actuar directamente sobre el control con el objeto de mejorar la eficiencia de estos sistemas.

hermanas(X,Y):-padre(Z,X), padre(Z,Y), mujer(X), mujer(Y) y de los hechos *mujer("Ana")*, *mujer("Elena")*, *padre("Luis","Ana")* y *padre("Luis","Elena")*.

Aunque la programación lógica se ha utilizado profusamente como base para lenguajes de programación y muy especialmente para representación de conocimiento, nos interesa aquí principalmente su correspondencia con las bases de datos relacionales. Existe una interpretación de la programación lógica en términos de las bases de datos. La Tabla 12.3 muestra la correspondencia entre ambas terminologías.

BASES DE DATOS	PROGRAMACIÓN LÓGICA
nombre de relación o tabla p	símbolo de predicado p
campo o atributo de una relación o tabla p	argumento de un predicado p
fila o tupla en una tabla relacional p tupla $\langle a_1, \dots, a_n \rangle$	hecho básico de un predicado p $p(a_1, \dots, a_n)$
una tabla relacional p (conjunto de tuplas)	definición extensional de un predicado p (conjunto de hechos básicos)
una relación p definida como una vista	definición intensional de un predicado p

Tabla 12.3. Correspondencia entre las bases de datos y la programación lógica.

Así, una base de datos relacional no es más que un programa lógico que sólo consta de hechos básicos. Con esta notación, la base de datos de la Figura 12.1 se expresa como el programa lógico:

```

curso(c01,"J.Campos",tutorial,octubre,16,1.0).
curso(c02,"M.Vidal",tutorial,noviembre,20,2.1).
curso(c03,"E.Ruiz",seminario,febrero,12,1.1).
...
matrícula(2345,c01).
matrícula(2345,c03).
matrícula(1845,c01).
matrícula(3102,c02).
...

```

Pero además, la programación lógica se ha usado para implementar las bases de datos deductivas [Lloyd 1987], ya que estos sistemas se fundamentan teóricamente en la lógica de primer orden. Una base de datos deductiva no es más que un programa lógico en el que algunas relaciones se definen extensionalmente y otras intensionalmente.

Ya que una base de datos se puede entender e implementar como un programa lógico, podemos también pensar en usar la lógica de primer orden para analizar los datos y extraer conocimiento en forma de programas lógicos. De esto se ocupa la programación lógica inductiva que se presenta a continuación.

12.3 Programación lógica inductiva

En esta sección se revisan las nociones más importantes sobre la programación lógica inductiva. Para cualquier concepto no incluido sugerimos al lector consultar [Muggleton 1992; Bergadano & Gunetti 1995] o [Nienhuys-Cheng & Wolf 1997].

En 1990, Stephen Muggleton introdujo el nombre de programación lógica inductiva (*Inductive Logic Programming*, ILP), y definió esta área como la intersección del aprendizaje

inductivo y la programación lógica [Muggleton 1990; Muggleton 1991]. Como el aprendizaje inductivo, la ILP tiene por objetivo desarrollar técnicas y herramientas para inducir hipótesis desde observaciones (ejemplos) o sintetizar nuevo conocimiento desde la experiencia, pero usando como lenguaje de expresión la programación lógica. Desde un punto de vista teórico, la ILP puede considerarse el proceso dual a la LP: si la LP se centra en la deducción de hechos desde un conjunto de cláusulas (un programa lógico), la ILP describe el proceso de inducir programas lógicos desde hechos lógicos.

En su formulación más simple, la tarea de la ILP consiste en aprender la definición de un predicado a partir de hechos que son ciertos para este predicado (ejemplos positivos), hechos que son falsos para este predicado (ejemplos negativos) y la definición de otros predicados auxiliares que se pueden usar en el proceso de aprendizaje (conocimiento previo o de base).

Formalmente, un problema de ILP se define como la inferencia de una teoría P (un programa lógico) desde una evidencia E (la cual puede estar compuesta de ejemplos positivos E^+ y, opcionalmente, de ejemplos negativos E^-) posiblemente usando una teoría de conocimiento de base B (otro programa lógico), tal que se cumplan las siguientes condiciones

1. $B \not\models E^+$ (necesidad a priori)
2. $\forall e^- \in E^- : B \not\models e^-$ (satisfacibilidad a priori)
3. $B \cup P \models E^+$ (suficiencia a posteriori)
4. $\forall e^- \in E^- : B \cup P \not\models e^-$ (satisfacibilidad a posteriori)

El símbolo \models representa la consecuencia lógica y $\not\models$ representa que no se cumple la consecuencia lógica. Por ejemplo, cuando dado un ejemplo o hecho e , se cumple que $H \models e$, decimos que H cubre e . La condición 3 establece que la hipótesis P es completa y la 4 que es consistente. Si una hipótesis cumple ambas propiedades se dice que es correcta.

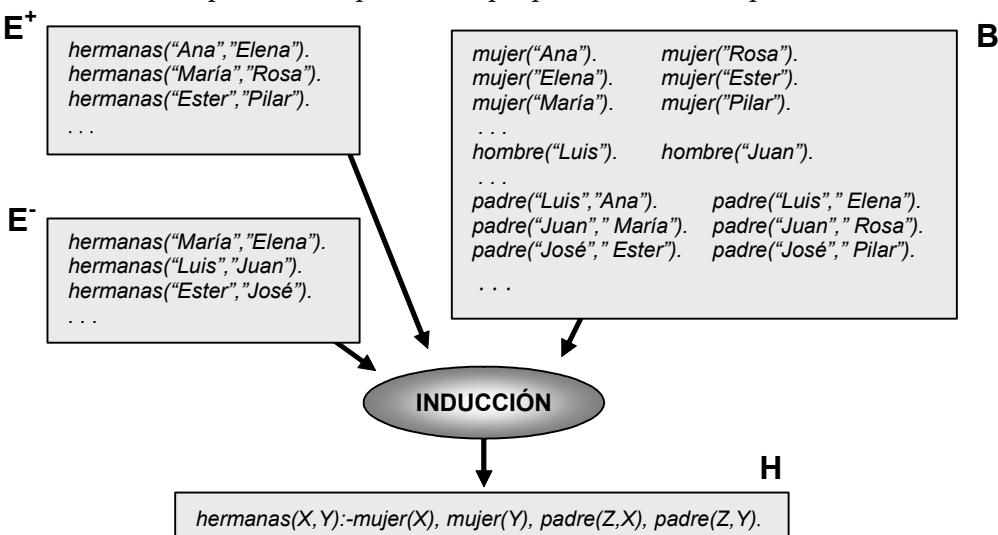


Figura 12.3. Inducción de la relación hermanas.

Inicialmente, la ILP se aplicó principalmente a la síntesis de programas (es decir, el aprendizaje de programas lógicos para asistir a los programadores). Así, desde la

perspectiva de la ILP y siguiendo con el ejemplo de la Figura 12.2, el objetivo sería inferir un programa lógico que defina el predicado *hermanas*. Para ello partimos de una evidencia positiva E^+ formada por hechos básicos sobre el predicado *hermanas*, de un conjunto de hechos básicos sobre el predicado *hermanas* que el programa inferido no debe satisfacer (la evidencia negativa E^-) y de una teoría de base que define los predicados *mujer*, *hombre* y *padre*, los cuales pueden usarse en el aprendizaje. El proceso se ilustra gráficamente en la Figura 12.3.

Ahora bien, la hipótesis inducida que se muestra en la Figura 12.3 es sólo una de las muchas hipótesis que podríamos haber generado. Por ejemplo, "*hermanas(X,Y):-mujer(X), mujer(Y), hombre(Z), padre(Z,X), padre(Z,Y).*" también sería una hipótesis completa y consistente (ya que cubre todos los ejemplos positivos y ninguno de los negativos). Entonces, si existe más de una hipótesis correcta, para encontrar la que consideremos satisfactoria (de acuerdo a ciertos criterios preestablecidos) deberemos efectuar una búsqueda entre todas las hipótesis posibles (lo que se ha venido en llamar el espacio de las hipótesis).

Al hilo de este razonamiento surge la cuestión de cómo realizar la búsqueda de la hipótesis en el espacio de hipótesis. En general, existen dos formas de proceder: los métodos *top-down* que parten de la hipótesis más general y van restringiéndola mediante operaciones de especialización, y los métodos *bottom-up* que parten de la hipótesis más específica y van generalizándola mediante operadores de generalización. Debemos mencionar que un sistema se clasifica en uno de estos dos grupos atendiendo a cuál es el sentido general de la búsqueda, independientemente de que tanto los sistemas *top-down* como los *bottom-up* puedan realizar localmente pasos de generalización o de especialización (respectivamente) para ajustarse a los ejemplos.

No obstante, aun definiendo un sentido, no es posible efectuar una búsqueda exhaustiva a base de probar todas las hipótesis del espacio de hipótesis, ya que esto resultaría computacionalmente caro e inefficiente. Una forma de resolver este problema es reducir el espacio de hipótesis imponiendo condiciones sobre el lenguaje para que sólo aquellas hipótesis que las cumplen sean generadas, o bien podando el espacio de búsqueda usando la evidencia. Así, si una hipótesis no cubre un ejemplo positivo ninguna de sus especializaciones lo cubrirá. De igual forma, si una hipótesis cubre un ejemplo negativo entonces todas sus generalizaciones también lo cubrirán. En los dos casos anteriores podemos usar esta información para podar algunas de las hipótesis del espacio. Otra posibilidad para que la búsqueda sea eficiente consiste en reducir la misma considerando que el conjunto de cláusulas está de algún modo estructurado. Esta estructura puede conseguirse imponiendo un orden de generalidad entre las cláusulas: la θ -subsunción.

12.3.1 θ -subsunción

Para poder definir este orden necesitamos introducir primero los conceptos de sustitución e instancia. Una sustitución es un conjunto finito de pares $\{X_1/t_1, \dots, X_n/t_n\}$ donde X_i son variables distintas entre sí y t_i son términos tales que $t_i \neq X_i$. Intuitivamente, cada par X_i/t_i de una sustitución representa que la variable X_i se reemplaza por el término t_i . Por ejemplo, $\theta = \{X/f(a), Y/h(X)\}$ es una sustitución.

Las sustituciones pueden aplicarse a cualquier tipo de expresiones: términos, literales o diyunciones o conjunciones de literales (es decir, cláusulas). El efecto es reemplazar las variables de la expresión por términos de acuerdo a la sustitución. Esto es a lo que se llama instancia. Sea E una expresión y $\theta = \{X_1/t_1, \dots, X_n/t_n\}$ una sustitución. Entonces la instancia de E por θ , denotado por $E\theta$, se obtiene reemplazando simultáneamente en E cada ocurrencia de X_i por t_i . Por ejemplo, consideremos la expresión $E = p(X, Y, b)$ y la sustitución $\theta = \{X/f(a), Y/h(X)\}$. Entonces, $E\theta$ es la expresión $p(f(a), h(X), b)$.

Dadas dos cláusulas C y D decimos que C θ -subsume la cláusula D , y lo denotamos como $C \leq_\theta D$ (o simplemente $C \leq D$), si $C\theta \subseteq D$ (es decir, cada literal de $C\theta$ es un literal de D), donde θ es una sustitución. Por ejemplo, si consideramos las dos cláusulas del ejemplo anterior $C = \text{hermanas}(X, Y) :- \text{mujer}(X), \text{mujer}(Y), \text{hombre}(Z), \text{padre}(Z, X), \text{padre}(Z, Y)$ y $D = \text{hermanas}(X, Y) :- \text{mujer}(X), \text{mujer}(Y), \text{padre}(Z, X), \text{padre}(Z, Y)$, se cumple $D\theta \subseteq C$ con θ la sustitución vacía.

La θ -subsunción define una noción de generalidad entre las cláusulas: dadas dos cláusulas C y D , decimos que C es más general que D si $C \leq_\theta D$. Muchos de los operadores de especialización y generalización definidos por los métodos *top-down* y *bottom-up* se basan en esta noción.

12.3.2 Métodos *top-down*

En general, los métodos de ILP *top-down* inducen los programas creando las cláusulas de una en una, usando para ello la especialización. La forma más simple de actuar se muestra en el algoritmo de la Figura 12.4.

<p>MÉTODO <i>TopDown</i></p> <p>$H = \emptyset$</p> <p>MIENTRAS H no cubra algún ejemplo pos. HACER</p> <p style="padding-left: 20px;">$C = \text{generar_por_especialización}.$</p> <p style="padding-left: 20px;">eliminar los ejemplos pos. cubiertos por C</p> <p style="padding-left: 20px;">añadir C a H</p> <p>FIN MIENTRAS</p> <p>FIN MÉTODO</p>	<p>FUNCIÓN <i>generar_por_especialización</i></p> <p>seleccionar el predicado p a aprender</p> <p>REPETIR</p> <p style="padding-left: 20px;">$C \leftarrow p(x_1, x_2, \dots, x_n) :- \text{true}.$ // regla sin cuerpo</p> <p>MIENTRAS C cubra ejemplos neg. y pos. HACER</p> <p style="padding-left: 20px;">seleccionar un literal L</p> <p style="padding-left: 20px;">añadir L al antecedente de C</p> <p>FIN MIENTRAS</p> <p>HASTA C cumple algún ejemplo pos. o se han agotado todas las posibilidades para L</p> <p>DEVOLVER C // no cubre ningún neg. y algún pos.</p> <p>FIN FUNCIÓN</p>
---	--

Figura 12.4. Métodos *top-down*: algoritmo general.

Como puede verse, para obtener las cláusulas que conforman la definición del predicado a inducir, p , en cada iteración del bucle principal se crea una cláusula (función *generar_por_especialización*). Este proceso termina cuando la hipótesis H cubre todos los ejemplos positivos. Por su parte, la función *generar_por_especialización* muestra que para obtener una cláusula C de p se parte de la cláusula más general ($p(x_1, x_2, \dots, x_n) :- \text{true}.$), que cubre todos los ejemplos del predicado p (positivos y negativos) y se va especializando a base de añadir literales al cuerpo hasta que éste deje de cubrir los negativos y todavía cubra algún ejemplo positivo. Entonces esta cláusula se añade a la hipótesis actual H y se eliminan los ejemplos positivos que cubre. Los distintos sistemas difieren en cómo se selecciona el literal a añadir al cuerpo de la cláusula que se está induciendo y en cómo verifican que una cláusula cubre los ejemplos.

Ejemplos de sistemas de ILP que siguen este método son MIS (*Model Inference System*) [Shapiro 1983], FOIL [Quinlan 1990] y FILP [Bergadano & Gunetti 1993].

12.3.3 Métodos *bottom-up*

Los métodos de ILP *bottom-up* parten siempre de la hipótesis más específica (por ejemplo, el programa formado por los ejemplos positivos) y aplican operadores de generalización. Operadores clásicos de generalización empleados en el aprendizaje automático son: la eliminación de condiciones, la transformación de constantes en variables y la transformación de conjunciones en disyunciones. Sin embargo, la ILP ha desarrollado sus propios operadores de generalización invirtiendo las reglas deductivas de la LP, siendo los más importantes la generalización menos general y la resolución inversa.

- La generalización menos general de dos cláusulas (*least general generalization, lgg*) [Plotkin 1970; Plotkin 1971] es la cláusula simple más específica que generaliza ambas. Intuitivamente, el *lgg* se obtiene emparejando las dos cláusulas, respetando aquellas partes en las que coinciden y sustituyendo por variables las partes en las que no coinciden. Esta idea puede extenderse a conjuntos de cláusulas. Formalmente, una cláusula C es el *lgg* de un conjunto de cláusulas S si C es el *lgg* de cada cláusula de S , es decir, si $C \leq D$ para toda cláusula $D \in S$ y para cualquier otra cláusula E tal que $E \leq D$ se cumple $E \leq C$. Por ejemplo, el *lgg* de las cláusulas *hermanas*("Ana", "Elena") :- *mujer*("Ana"), *mujer*("Elena") y *hermanas*("María", "Rosa") :- *mujer*("María"), *mujer*("Rosa") es la cláusula *hermanas*(X, Y) :- *mujer*(X), *mujer*(Y). La noción de *lgg* puede ampliarse para tener en cuenta el conocimiento de base B . En este caso, se habla de generalización menos general relativa a B o simplemente *rlgg*. El sistema GOLEM [Muggleton & Feng 1990] es un ejemplo de sistema ILP *bottom-up* basado en *rlgg*.
- La resolución inversa fue definida en [Muggleton & Buntine 1988] como una herramienta para la inducción, descrita por dos operadores que invertían los pasos de resolución: el operador V y el operador W . Dados C_1 y R , el operador V encuentra la cláusula C_2 tal que R es una instancia de un resolvente de C_1 y C_2 . Por lo tanto, el V -operador generaliza $\{C_1, R\}$ a $\{C_1, C_2\}$. El W -operador combina dos operadores V y generaliza $\{R_1, R_2\}$ a $\{C_1, C_2, C_3\}$ tal que R_1 es una instancia de un resolvente de C_1 y C_2 y R_2 es una instancia de un resolvente de C_2 y C_3 . Además, el operador W es capaz de inventar predicados, es decir introducir nuevos predicados diferentes del predicado a inducir y no presentes ni en la evidencia ni en el conocimiento de base. Usando la resolución inversa Muggleton y Buntine implementaron un sistema *bottom-up* interactivo llamado Cigol (cuyo nombre es *Logic* al revés). Una evolución de esta idea es la implicación inversa (*inverse entailment*) que es la base del sistema PROGOL [Muggleton 1995].

12.3.4 Bias inductivos en ILP

El bias inductivo, como comentamos en la Sección 6.1, es cualquier información que influye en el aprendizaje inductivo desde ejemplos. Existen tres categorías de bias inductivo: bias del lenguaje, bias de búsqueda y criterios de parada. Todos ellos buscan un mejor comportamiento de los algoritmos de inducción, bien reduciendo el espacio de hipótesis

(bias del lenguaje), bien explorando parcialmente este espacio (bias de búsqueda) o bien determinando cuándo debe dejar de explorarse (para lo que se usan los criterios de parada). El bias es importante para la eficiencia del aprendizaje. Si hay muchos posibles programas a considerar, entonces podemos tardar mucho tiempo en encontrar uno que sea aceptable.

El bias del lenguaje es cualquier descripción del espacio de hipótesis que contiene todos los programas lógicos que el sistema puede aprender. Algunas formas sencillas de bias del lenguaje son, por ejemplo, restringir los programas inducidos a ser programas definidos o normales, a contener cláusulas sin funciones o cláusulas no recursivas, acotar el número de cláusulas en la teoría, acotar la profundidad o el número de literales o variables en una cláusula o usar sólo cláusulas permitidas. Uno de los marcos generales para especificar bias del lenguaje consiste en definir esquemas o plantillas de cláusulas (es decir, se trata de un esquema de segundo orden en el que se definen cláusulas genéricas en las que los símbolos de predicado se han reemplazado por variables). Por ejemplo, el sistema MOBAL [Morik et al. 1993] usa plantillas a las que denomina modelos de cláusulas. Otra forma de restringir el espacio de hipótesis consiste en dar información sobre el modo y los tipos de los argumentos de los predicados. Esto es lo que hacen sistemas como por ejemplo MIS y PROGOL. Otro tipo de bias de búsqueda es el que sugiere el propio esquema de la bases de datos, si el sistema puede aprovecharlo, que “conecta” unas tablas con otras.

El bias de búsqueda es cualquier información que le dice al sistema cómo debe buscar las hipótesis: qué programas deben ignorarse, qué programas y cláusulas deben considerarse primero y qué predicados deben ser añadidos o eliminados de los antecedentes de las cláusulas. En cuanto al método de búsqueda, éste puede ser una búsqueda sistemática (primero en profundidad, primero en amplitud, primero el mejor), usar una estrategia de búsqueda (divide y vencerás, *hill climbing*), o bien usar heurísticas de búsqueda (cobertura de los ejemplos positivos, exclusión de los ejemplos negativos, precisión, contenido de información...).

Se han usado un gran número de criterios de parada. Por ejemplo, se puede usar como criterio el minimizar el error cometido sobre los datos de entrenamiento (aunque éste es un criterio bastante pobre). Mucho mejor que éste es establecer que el algoritmo debe parar cuando la hipótesis tiene una calidad aceptable (medida por ejemplo en términos de precisión, coste de error en la clasificación...). En estos casos el usuario determina el umbral mínimo que debe superar una hipótesis para ser considerada aceptable.

12.3.5 ILP y recursividad

En el Capítulo 6 se comentó que cuando los ejemplos tenían estructura podía ser necesario utilizar *recursividad* para capturar el concepto a aprender. Dijimos que la recursividad se requería cuando la profundidad (el nivel) de las relaciones no se conocía a priori (objetos que contienen o se relacionan con un número variable de objetos). Uno de los ejemplos que vimos fue el de aprender el concepto de “ser familia de” (*relative*).

Un modelo resultante de un sistema ILP podría ser el de la Figura 6.10, que reproducimos a continuación:

```

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
relative(X,Y) :- ancestor(X,W), ancestor(Y,W).

```

Figura 12.5. *Modelo de gran expresividad representado mediante reglas lógicas.*

Aunque inicialmente se estudió la inducción de modelos recursivos para la síntesis de programas, los modelos recursivos son útiles para muchas otras aplicaciones. Por ejemplo, el concepto de accesibilidad de una página web a otra página web, o de un nodo de una red a otro, es un concepto recursivo.

Pese a su potencialidad, el aprendizaje de modelos recursivos es uno de los mayores retos de ILP (véase, por ejemplo [Flener 2002]). Aunque muchos de los sistemas de ILP son capaces de aprender modelos recursivos de ejemplos sencillos (como por ejemplo de ordenación de listas, aritméticos, etc.), no son capaces de abordar problemas más complejos que necesiten recursividad. Este campo sigue siendo un área de intenso estudio.

12.4 Programación lógica inductiva y minería de datos

Pensando en términos de minería de datos, el proceso ilustrado en la Figura 12.3 puede verse como un problema de clasificación [Džeroski 2003]. La hipótesis inducida H es una regla con respecto a la cual se puede clasificar la evidencia en una de estas dos clases: ser hermanas y no ser hermanas. De hecho, puesto que en lógica de primer orden el valor de verdad de un hecho básico es o bien cierto o falso, esto nos sirve de forma natural para tareas de clasificación cuando hay dos clases A y B , interpretando que el programa me dice cuándo un hecho es de clase A y cuándo no es de clase A , lo que se interpreta como ser de clase B . Para problemas con más de una clase, se puede proceder convirtiendo el problema en otro equivalente de dos clases, reorganizando la evidencia y aprendiendo primero un programa lógico que defina la clase A (considerando que las evidencias correspondientes al resto de las clases son evidencias negativas de la clase A), luego aprendiendo otro programa lógico para la clase B y así sucesivamente. Esta aproximación a la binarización es la conocida como “uno frente al resto” (*one-vs-all*), aunque en la Sección 6.2.1 vimos algunas otras. Pero además de para la clasificación, la ILP puede usarse también para otras tareas de minería de datos convencionales.

En general, la ILP puede verse desde una doble perspectiva en función del objetivo perseguido, del mismo modo que otras muchas técnicas de minería de datos. Así, si el objetivo es aprender la definición de un predicado hablamos de inducción predictiva, siendo la clasificación la tarea predictiva más estudiada en ILP, mientras que si buscamos aprender patrones generales (que relacionen varios predicados entre sí) entonces decimos que la inducción es descriptiva, siendo el descubrimiento de reglas de asociación la tarea descriptiva más estudiada en ILP, aunque también existen aproximaciones al agrupamiento.

12.4.1 Aproximación directa

Los métodos por aproximación directa utilizan las técnicas y sistemas comentados en el apartado anterior y trabajan directamente sobre la base de datos representada como un

programa lógico. Pese a ser una aproximación directa, es necesario casi siempre un preprocesamiento.

Normalmente, los datos de origen están en una base de datos relacional y es preciso, para muchos de estos sistemas, convertirlos en un formato textual que se ajuste a la sintaxis del sistema en cuestión. Generalmente, suele ser un formato estilo PROLOG. Esta transformación no siempre es sencilla, ya que los datos de una base de datos tienen tipos (cadenas, nominales, booleanos, enteros, reales, etc.) y muchos lenguajes lógicos no son tipados o necesitan especificar los tipos de maneras muy peculiares (mediante directrices especiales). No es un proceso muy complicado, pero requiere su tiempo.

Migrados los datos a la representación del sistema ILP es necesario especificar la tarea. En muchos casos el sistema sólo es capaz de realizar una tarea (clasificación, reglas de asociación, etc.) y sólo es necesario indicar cuál es el predicado principal. En el caso de clasificación, hay que indicar qué atributo o argumento es la clase, en el caso de las reglas de asociación, qué predicados pueden aparecer en el antecedente y consecuente.

Ejemplo de clasificación

Como muestra del uso de sistemas ILP para clasificación hemos elegido el problema de aprender un programa que clasifique animales. Los diferentes animales pueden pertenecer a una de estas cuatro clases: mamífero, pez, reptil y pájaro. Dado que se trata de un problema con más de dos clases, el predicado a aprenderse (denominado *class*) tiene dos argumentos: el primero representa el animal y el segundo su clase. Para la tarea de aprendizaje se dispone de información adicional sobre los animales en una teoría de base *B*. Concretamente, los predicados definidos en *B* son:

- *has_covering(A,C)*: que indica el tipo de cobertura *C* del animal *A*. Los valores de *C* son *hair* (pelo), *scales* (escamas), *feathers* (plumas) y *none* (ninguna especial).
- *has_legs(A,L)*: que indica el número *L* de patas del animal *A*. Los valores de *L* son 4, 2 y 0.
- *has_milk(A)*: que indica si la hembra del animal *A* tiene glándulas mamarias.
- *homeothermic(A)*: que indica si el animal *A* es de sangre caliente.
- *habitat(A,H)*: que indica el tipo de hábitat *H* del animal *A*. Los valores de *H* son *land* (tierra), *water* (agua), *air* (aire) y *caves* (cueva).
- *has_eggs(A)*: que indica si el animal *A* se reproduce por huevos.
- *has_gills(A)*: que indica si el animal *A* respira por branquias.

El sistema ILP elegido para el ejemplo es Progol. Progol [Muggleton 1995] es un sistema *top-down* que aprende las cláusulas una a una. Para ello, comienza con un ejemplo positivo, genera el espacio de hipótesis que cubre este ejemplo y efectúa una búsqueda exhaustiva en este espacio hasta encontrar la cláusula mejor (aquella que maximiza la compactación: talla(ejemplos cubiertos)-talla(cláusula)), y luego repite el proceso hasta no encontrar más cláusulas buenas. Como la búsqueda es exhaustiva, impone un bias del lenguaje consistente en especificar qué predicados tienen que usarse en la cabeza y cuerpo de las cláusulas, y cuál es el tipo y modo de los atributos de los predicados.

Para el ejemplo que nos ocupa la declaración de tipos constaría de los tipos *animal*, *class*, *covering* y *habitat*. Por ejemplo, el tipo *habitat* se define con los hechos

```
habitat(land).    habitat(water).    habitat(air).    habitat(caves).
```

La declaración de modos indica para cada predicado si es ése realmente el predicado a aprender o si se trata de un predicado a usar en el cuerpo de las reglas inducidas (predicados del conocimiento de base o negación de estos predicados), el tipo de sus argumentos y cuáles de éstos deben ser variables de entrada, cuáles variables de salida y cuáles deben ser constantes del tipo especificado.

El conjunto de datos utilizado consta sólo de ejemplos positivos y puede encontrarse en la página web del sistema Progol. La hipótesis inducida está formada por las cuatro reglas mostradas en la Figura 12.6:

```
class(A,bird):-has_covering(A,feathers).
class(A,mammal):- has_covering(A,hair).
class(A,fish):-has_gills(A).
class(A,reptile):-not(has_gills(A)).
```

Figura 12.6. Solución encontrada por Progol para el problema de la clasificación de animales.

Ejemplo de reglas de asociación

En este apartado mostramos un ejemplo de minería de datos con el sistema TERTIUS ([Flach & Lachiche 2001]). El objetivo es ilustrar el tipo de reglas de asociación que algunos sistemas ILP son capaces de descubrir. La aplicación elegida es el problema de encontrar dependencias funcionales en una base de datos relacional.

TERTIUS es un sistema para el descubrimiento de reglas de asociación en lógica de primer orden. Tertius es capaz de tratar conocimiento tanto extensional como intensional, y usa una búsqueda *top-down* donde la noción de generalidad entre las cláusulas se establece a través de la θ -subsunción. Pueden emplearse varios bias del lenguaje (por ejemplo cláusulas Horn, cláusulas con negación en el cuerpo...) así como cierto orden entre los predicados y en los operadores de refinamiento de las cláusulas para reducir el espacio de hipótesis y realizar así una búsqueda más eficiente.

El conjunto de datos utilizado consta de los nueve ejemplos sobre información ferroviaria que se muestran en la Figura 12.7. En concreto, cada ejemplo muestra la dirección del tren, la hora y minuto de salida y la primera parada.

train(utrecht,8,8,den-bosch).	train(utrecht,9,8,den-bosch).
train(tilburg,8,10,tilburg).	train(tilburg,9,10,tilburg).
train(maastricht,8,10,weert).	train(maastricht,9,10,weert).
train(utrecht,8,13,eindhoven-bkln).	train(utrecht,9,13,eindhoven-bkln).
train(tilburg,8,17,eindhoven-bkln).	train(tilburg,9,17,eindhoven-bkln).
train(utrecht,8,25,den-bosch).	train(utrecht,9,25,den-bosch).
train(utrecht,8,31, utrech).	train(utrech,9,43, eindhoven-bkln).
train(utrecht,8,43, eindhoven-bkln).	train(tilburg,8,47, eindhoven-bkln).
train(tilburg,8,47, eindhoven-bkln).	

Figura 12.7. Ejemplo del problema de los trenes.

Vamos a utilizar Tertius para la búsqueda de dependencias funcionales. La solución encontrada por Tertius muestra algunas dependencias interesantes como las siguientes:

```

(1)  equal_dir(G,E) :- train(G,F,C,D),train(E,F,C,D).
(2)  equal_dir(A,E) :- train(A,B,C,D),train(E,F,C,D).
(3)  equal_first(D,G) :- train(A,E,C,D),train(A,E,C,G).
(4)  equal_first(F,D) :- train(A,B,C,D),train(A,E,C,F).

```

La regla (2) muestra la dependencia “los minutos y la primera parada determinan la dirección”. La regla (4) establece que “la dirección y los minutos determinan la primera parada”. Las reglas (1) y (3) son refinamientos de las reglas (2) y (4) respectivamente (ya que añaden la hora para determinar la dirección y la primera parada) y podrían eliminarse.

En la dirección <http://www.cd.bris.ac.uk/Research/MachineLearning/Tertiis/index.html> se puede obtener el sistema TERTIUS así como los conjuntos de datos de algunas aplicaciones, incluido el ejemplo aquí mostrado. Existe además una extensión de este sistema en el sistema WEKA (véase el Apéndice A).

12.4.2 Aproximación mediante proposicionalización

Tal y como hemos comentado al principio del capítulo, muchos problemas reales son esencialmente relacionales, mientras que un gran número de algoritmos de aprendizaje en general y de minería de datos en particular requieren una representación proposicional. A pesar de las limitaciones en expresividad de los métodos proposicionales, éstos son a veces preferibles a los de la RDM [Brandt et al. 2001] (por diversas razones como, por ejemplo, su superioridad en algunas cuestiones como el manejo de valores numéricos, la regresión o las medidas de distancia; o bien, puede suceder que el usuario prefiera seguir usando un método que le es familiar o una herramienta integrada o *suite* donde no dispone de métodos relacionales).

Por ello, y como se ha comentado brevemente con anterioridad, se ha propuesto desde la ILP otra forma de abordar la minería de datos relacional y que consiste en transformar el problema relacional en otro equivalente en notación proposicional (es decir, en notación atributo-valor), aplicar métodos estándar de la minería de datos y luego transformar la hipótesis inducida a una notación relacional. Este proceso, que se conoce como proposicionalización, puede verse como el proceso de transformar bases de datos multi-relacionales que contienen ejemplos estructurados, en un conjunto de datos proposicionales con características derivadas de la forma atributo-valor que describen las propiedades estructurales de los ejemplos. La idea básica es que el conocimiento de base puede usarse para definir nuevos predicados, que se emplean como atributos o características booleanas en la descripción transformada de los datos. Por lo tanto, la proposicionalización puede verse como el proceso de resumir datos almacenados en múltiples tablas en una única tabla que contiene un único registro por ejemplo.

La proposicionalización es un caso especial de construcción de características (visto en la Sección 4.3.2 principalmente) y debe entenderse como un pre-proceso anterior al de la aplicación del algoritmo de aprendizaje. En cualquier caso, sobre la proposicionalización, cuando es posible obtener un problema equivalente, el número de atributos de la versión transformada suele ser alto. Esto ocurre mayormente en los dominios de aplicación interesantes [De Raedt 1998b].

Pese a estas limitaciones, la proposicionalización es una aproximación viable. Por ejemplo, el sistema LINUS [Lavrač et al. 1991] es uno de los primeros sistemas de ILP que

siguen esta aproximación. La construcción de características se realiza de la siguiente forma: cada relación del conocimiento de base B aplicada sobre los argumentos de la relación objetivo (teniendo en cuenta los tipos) define una característica booleana. Cada una de estas nuevas características booleanas es una nueva variable en un problema proposicional del estilo atributo-valor. Veámoslo con un ejemplo.

Ejemplo

En el ejemplo de la Figura 12.3, si consideramos que los argumentos de la relación *hermanas* son X e Y , la relación *mujer/1* (cuyo argumento es del mismo tipo que X e Y) define las características $m(X)$ y $m(Y)$ cuyo valor para una instancia dada, *hermanas("Ana", "Elena")*, será $m("Ana") = \text{true}$ y $m("Elena") = \text{true}$ ya que *mujer("Ana")* y *mujer("Elena")* son ciertas en B .

De igual forma se definen las características $h(X)$ y $h(Y)$ a partir de la relación *hombre/1* y las características $p(X,X)$, $p(X,Y)$, $p(Y,X)$, $p(Y,Y)$ a partir de *padre/2*. La tabla siguiente muestra esta representación proposicional.

CLASE	X	Y	$m(X)$	$m(Y)$	$h(X)$	$h(Y)$	$p(X,X)$	$p(X,Y)$	$p(Y,X)$	$p(Y,Y)$
Pos.	Ana	Elena	true	true	false	false	false	false	false	false
Pos.	María	Rosa	true	true	false	false	false	false	false	false
Pos.	Ester	Pilar	true	true	false	false	false	false	false	false
Neg.	María	Elena	true	true	false	false	false	false	false	false
Neg.	Luis	Juan	false	false	true	true	false	false	false	false
Neg.	Ester	José	true	false	false	true	false	false	true	false

Tabla 12.4. Proposicionalización.

A partir de la representación anterior podemos utilizar cualquier aproximación proposicional: árboles de decisión, clasificador bayesiano naive, vecinos más próximos, etc. En particular LINUS utiliza el árbol de decisión C4.5 en esta segunda fase. Una vez obtenido un modelo proposicional (si éste es en forma de reglas, como ocurre con el C4.5) se desharía el cambio para mostrarlo según el conocimiento y predicados originales.

En este ejemplo se muestra que el número de columnas se dispara incluso para un ejemplo tan sencillo como éste, lo que obliga a utilizar heurísticas para obtener un subconjunto de todas las características. Además, es necesario crear columnas con variables libres. De hecho, en el caso anterior no hemos representado el concepto *mismopadre(X, Y) = p(Z, X) ∧ p(Z, Y)*, que es lo que necesitamos para aprender el concepto *hermanas*. LINUS también aporta técnicas en este sentido.

12.4.3 Bases de datos inductivas

Una base de datos inductiva es una base de datos en la que se almacenan los patrones inducidos por alguna técnica de minería de datos (ILP, árboles de decisión...), preferiblemente declarativa, y que puede ser interrogada usando lenguajes de consulta especialmente diseñados para ello.

La idea que subyace en las bases de datos inductivas es que en muchas ocasiones es más fácil expresar el estado de la base de datos como un conjunto de ejemplos en el que todos los predicados (relaciones) están definidos extensionalmente. En este caso, las técnicas de ILP pueden usarse para descubrir cláusulas para un predicado que son válidas

en dicho estado (es decir, cubren los ejemplos de ese predicado) y que corresponden a la definición intensional del mismo (al modo de una vista). Por lo tanto, una base de datos inductiva puede verse como la versión compactada de una base de datos relacional (es decir, extensional). La relación de las bases de datos inductivas con las bases de datos deductivas (y la programación lógica) es muy estrecha. En [Blockeel & De Raedt 1996] se propone un sistema basado en CLAUDIEN [De Raedt & Bruynooghe 1993] para el diseño de bases de datos inductivas que deriva automáticamente reglas y construye definiciones intensionales.

Otra forma de extraer patrones complejos de una base de datos es usar principios de la programación lógica inductiva mediante lenguajes de consulta (de forma similar a los lenguajes de consulta inductivos vistos en la Sección 5.5.1). Un ejemplo de lenguaje de consulta basado en ILP es RDM [De Raedt 2000] el cual proporciona primitivas para descubrir patrones frecuentes (o patrones infrecuentes si se centra en los ejemplos negativos) y reglas de asociación, para realizar inducción predictiva y descriptiva y para razonar sobre la generalidad de hipótesis. Este lenguaje actúa sobre bases de datos tradicionales, deductivas o inductivas, lo que le permite descubrir patrones sobre múltiples predicados, sean éstos extensionales o intensionales.

12.5 Otros métodos relacionales y estructurales

Tal y como hemos comentado en la introducción, aunque la ILP es el área más estudiada en la minería de datos relacional no es la única aproximación existente. En esta sección presentamos algunos de estos métodos alternativos [Mooney et al. 2002; Džeroski 2003], así como otros que son más gráficos que declarativos (como por ejemplo el aprendizaje de grafos), e ilustramos las tareas de minería de datos y las aplicaciones para las que han sido empleados.

12.5.1 Aprendizaje basado en grafos

En el apartado anterior hemos comentado algunos ejemplos con estructura, por ejemplo, una molécula. Otros ejemplos con estructura compleja podrían ser rutas de transportes, rutas en una red de computadores, influencias de los antecedentes familiares en una enfermedad, etc. Todos estos ejemplos se pueden representar con expresividad relacional, utilizando para ello varias relaciones para representar un individuo. No obstante, existen maneras alternativas de representar este tipo de individuos con estructura: los grafos. De hecho, los grafos son algunas de las estructuras más estudiadas en computación y en matemática discreta.

Por todo lo anterior, la minería de datos basada en grafos (MG) ha cobrado importancia en los últimos años [Washio & Motoda 2003]. Esta área reciente es muy prometedora desde el punto de vista práctico ya que los grafos son estructuras de datos habituales en campos como la biología, la química, las redes de comunicación o los textos semiestructurados (HTML y XML).

Aunque la minería de grafos tiene una fuerte relación con la RDM no es exactamente lo mismo: la minería de grafos tiene como objetivo el aprendizaje de subestructuras dentro de los grafos. Veamos un ejemplo de aprendizaje supervisado basado en grafos [Holder &

Cook 2003]. Consideremos el conjunto de ejemplos positivos y negativos de la Figura 12.11 que definen figuras compuestas por objetos apilados de diferentes formas.

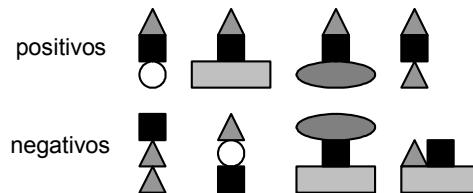


Figura 12.8. Ejemplos positivos y negativos de figuras compuestas por objetos.

Estos datos se pueden representar como pequeños grafos, por ejemplo, el primer ejemplo positivo puede expresarse mediante el grafo de la Figura 12.9.

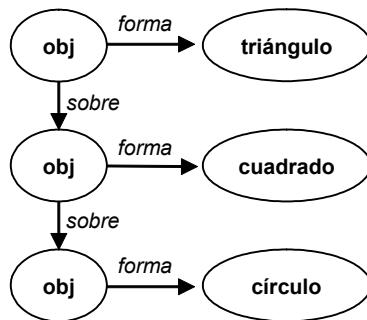


Figura 12.9. Representación de una evidencia positiva con notación de grafos.

Una aproximación al aprendizaje supervisado podría encontrar el siguiente subgrafo de la Figura 12.10 que aparece habitualmente en los grafos positivos pero no en los grafos negativos

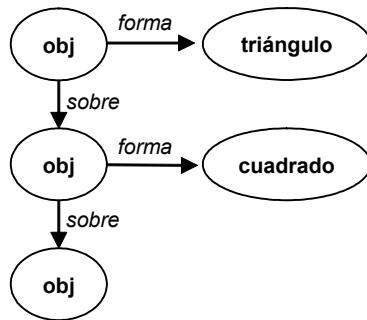


Figura 12.10. Subgrafo patrón de la evidencia positiva.

El ejemplo anterior muestra que la búsqueda de subgrafos puede utilizarse para resolver bastantes tareas de minería sobre grafos o, en general, minería estructural. Este problema, es decir, el determinar cuándo un grafo es subgrafo de otro, se resuelve en MG usando eficientes métodos de búsqueda.

De una manera más general, entre las aproximaciones a la MG destacamos: las basadas en búsqueda voraz, las basadas en ILP, las basadas en bases de datos inductivas, las basadas en la teoría de grafos y las basadas en funciones kernel.

- Aproximación basada en búsqueda voraz: éste es el tipo de búsqueda seguido en los primeros trabajos de MG. La búsqueda voraz puede ser "primero en profundidad" (*depth-first*, DF), "primero en amplitud" (*breadth-first*, BF) o *beam* (una búsqueda BF pero limitando el número máximo de ramas). Entre los sistemas pioneros destacamos SUBDUE [Cook & Holder 2000] el cual es capaz de descubrir subgrafos altamente repetidos en un grafo etiquetado aunque también puede usarse para tareas de clasificación y agrupamiento de grafos. SUBDUE se ha aplicado en varios problemas importantes de la RDM en áreas como la biología molecular, la geología y el análisis de programas.
- Aproximación basada en ILP: Otra posibilidad es usar técnicas de ILP para MG, dado que cualquier grafo general tiene una fácil representación en lógica de primer orden. Uno de los primeros sistemas usados para encontrar subestructuras frecuentes en grafos es WARMR [Dehaspe & Toivonen 1999]. Este sistema combina ILP con una búsqueda similar al algoritmo Apriori y se ha aplicado al problema de la predicción de carcinogénesis de compuestos químicos. Dado que la ILP permite inducir patrones en los que los argumentos de los predicados pueden contener variables, estos patrones son más generales y complejos que los grafos. Desde el punto de vista de los grafos esto podría significar patrones más ricos, por ejemplo que una porción de un patrón de un grafo podría emparejar con cualquier subgrafo, o que dos partes de un patrón deben ser idénticos sin especificar la estructura de estas partes. Otra ventaja de los métodos ILP es la posibilidad de expresar conocimiento de base, lo que permite introducir patrones de subgrafos predeterminados y emplearlos para minar únicamente los patrones que los contienen o que son similares a éstos.
- Aproximación basada en bases de datos inductivas: el marco de las bases de datos inductivas se puede aplicar a la MG almacenando en una base de datos inductiva los subgrafos y/o relaciones entre subgrafos que se han obtenido mediante alguna técnica de MG para luego hacer consultas a esa base de datos. Un ejemplo de sistema que sigue este marco es MolFea [De Raedt & Kramer 2001].
- Aproximación basada en la teoría de grafos: permite minar un conjunto completo de subgrafos haciendo uso de una medida de soporte. La idea básica es similar a los algoritmos de reglas de asociación como Apriori, que buscan conjuntos de datos frecuentes (*frequent itemsets*). Se comienza desde los grafos frecuentes (aquellos cuyo soporte es mayor que un soporte de referencia *minsup*) que constan de un solo vértice. A continuación, se buscan grafos frecuentes candidatos que tienen dos vértices. Este proceso se repite incrementando cada vez la talla del grafo en un vértice hasta que se han explorado todos. Como ejemplo de esta aproximación mencionamos el sistema AGM [Inokuchi et al. 2003], el cual se ha aplicado al análisis de la asociación de la subestructura molecular de componentes químicos con su actividad mutagenética.
- Aproximación basada en funciones núcleo (kernel): se usa para la clasificación de grafos [Kashima & Inokuchi 2002], aunque en realidad esta aproximación no trabaja con datos grafo sino con datos expresados como vectores de características (las etiquetas de los vértices y los ejes). La idea consiste en encontrar una función núcleo

que defina la similitud entre grafos, la cual se usa para clasificar los grafos en clases binarias usando máquinas de vectores soporte (véase el Capítulo 14).

La minería de grafos también tiene un aplicación directa en la minería de la estructura web, que veremos en el Capítulo 21, ya que, lógicamente, la web se puede ver como un grafo dirigido, donde los documentos son nodos y las aristas son hipervínculos.

12.5.2 Modelos probabilísticos relacionales

Los modelos relacionales probabilísticos (*Probabilistic relational models*, PRM, [Koller & Pfeffer 1998; Friedman et al. 1999]) son una extensión de las redes bayesianas (véase el Capítulo 10) para manejar datos relacionales. Este formalismo no usa la lógica de las cláusulas de Horn como marco de representación, sino el modelo entidad-relación³⁸. La idea es que la información sobre un tipo entidad se almacene en una relación cuyos argumentos son los atributos de la entidad. Hay ciertos atributos fijos y conocidos para todas las instancias de las relaciones. El resto de atributos son llamados probabilísticos y sus valores están por determinar. Las PRM permiten que los atributos probabilísticos de una entidad dependan (probabilísticamente) de otros atributos de la misma entidad o bien de atributos de otra entidad relacionada. Las PRM's se han aplicado a un cierto número de problemas interesantes en los campos de la biología molecular, la clasificación de hipertexto y el agrupamiento relacional.

Finalmente, se han desarrollado otras extensiones de las redes bayesianas [De Raedt & Kersting 2003], como las redes bayesianas orientadas a objetos [Koller & Pfeffer 1997], así como otros modelos que combinan otros tipos de razonamiento probabilístico con la representación relacional, como los modelos relacionales estocásticos [Muggleton 2002].

12.5.3 Aproximaciones relacionales basadas en distancia

Para definir aproximaciones relacionales basadas en distancia para tareas de clasificación y agrupamiento, primero hay que actualizar las medidas de distancia del caso proposicional al relacional. Estas nuevas medidas relacionales pueden luego usarse en aproximaciones estándar (véase el Capítulo 16) obteniéndose así métodos con expresividad relacional.

Una de estas medidas de distancia relacional es la definida en el sistema RIBL [Emde & Wettschereck 1996]. Dados dos ejemplos del predicado objetivo ($e_1=pred(E_1, A_{11}, \dots, A_{1n})$ y $e_2=pred(E_2, A_{21}, \dots, A_{2n})$) su distancia se calcula de forma jerárquica a través de los hechos relacionados con las instancias iniciales e_1 y e_2 . En el nivel cero, se calcula la distancia $d(e_1, e_2)$ entre estos dos ejemplos como en el caso proposicional, es decir comparando entre sí los valores de cada par de argumentos, $d(A_{1i}, A_{2i})$, usando medidas de distancia proposicionales. En el nivel a la profundidad 1, se calcula la distancia entre los objetos inmediatamente relacionados con los objetos iniciales (es decir, aquellos que tienen algún argumento coincidente con las instancias iniciales). Para este segundo nivel y posteriores se puede utilizar la información del esquema de la base de datos. Este proceso continúa hasta que se alcanza una profundidad límite. A continuación ilustramos con un ejemplo este proceso.

³⁸ El modelo entidad-relación es un modelo semántico usado generalmente a un nivel más conceptual y abstracto que el nivel lógico usual del modelo relacional (véase por ejemplo [Celma et al. 2003]).

Supongamos un almacén de perfumería que mantiene información sobre los productos, los clientes, los pedidos y las ventas efectuadas. Un hecho de la relación objetivo *producto(Id_prod, Stock, Precio,Prov)* expresa que el producto *Id_prod* tiene un *stock* de *Stock* unidades, un precio igual a *Precio* y que es suministrado por el proveedor *Prov*. Las relaciones definidas en el conocimiento de base son *pedido(Id_prod, Cant_ped)*, que expresa la cantidad pedida *Cant_ped* del producto *Id_prod*, *venta(Id_prod, Cant_ven, Id_cli)*, que expresa la cantidad vendida *Cant_ven* del producto *Id_prod* al cliente *Id_cli* y *cliente(Id_cli, Dirección)*, que indica la dirección *Dirección* del cliente *Id_cli*. A continuación se muestran algunos hechos de estas relaciones.

```
producto(prod1, 100, 69, prov1).
producto(prod2, 52, 75, prov2).
```

```
pedido(prod1, 10).
pedido(prod2, 3).
```

```
venta(prod1, 2, comercio1).
venta(prod1, 5, comercio2).
venta(prod2, 7, comercio1).
```

```
cliente(comercio1, "C/Rosas, Centro Comercial Este").
cliente(comercio2, "C/Margarita, Centro Comercial Oeste").
```

Para calcular la distancia entre los dos ejemplos $e_1 = \text{producto}(\text{prod1}, 100, 69, \text{prov1})$ y $e_2 = \text{producto}(\text{prod2}, 52, 75, \text{prov2})$, procederíamos jerárquicamente por niveles. La Figura 12.11 muestra los objetos relacionados con e_1 .

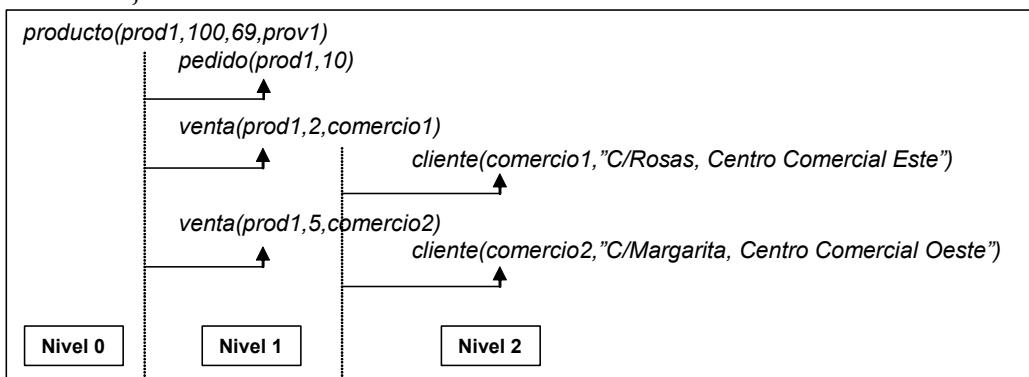


Figura 12.11. Jerarquía por niveles de los objetos relacionados con el ejemplo *producto(prod1,100,69,prov1)*.

La distancia $d(e_1, e_2)$ al nivel 0 se calcula como en el caso proposicional, es decir

$$d(e_1, e_2) = [d(\text{prod1}, \text{prod2}) + d(100, 52) + d(69, 75) + d(\text{prov1}, \text{prov2})]/4$$

Al nivel 1 se comparan los pedidos y las ventas de los dos productos, al nivel 2 se comparan los clientes y así sucesivamente, utilizando para ello las claves ajenas. Por ejemplo, al nivel 1 existen varios conjuntos de hechos relacionados con los ejemplos: relacionados con e_1 tenemos $e_{1,\text{compra}} = \{ \text{compra}(\text{prod1}, 10) \}$ y $e_{1,\text{venta}} = \{ \text{venta}(\text{prod1}, 2, \text{comercio1}), \text{venta}(\text{prod1}, 5, \text{comercio2}) \}$, y con e_2 tenemos $e_{2,\text{compra}} = \{ \text{compra}(\text{prod2}, 3) \}$ y $e_{2,\text{venta}} = \{ \text{venta}(\text{prod2}, 7, \text{comercio1}) \}$. Entonces, la distancia $d(e_1, e_2)$ al nivel 1 se calcula como

$$d(e_1, e_2) = [d(e_{1,\text{compra}}, e_{2,\text{compra}}) + d(e_{1,\text{venta}}, e_{2,\text{venta}})]/2$$

Las distancias a varios niveles se ponderan en una distancia integrada. Para calcular distancias entre conjuntos, se selecciona el conjunto de menor cardinalidad y se calcula la distancia de cada uno de sus elementos a los elementos del otro conjunto, y luego se selecciona la distancia menor. Una vez calculadas las distancias, el sistema RIBL las utiliza junto con el método de los k-vecinos más próximos.

12.5.4 Árboles de decisión relacionales

Los árboles de decisión relacionales tienen la misma estructura que los árboles de decisión proposicionales (Capítulo 11): los nodos internos contienen condiciones mientras que las hojas indican el valor predicho para la clase. La principal diferencia entre ambos tipos de árboles es que los test en los nodos internos de un árbol relacional son en general conjunciones de literales cuyas variables están cuantificadas existencialmente, en lugar de condiciones sobre el valor de un atributo. Por lo tanto, se trata de árboles binarios en los que cada nodo interno tiene dos ramas: la de la izquierda indica que el test es cierto y la rama de la derecha representa que el test es falso. Por ejemplo, la Figura 12.12 muestra un árbol de decisión relacional para el ejemplo del almacén de perfumería del apartado anterior (ampliado ahora con gestión de stocks) para determinar cuál es la acción O a realizar (dar de baja, promocionar o hacer un pedido) para cada producto P ($acción(P,O)$) en función de sus ventas en el semestre actual. Otros atributos son ($ventas_semestre(P,M)$) y su stock actual ($stock(P,S)$) y mínimo ($stock_min(P,Sm)$).

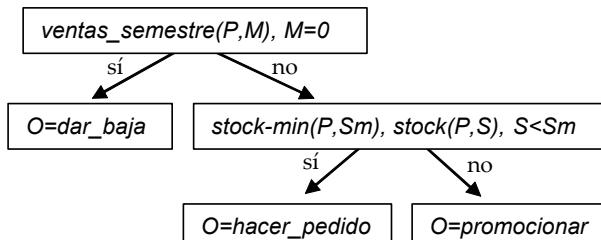


Figura 12.12. Árbol de decisión relacional para predecir la acción a tomar con los productos de un almacén.

Al igual que con los árboles de decisión proposicionales, los árboles relacionales pueden expresarse como un conjunto ordenado de reglas que se llaman listas de decisión de primer orden. Estas listas ordenadas se obtienen recorriendo el árbol (en un orden primero en profundidad) pudiendo representarse mediante programas Prolog haciendo uso del corte³⁹ para garantizar que las cláusulas se prueban en el orden adecuado. Por ejemplo, la Figura 12.13 muestra el árbol de la Figura 12.12 expresado como un programa Prolog.

```

accion(P, dar_baja) :- ventas_semestre(P, M), M=0, !.
accion(P, hacer_pedido) :- stock-min(P, Sm), stock(P, S), S<Sm, !.
accion(P, promocionar).
  
```

Figura 12.13. Lista de decisión correspondiente al árbol de la Figura 12.12.

Los dos algoritmos más destacados para inducir árboles relacionales, SCART [Kramer & Widmer 2001] y TILDE [Blockeel & De Raedt 1998], son extensiones de dos famosos

³⁹ El operador de corte (en notación `!`) es una facilidad extra-lógica del lenguaje Prolog (y similares) que permite podar el espacio de búsqueda.

algoritmos para la inducción de árboles proposicionales por divide y vencerás, CART y C4.5, respectivamente. FOIL [Quinlan 1990] es la extensión relacional más popular de algoritmos por cobertura.

12.5.5 Reglas de asociación relationales

El descubrimiento de patrones recurrentes en grandes colecciones de datos se ha convertido en uno de los tópicos centrales de la minería de datos. Las reglas de asociación son el ejemplo básico de este tipo de planteamiento, y el ejemplo prototípico “el análisis de la cesta de la compra”.

En [Dehaspe & Toivonen 1999] y [Dehaspe & Toivonen 2001] se aborda el problema de descubrir preguntas datalog frecuentes (es decir, preguntas al estilo de Prolog, con la forma “?- A_1, \dots, A_n ”, y donde los A_i son átomos). Estas preguntas son equivalentes a consultas SQL. Por ejemplo, la pregunta “?- $cliente(C,D)$, $pedido(P,Cant_ped,C)$, $Cant_ped > 50$ ”, sobre los clientes que realizan pedidos de productos solicitando cantidades superiores a 50 unidades, puede expresarse como la consulta SQL

```
SELECT Cliente.Id
  FROM Cliente, Pedido, Producto
 WHERE Cliente.Id=Pedido.Cid AND Pedido.Cant>50
```

La principal diferencia entre las preguntas en Prolog y las consultas SQL es que las primeras pueden contener variables que comparten su valor. Así, el cliente C en $cliente(C,D)$ y el cliente C que realiza el pedido en $pedido(P,Cant_ped,C)$ son el mismo. La respuesta a una pregunta son todas las sustituciones que hacen que la pregunta tenga éxito. Para encontrar patrones frecuentes se selecciona un átomo como clave; este átomo tiene que estar presente en todas las preguntas y con respecto al cual se calcula la frecuencia de la pregunta. La frecuencia absoluta de una pregunta Q es el número de sustituciones respuesta θ para las variables del átomo clave para las que la pregunta $Q\theta$ tiene éxito en la base de datos y la frecuencia relativa (soporte) es el cociente entre la frecuencia absoluta y el número de sustituciones respuesta del átomo clave.

Las reglas de asociación relationales pueden obtenerse de forma similar al caso proposicional a partir de preguntas datalog frecuentes. Así, dadas dos preguntas frecuentes Q_1 y Q_2 tales que Q_2 θ -subsume Q_1 , podemos derivar la regla $Q_1 \rightarrow Q_2$. Ya que Q_2 extiende Q_1 , la regla se puede expresar simplificadamente poniendo en el consecuente únicamente los átomos de Q_2 no presentes en Q_1 . Por ejemplo, si $Q_1=$ “?- $cliente(C,D)$, $pedido(P,Cant_ped,C)$ ” y $Q_2=$ “?- $cliente(C,D)$, $pedido(P,Cant_ped,C)$, $Cant_ped > 50$ ” son preguntas frecuentes con frecuencias absolutas de 30 y 20 respectivamente, entonces “ $cliente(C,D)$, $pedido(P,Cant_ped,C)$ ” \rightarrow $Cant_ped > 50$ ” es una regla de asociación con soporte 20 y confianza 20/30 (66,7 por ciento).

El sistema WARMR [Dehaspe & Toivonen 1999] descubre preguntas frecuentes mediante un algoritmo cuyo bucle principal se basa en dos fases: fase de generación y fase de evaluación. Inicialmente, se parte de la pregunta formada sólo por el átomo clave, a continuación todas las preguntas formadas por un átomo cuya frecuencia está por debajo de un cierto umbral se eliminan (fase de evaluación) y se calcula el conjunto de preguntas candidatas formadas por dos átomos (fase de generación). El bucle se repite incrementando en 1 cada vez el número de átomos de las preguntas candidatas hasta el nivel máximo de átomos en una pregunta. Las preguntas más frecuentes obtenidas por el algoritmo anterior

se pos-procesan para obtener reglas de asociación cuya confianza exceda un cierto valor umbral (siguiendo el proceso antes descrito). Este sistema se ha usado, por ejemplo, para la predicción de carcinogénesis de compuestos químicos.

12.5.6 Inducción de programas lógico-funcionales

Como hemos visto a lo largo del capítulo muchas son las ventajas que ofrece la ILP a la minería de datos. Sin embargo, existen dos limitaciones en este formalismo desde la perspectiva de la minería de datos: primero, en la programación lógica las funciones deben tratarse artificialmente a través de predicados con un argumento más que representa el resultado de la función (por ejemplo, *predicado(Entrada1, Entrada2, ..., EntradaN, Salida)*), y segundo, los problemas de minería se caracterizan por usar sólo ejemplos positivos, mientras que muchos sistemas de ILP requieren la presencia de ejemplos negativos. Aunque existen formas de evitar este último problema [Muggleton 1996], ya hemos comentado que la ILP aborda de forma natural problemas de clasificación con sólo dos clases pero son necesarias transformaciones para tratar problemas con más de dos clases, algo habitual en minería de datos. En [Ferri et al. 2000] se presenta una aproximación a la minería de datos usando la programación lógico-funcional como paradigma subyacente, denominada IFLP (*Inductive Functional Logic Programming*). Usar lenguajes lógico-funcionales para la representación del conocimiento nos permite disponer de las ventajas de ambos paradigmas: la presencia de variables lógicas propias de la programación lógica, y la posibilidad de usar información de tipos y de definir funciones, propias de la programación funcional. Los programas lógico-funcionales son programas lógicos extendidos con teorías ecuacionales de Horn, cuyas cláusulas son de la forma $l = s \leftarrow e_1, \dots, e_n$, donde las e_i son ecuaciones.

Veamos con un ejemplo tomado del repositorio UCI (*lenses dataset*, véase el Apéndice B) las ventajas de usar este tipo de representación. Supongamos que un óptico desea implantar un sistema para asistirle en el tipo de lente de contacto que debe usar primero con un nuevo cliente/paciente. Para ello, dispone de bastantes casos previos en los que recomendó el tipo correcto de lente (dura o blanda) o bien el uso de gafas.

```

lenses(X0, hipermetropía, no, normal)=blanda
lenses(joven, miopía, no, normal)=blanda
lenses(X0, miopía, sí, normal)=dura
lenses(joven, hipermetropía, sí, normal)= dura
lenses(pre-presbiótico, miopía, no, normal)=blanda
lenses(pre-presbiótico, hipermetropía, sí, normal)=no
lenses(presbiótico, miopía, no, normal)=no
lenses(X0, X1,X2, reducida)=no
lenses(presbiótico, hipermetropía, sí, normal)=no

```

Figura 12.14. Solución IFLP al problema de la recomendación del tipo de lentes de contacto.

El objetivo es construir un programa que clasifique cada nuevo paciente en una de estas tres clases (que se corresponden con el tipo de lente): *dura*, *blanda* y *no*. Tal y como hemos comentado, si abordáramos este problema en el marco de la ILP tendríamos que optar por una de estas dos posibles formas de resolverlo, dado que directamente la programación lógica inductiva sólo puede tratar con dos clases:

- Añadir un atributo más a la relación *lenses* que representaría la clase.

- Convertir el problema a clases binarias (binarización) y obtener tres modelos diferentes (uno para cada posible valor del atributo clase, tal y como se ha comentado en la Sección 6.2.1).

Pero en realidad, *lenses* es una función cuyo resultado es el tipo de lente recomendada. Desde esta perspectiva la IFLP podría obtener un modelo como el de la Figura 12.14.

Otro tipo de aplicaciones de minería de datos para los que la inducción de programas lógico-funcionales es ventajosa es la minería de datos no estructurados o semiestructurados (muy relacionada con la minería web, véase el Capítulo 21). Por ejemplo, un documento XML puede convertirse automáticamente en un árbol de términos funcionales y éste puede tratarse directamente en el marco de la IFLP. La Figura 12.15 muestra un ejemplo de clasificación de buenos clientes en el marco IFLP.

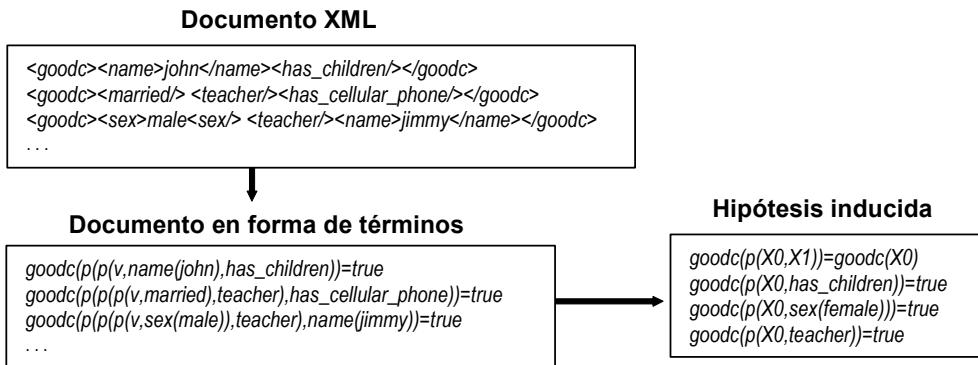


Figura 12.15. Problema de clasificar los buenos clientes en el marco IFLP.

La información inicial está en un documento XML, cuyos datos se transforman en términos usando listas (donde v representa la lista vacía y $p(lista, elemento)$ construye una lista a partir de otra añadiéndole un elemento), que son usados para inducir la hipótesis.

Una de las características que se observan en este ejemplo es que las aproximaciones funcionales (y, por tanto, las lógico-funcionales) permiten tratar con términos con estructura del estilo $goodc(p(p(sex(female), ...))$ y no “aplanado” del estilo $goodc(jenny, no_children, yes_cellularphone, female, ?, ?, firewoman, ?...)$, que muchas veces obliga a tener campos en blanco.

12.6 Sistemas

En esta sección revisamos algunos de los sistemas ILP (además de los ya mencionados en las secciones anteriores) que han sido utilizados para tareas de minería de datos relacional. Cada uno de estos sistemas utiliza lenguajes de hipótesis ligeramente diferentes.

CLAUDIEN (*CLAUsal Discovery ENgine* [De Raedt & Bruynooghe 1993]) es un sistema ILP *top-down* que opera sólo con ejemplos positivos y genera hipótesis que son teorías clausales. La forma de las cláusulas de la hipótesis se especifica a través de un bias del lenguaje que combina esquemas de reglas y conjuntos de reglas. CLAUDIEN ha sido aplicado a una amplia variedad de problemas [Dehaspe et al. 1994], incluyendo el descubrimiento de dependencias funcionales y restricciones de integridad, en áreas diversas, como el de la calidad del agua de los ríos. CLAUDIEN es accesible para

propósitos académicos en la dirección http://www.cs.kuleuven.ac.be/~ml/CWIS/claudien/claudien_avail-E.shtml.

MOBAL [Morik et al. 1993] es un sistema basado en θ -subsunción que usa un bias del lenguaje para especificar modelos de reglas, las cuales indican la forma que deben tener las cláusulas que forman la hipótesis (es decir, son como plantillas). Este sistema se ha aplicado al problema de aprender reglas de acceso a equipos de telecomunicaciones en contextos seguros. MOBAL se puede encontrar en <http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/learning/systems/mobal/>.

GOLEM [Muggleton & Feng 1990] es un sistema usado para encontrar reglas de clasificación que expliquen los ejemplos de entrenamiento. Se basa en generalización menos general restringida (rlgg) y la búsqueda es *bottom-up*. Este sistema, disponible vía ftp <ftp://ftp.mlnet.org/ml-archive/ILP/public/software/golem/>, se ha aplicado a problemas de clasificación en el área de la biología, como por ejemplo encontrar la estructura secundaria de proteínas o el diagnóstico de fallos en satélites.

FOIL [Quinlan 1990] es una extensión al caso de primer orden del algoritmo de aprendizaje de reglas (como el CN2, basado en cobertura, visto en el Capítulo 11), usado en minería de datos tradicional (proposicional). Este sistema aprende a partir de ejemplos positivos, negativos y de conocimiento de base extensional (en la forma de tuplas de constantes, es decir, de hechos) definiciones de predicados expresadas como cláusulas de Horn libres de funciones y que pueden ser recursivas. La estrategia de búsqueda comienza con una cláusula general ($p(X_1, \dots, X_n)$ siendo p el predicado a aprender) y se va aplicando un operador de refinamiento bajo θ -subsunción (instanciando variables o añadiendo literales al cuerpo de la cláusula seleccionados según cierta heurística basada en ganancia de información). FOIL se ha aplicado a un gran número de problemas, como el aprendizaje de los tiempos pasados de los verbos en inglés a través de listas de decisión de primer orden [Mooney & Califff 1995] o el problema de la mutagénesis. FFOIL [Quinlan 1996a] es una variante del FOIL que trabaja mejor con problemas de más de dos clases, al tener una formulación funcional. FOIL y FFOIL están libremente disponibles desde la página personal de R. Quinlan <http://www.cse.unsw.edu.au/~quinlan/>.

Progol [Muggleton 1995] es un sistema ILP bastante popular. Es del tipo *top-down* y aprende las cláusulas una a una. Véase el ejemplo de la Sección 12.4.1. Este sistema se ha empleado, entre otros, en diversos problemas de biomedicina (predecir la actividad cancerígena de compuestos químicos) y en la identificación de los estados legales de un sistema. Progol es accesible desde la página personal de S.H. Muggleton (<http://www.doc.ic.ac.uk/~shm/>).

Para finalizar, y en lo que respecta a las direcciones futuras de la RDM, debemos mencionar que el principal reto al que se enfrenta la RDM es la escalabilidad de sus métodos para trabajar con bases de datos muy grandes. La dificultad estriba en que la RDM requiere explorar un espacio de hipótesis posibles mucho más grande que los métodos basados en representaciones atributo-valor, y en que el proceso de inferencia es más complejo. Por ello, creemos que la RDM va a ser uno de los tópicos a ser investigados en el desarrollo de la siguiente generación de sistemas de minería de datos.

Capítulo 13

REDES NEURONALES ARTIFICIALES

Emilio Corchado y Colin Fyfe

Las redes neuronales artificiales (RNA) son sistemas conexionistas dentro del campo de la Inteligencia Artificial, las cuales, dependiendo del tipo de arquitectura neuronal, pueden tener diferentes aplicaciones. Pueden utilizarse para el reconocimiento de patrones, la compresión de información y la reducción de la dimensionalidad, el agrupamiento, la clasificación, la visualización, etc. Ya que muchas de estas tareas lo son también de la minería de datos, las RNA se pueden utilizar, por tanto, como una herramienta para llevar a cabo minería de datos.

En este capítulo revisamos los dos tipos principales de RNA, aquellas que emplean aprendizaje supervisado y aquellas que utilizan aprendizaje no supervisado. Respecto a la primera categoría, nos centraremos en el estudio del Perceptrón Multicapa y las redes denominadas Funciones de Base Radial. En el caso de la segunda categoría presentaremos el aprendizaje basado en el empleo de la regla de Hebb y el aprendizaje competitivo.

13.1 Introducción

Las redes neuronales artificiales son un método de aprendizaje cuya finalidad inicial era la de emular los procesadores biológicos de información. Las RNA parten de la presunción de que la capacidad humana de procesar información se debe a la naturaleza biológica de nuestro cerebro. Por tanto, para imitar esta característica debemos estudiar y basarnos en el uso de soportes artificiales semejantes a los existentes en nuestro cerebro.

Primero vamos a presentar las propiedades más interesantes del procesamiento neuronal humano. Éstas pueden ser descritas de la siguiente manera:

- El procesamiento de información biológico es robusto y tolerante a fallos. Al comienzo de nuestras vidas (semanas antes de nuestro nacimiento) poseemos el máximo número de neuronas. Desde ese momento, diariamente perdemos muchos

miles. A pesar de ello, nuestro cerebro continúa funcionando durante muchos años sin alcanzar un grado de deterioro que pueda afectar a nuestras capacidades.

- Los procesadores de información biológicos son flexibles. No necesitan volver a ser programados cuando se cambia de entorno o ambiente, sino que ellos mismos se reajustan al entorno. Por tanto, nos adaptamos a los nuevos entornos y aprendemos.
- Son capaces de trabajar con información incompleta, con ruido o inconsistente de una manera semejante a la que se puede alcanzar empleando computadoras cuando se usa una gran cantidad de programación sofisticada y sólo cuando el contexto de ese conjunto de datos se ha analizado en detalle.
- La maquinaria que realiza estas funciones es altamente paralela, pequeña, compacta y disipa poca cantidad de energía.

Comencemos con una descripción de la maquinaria biológica que vamos a emular.

13.1.1 Neuronas biológicas y artificiales

Una neurona simple se muestra en la parte izquierda de la Figura 10.2. La neurona recibe información a través de las sinapsis de sus dendritas. Cada sinapsis representa la unión de un axón de otra neurona con una dendrita de la neurona representada en la figura. Una transmisión electro-química tiene lugar en la sinapsis, la cual permite a la información ser transmitida desde una neurona a la próxima. La información es entonces transmitida a lo largo de las dendritas hasta que alcanza el cuerpo de la célula. Allí tiene lugar el sumatorio de los impulsos eléctricos que lo alcanzan y se aplica algún tipo de función de activación a éste. La neurona se activará si el resultado es superior a un determinado límite o umbral. Esto significa que enviará una señal (en forma de una onda de ionización) a lo largo de su axón con la finalidad de comunicarse con otras neuronas. Ésta es la manera en la que la información pasa de una parte de la red de neuronas a otra. Es muy importante tener en cuenta que las sinapsis tienen diferente rendimiento y que éste cambia al transcurrir el tiempo de vida de la neurona. Volveremos a hablar de esta característica más adelante cuando nos centremos en el aprendizaje.

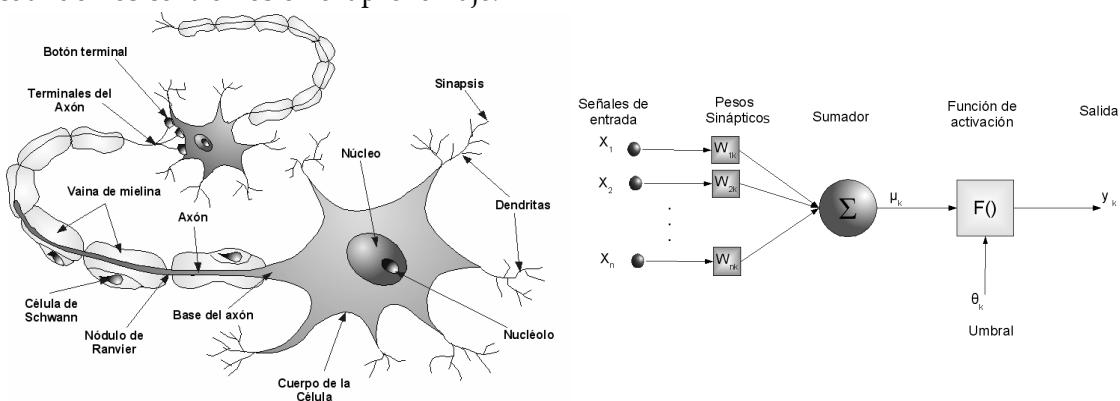


Figura 13.1. Izquierda: diagrama representativo de una neurona real. Derecha: diagrama de una neurona artificial.

Normalmente modelamos una neurona biológica de la manera que se muestra en la Figura 10.2 (derecha). Esta figura incluye una entrada externa adicional, denominada polarización o “bias” y denotada por θ_k , cuya finalidad es la de poder aumentar o disminuir el umbral

de excitación de la neurona dependiendo de si es un valor positivo o negativo, respectivamente.

Las entradas se representan por el vector de entrada, x , y el rendimiento de las sinapsis se modela mediante un vector de pesos, w . Entonces el valor de salida de esta neurona viene dado por:

$$y = f\left(\sum_i w_i x_i\right) = f(\mathbf{w} \cdot \mathbf{x}) = f(\mathbf{w}^T \mathbf{x})$$

donde f es la función de activación.

Cuando tenemos una red de neuronas, las salidas de unas se conectan con las entradas de otras. Si el peso entre dos neuronas conectadas es positivo, el efecto producido es de excitación. Por el contrario, si es negativo, este efecto es de inhibición.

Por tanto, podemos ver que una única neurona es una unidad de procesamiento muy simple. Se considera que el potencial de las redes neuronales artificiales proviene de la capacidad que proporciona el empleo de muchas de estas unidades simples y robustas al actuar en paralelo. De nuevo podemos considerar que estamos simulando (o intentando simular) lo que ocurre en la realidad debido a que los seres humanos poseen varios cientos de billones de neuronas. Normalmente imaginamos las neuronas actuando conjuntamente en capas como se muestra en la Figura 13.2.

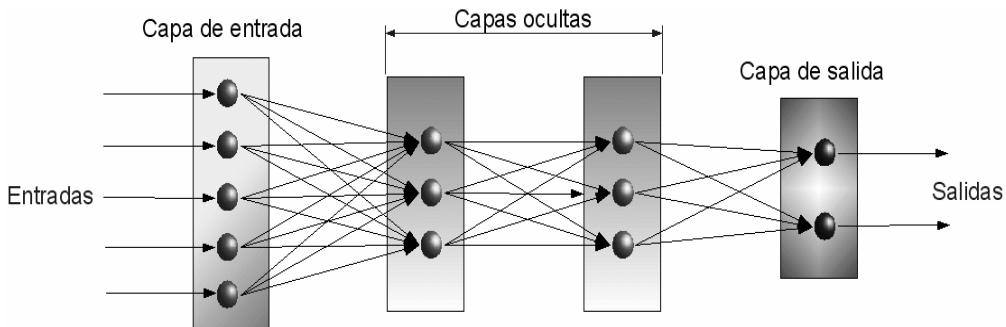


Figura 13.2. Ejemplo de red neuronal artificial formada por cuatro capas de neuronas.

En esta figura se observa un conjunto de entradas (el vector de entrada, x) accediendo a la red desde el lado izquierdo y se propaga a través de la red hasta que la activación alcanza la capa de salida. Las capas intermedias son conocidas como las capas ocultas ya que son invisibles desde fuera de la red.

Hay dos modos de trabajo en una RNA:

- modo de transferencia de la activación: cuando la activación es transmitida por toda la red. Éste es el modo de funcionamiento o de aplicación y está asociado a la operación de propagación hacia adelante.
- modo de aprendizaje: cuando la red se organiza normalmente a partir de la transferencia de activación más reciente.

En las secciones siguientes consideraremos el modo de aprendizaje.

13.2 El aprendizaje en las redes neuronales artificiales

Hemos afirmado que las RNA no necesitan volver a ser programadas al cambiar de entorno. Esto no quiere decir que sus comportamientos no cambien con la finalidad de adaptarse al nuevo entorno. Estos cambios son debidos a variaciones en los pesos de la red. Los cambios en los pesos de una red neuronal dan lugar al aprendizaje. Éstos se producen para modelar los cambios en el rendimiento de las sinapsis de las redes neuronales reales. Se cree que nuestro aprendizaje se debe a cambios en el rendimiento o eficiencia de las sinapsis, a través de las cuales se transmite la información entre neuronas.

Hay dos tipos principales de aprendizaje en RNA:

- Aprendizaje supervisado. Con este tipo de aprendizaje, proporcionamos a la red un conjunto de datos de entrada y la respuesta correcta. El conjunto de datos de entrada es propagado hacia delante hasta que la activación alcanza las neuronas de la capa de salida. Entonces podemos comparar la respuesta calculada por la red con aquella que se desea obtener, el valor real, objetivo o “blanco” (de *target*, en inglés). Entonces se ajustan los pesos para asegurar que la red produzca de una manera más probable una respuesta correcta en el caso de que se vuelva a presentar el mismo o similar patrón de entrada. Este tipo de aprendizaje será útil especialmente para las tareas de regresión y clasificación.
- Aprendizaje no supervisado. Sólo se proporciona a la red un conjunto de datos de entrada. La red debe auto-organizarse (es decir, auto enseñarse) dependiendo de algún tipo de estructura existente en el conjunto de datos de entrada. Típicamente esta estructura suele deberse a redundancia o agrupamientos en el conjunto de datos. Este tipo de aprendizaje será útil especialmente para las tareas de agrupamiento y reducción de dimensionalidad.

Al igual que otros paradigmas de la Inteligencia Artificial, la faceta más interesante del aprendizaje no es sólo la posibilidad de que los patrones de entrada puedan ser aprendidos/clasificados/identificados sino la capacidad de generalización que posee. Es decir, mientras el aprendizaje tiene lugar en un conjunto de patrones de entrenamiento, una propiedad importante de éste es que la red pueda generalizar sus resultados en un conjunto de patrones de prueba los cuales no han sido vistos durante el aprendizaje. Uno de los problemas a tener en cuenta es el peligro de sobreaprendizaje, denominado más técnicamente “sobreajuste”.

13.3 Aprendizaje supervisado en RNA

Para introducir este tipo de aprendizaje primero presentamos dos de las primeras redes neuronales que lo emplearon en su diseño y posteriormente mostraremos dos de las redes neuronales más usadas basadas en la utilización de éste.

13.3.1 Perceptrón simple y Adaline

El perceptrón simple fue inicialmente investigado por Rosenblatt en 1962 [Rosenblatt, 1962]. El perceptrón simple tiene una estructura de varios nodos o neuronas de entrada y uno o más de salida. Un perceptrón simple, por tanto, no tiene capa oculta y así su estructura es como la red neuronal artificial de la Figura 13.2, pero sin ninguna capa oculta o intermedia.

Asociado a un patrón de entrada particular, x^P , tenemos una salida o^P y un "blanco" o salida correcta t^P . El algoritmo tiene la siguiente forma:

1. La red comienza en un estado aleatorio. Los pesos entre neuronas poseen valores pequeños y aleatorios (entre -1 y 1).
2. Seleccionar un vector de entrada, x^P , a partir del conjunto de ejemplos de entrenamiento.
3. Se propaga la activación hacia delante a través de los pesos en la red para calcular la salida $o^P = w \cdot x^P$.
4. Si $o^P = t^P$ (es decir, si la salida de la red es correcta) volver al paso 2.
5. En caso contrario el cambio de los pesos se realiza atendiendo a la siguiente expresión: $\Delta w_i = \eta x_i^P (t^P - o^P)$ donde η es un número pequeño positivo conocido como coeficiente de aprendizaje. Volver al paso 2.

Lo que se hace, por tanto, es ajustar los pesos de una manera en la que las salidas de la red, o^P , se vayan haciendo cada vez más semejantes al valor de los blancos, t^P , a medida que cada entrada, x_P , se va presentando a la red.

Otra red neuronal importante fue la Adaline (*ADaptive LINear Element*), concebida por Widrow y sus colaboradores en 1960 [Widrow & Hoff 1960]. Su topología es idéntica al perceptrón simple, es decir, no tiene capa oculta, pero la red Adaline calcula sus salidas empleando la siguiente expresión:

$$o = \sum_j w_j x_j + \theta$$

con la misma notación de antes. La diferencia entre esta red y el Perceptrón es la presencia o no de un umbral, θ . El interés en esta red se debió parcialmente al hecho de que se puede implementar fácilmente empleando un conjunto de resistores e interruptores.

La suma del error cuadrático a partir del uso de esta red en todos los patrones de entrenamiento viene dada por la siguiente expresión:

$$E = \sum_p E^P = \frac{1}{2} \sum_p (t^P - o^P)^2$$

y el incremento de los pesos viene dado por su gradiente:

$$\Delta P w_j = -\gamma \frac{\partial E^P}{\partial w_j}$$

donde γ representa el coeficiente de aprendizaje. Esta regla se denomina Error Cuadrático Medio (*Least Mean Square error*, LMS) o regla Delta o de Widrow-Hoff.

Ahora, en el caso del modelo Adaline con una sola salida, o , tenemos:

$$\frac{\partial E^P}{\partial w_j} = \frac{\partial E^P}{\partial o^P} \frac{\partial o^P}{\partial w^P}$$

y debido a la linealidad de las unidades Adaline,

$$\frac{\partial o^P}{\partial w_j} = x_j^P \quad \text{y también: } \frac{\partial E^P}{\partial o^P} = -(t^P - o^P).$$

Por tanto:

$$\Delta_P w_j = \gamma(t^P - o^P) x_j^P$$

Nótese la similitud entre esta regla de aprendizaje y la del perceptrón. Sin embargo, esta regla tiene mayor aplicación ya que se puede usar tanto para neuronas binarias como continuas, es decir, tanto para neuronas cuyas salidas son solamente ceros y unos o aquellas cuya salida son números reales. Es una de las reglas más potentes y se emplea como base de muchos métodos que utilizan aprendizaje supervisado.

El perceptrón simple y el modelo Adaline son redes sin capa intermedia y, por tanto, si ignoramos las funciones de activación, son equivalentes a una función discriminante lineal, como las vistas en el Capítulo 7.

13.3.2 Perceptrón multicapa

Tanto el Perceptrón y el modelo Adaline son métodos potentes de aprendizaje aunque hay algunas situaciones en las que no dan lugar a buenos resultados. Estos casos se caracterizan por ser no linealmente separables (el concepto de separabilidad lineal se trató en el Capítulo 6). Hoy en día es posible mostrar que muchos conjuntos de datos que no son linealmente separables pueden ser modelados mediante el empleo del Perceptrón Multicapa (*Multilayer Perceptron*, MLP), es decir una red neuronal en forma de cascada, que tiene una o más capas ocultas, como la vista en la Figura 13.2.

Aunque esta potencialidad del MLP se descubrió pronto, se tardó bastante tiempo en encontrar un método o regla de aprendizaje apropiada para construirlas a partir de ejemplos. Esta regla parece que fue descubierta de manera independiente varias veces, y no existe acuerdo de la fecha exacta ni de su descubridor, pero fue popularizada principalmente por el Grupo PDP (*Parallel Distributed Processing*) [McClelland et al. 1986], bajo el nombre de Retropropagación o Propagación hacia atrás, que veremos más adelante.

Respecto al uso de la red o de la activación, la activación se propaga en la red a través de los pesos desde la capa de entrada hacia la capa intermedia donde se aplica alguna función de activación a las entradas que le llegan. Entonces la activación se propaga a través de los pesos hacia la capa de salida.

Por tanto, si pensamos en el aprendizaje, hay que actualizar dos conjuntos de pesos: aquellos entre la capa oculta o intermedia y la de salida, y aquellos entre la capa de entrada y la capa intermedia. El error debido al primer conjunto de pesos se calcula empleando el método del error cuadrático medio anteriormente descrito. Entonces se propaga hacia atrás la parte del error debido a los errores que tienen lugar en el segundo conjunto de pesos y se asigna el error proporcional a los pesos que lo causan.

Podemos utilizar cualquier número de capas ocultas que queramos ya que el método es bastante general. Sin embargo, un factor a tener en cuenta es normalmente el tiempo de entrenamiento, el cual puede ser excesivo para arquitecturas con muchas capas. Además, se ha demostrado que redes con una única capa oculta son capaces de aproximar cualquier función continua (o incluso cualquier función con sólo un número finito de discontinuidades), en el caso de utilizar funciones de activación diferenciables (no lineales) en la capa oculta.

13.3.3 Algoritmo de retropropagación

Debido a su importancia, mostramos el algoritmo completo de una manera detallada en la Figura 13.3.

1. Inicializar los pesos a valores pequeños aleatorios.
2. Escoger un patrón de entrada, x^P , y presentarlo a la capa de entrada.
3. Propagar la activación hacia delante a través de los pesos hasta que la activación alcance las neuronas de la capa de salida.
4. Calcular los valores de “ δ ” para las capas de salida $\delta_j^P = (t_j^P - o_j^P) f'(Act_j^P)$ usando los valores de los blancos deseados para el patrón de entrada seleccionado.
5. Calcular los valores de “ δ ” para la capa oculta usando $\delta_i^P = \sum_{j=1}^N \delta_j^P w_{ji} f'(Act_i^P)$.
6. Actualizar los pesos de acuerdo con: $\Delta_P w_{ij} = \gamma \delta_i^P o_j^P$.
7. Repetir del paso 2 al 6 para todos los patrones de entrada.

Figura 13.3. Algoritmo de retropropagación.

Donde $f'(Act^P)$ representa la derivada de la función de activación.

Para especificar completamente el algoritmo anterior es necesario determinar una serie de factores que influyen en el mismo, como, por ejemplo, la manera de proporcionar los ejemplos (en batería o en línea), la inicialización de los pesos iniciales, las funciones de activación, el valor del coeficiente de aprendizaje γ , etc.

Aprendizaje en batería frente al aprendizaje en línea

La convergencia del algoritmo de propagación está garantizada teóricamente si se utiliza el modo “en batería”. Éste tiene lugar cuando todos los ejemplos son presentados a la red a la vez, el error total es calculado y los pesos actualizados en una etapa separada al final de cada época de entrenamiento. Sin embargo, es más común el uso de la versión “en línea” (*online*) o incremental, donde los pesos son actualizados después de la representación individual de cada patrón. Se ha demostrado experimentalmente que este último método permite una convergencia más rápida, teniendo en cuenta que existe una posibilidad teórica de introducir un ciclo de cambios repetidos, es decir, pasamos de un estado A a un estado C pasando por un estado B y de nuevo, de A a B, de B a C, de C a A, etc. Para evitar estos posibles ciclos, los patrones son presentados a la red en un orden aleatorio.

El algoritmo en línea tiene la ventaja de que requiere menos espacio de almacenamiento que el método en batería. Por otra parte el método en batería es más preciso mientras que el algoritmo en línea “zigzaguea” hasta alcanzar el estado final. Se puede mostrar que el cambio esperado en pesos usando el algoritmo en línea es igual al cambio en pesos usando el método en batería.

Funciones de activación

En el algoritmo anterior no se especifica la función de activación utilizada, sólo que su derivada aparece en dos expresiones en los pasos 4 y 5. Las funciones de activación más populares son la función sigmoide:

$$f(s) = \frac{1}{1 + \exp(-bx)}$$

y la función tangente hiperbólica, $\tanh(x)$. Ambas funciones satisfacen el criterio básico de ser diferenciables. Además, ambas son monótonas y tienen la propiedad importante de que su razón de cambio es mayor para valores intermedios y menor para valores extremos. Esto hace posible saturar la salida de una neurona debido a uno u otro de sus valores extremos. Por ejemplo, si la función de activación es la función tangente hiperbólica, ésta no puede nunca tomar valores mayores que "1" o menores que "-1". En la Figura 13.4 se muestra una gráfica de ambas funciones:

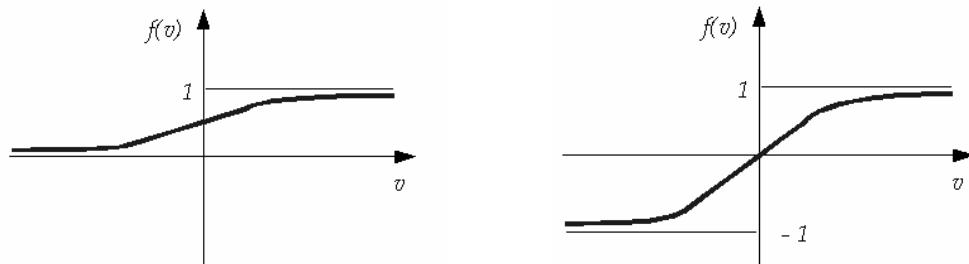


Figura 13.4. Dos funciones de activación usuales: sigmoide y tangente hiperbólica.

El último punto a considerar es la facilidad con la que pueden ser calculadas sus derivadas. La derivada de la función sigmoide es $f'(x)=b \cdot (1-f(x)) \cdot f(x)$ y la de la tangente hiperbólica es: $f'(x)=b \cdot (1-f(x)) \cdot f(x)$.

Se sabe que la convergencia se alcanza antes cuando se emplea la tangente hiperbólica. Hay que tener en cuenta que la función empleada en cada caso debe ser la adecuada. Si tenemos un rango de valores al que queremos aproximarnos, debemos utilizar una capa de salida lineal. Si deseamos realizar, por ejemplo, predicciones que emplean índices con valores próximos a 1.000, no podemos usar ni la función tangente hiperbólica ni la función sigmoide en la capa de salida debido a que la primera da lugar a valores entre -1 y 1 y la función sigmoide entre 0 y 1. Por este motivo debemos usar una capa de salida lineal.

Inicialización de los pesos

El valor inicial de los pesos afecta en muchos casos a los valores finales de convergencia de la red. Si empleamos el método de entrenamiento en batería y consideramos una superficie de energía con un número de pozos de potencial, los valores iniciales de pesos son los únicos elementos estocásticos, es decir no previamente determinados, del régimen de entrenamiento. Para comprender mejor estos conceptos comentamos brevemente la idea de superficie de energía y pozo de potencial. Si imaginamos diferentes valores de la función de energía (o de error o de pérdida) asociada con cada conjunto de valores de los pesos, entonces tenemos una superficie (multidimensional) en el espacio de pesos. Si imaginamos una bola rodando en esta superficie multidimensional, siempre descenderá hacia un pozo de potencial, el cual es una parte de esa superficie que es más profunda que cualquier otra parte de la superficie de su entorno próximo.

Entonces la red convergerá a un valor particular dependiendo del "cuenco" en el cual el vector original esté situado. Hay peligro de que si los valores iniciales de la red son lo suficientemente grandes, la red inicialmente se encontrará en una superficie con un gran mínimo local. Normalmente se comienza con valores aleatorios de los pesos uniformemente distribuidos dentro de un pequeño rango. [Haykin 1998] recomienda el rango:

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i} \right)$$

donde F_i es el número de pesos en la entrada i -ésima.

Momento y velocidad de convergencia

El método de retropropagación básico no es conocido por su rapidez de convergencia. Si simplemente incrementamos la velocidad de aprendizaje, el efecto será el de introducir inestabilidad en la regla de aprendizaje causando oscilaciones violentas en los pesos aprendidos. Es posible acelerar el método básico de varias maneras. Lo más simple es aplicar un término, denominado momento, al cambio de los pesos. La idea básica es incrementar si el nuevo cambio de pesos coincide con la dirección de los cambios anteriores y se reduce en caso contrario. Entonces utilizamos la expresión:

$$\Delta w_{ij}(t+1) = (1 - \alpha)\delta_j o_i + \alpha \Delta w_{ij}(t),$$

donde α determina la influencia del momento. Este término debe estar entre cero y uno.

Mínimo local

En la técnica del descenso del gradiente (visto en el modelo Adaline) se debe tener en cuenta la existencia de mínimos locales, es decir, mínimos en los que el descenso del gradiente es atrapado, sin ser el mínimo global (el mejor resultado), como se observa en la Figura 13.5. Éstos corresponden a soluciones parciales obtenidas por la red neuronal en respuesta a un conjunto de datos de entrenamiento.

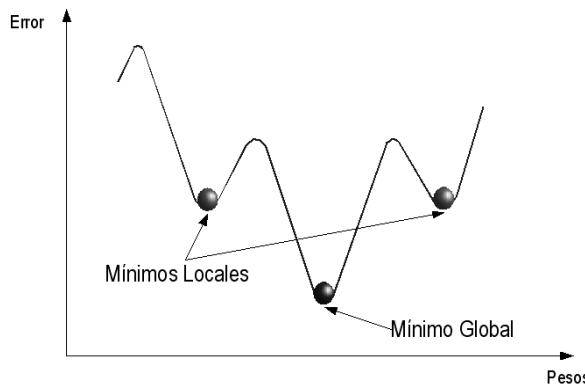


Figura 13.5. Función del error donde se presentan dos mínimos locales y un mínimo global.

Existe la opinión de que los mínimos locales no son un gran problema para las RNA, ya que los pesos de la red convergen típicamente a soluciones, las cuales, incluso si no son óptimas globalmente, son suficientemente buenas. Si se alcanza un mínimo local y el error es satisfactorio, el entrenamiento ha sido un éxito. Hay incluso pequeñas evidencias analíticas que apoyan estas creencias. Algunas heurísticas sugieren inicializar arbitrariamente los pesos de la neurona j -ésima alrededor de un valor $1/\sqrt{N}$, donde N es el número de pesos relacionados con la neurona j -ésima. Una segunda heurística consiste en introducir un ruido pequeño y arbitrario en la red, bien en las entradas o afectando a los cambios en los pesos. Típicamente este ruido se va reduciendo a lo largo de la simulación.

Caída de los pesos y generalización

Aunque deseamos alcanzar el mejor resultado posible durante el entrenamiento, estamos más interesados en el resultado de la red con el conjunto de datos de prueba ya que éste nos da una medida de lo bien que generaliza la red. Hay un balance entre la precisión asociada al conjunto de entrenamiento y la de test. La generalización es importante no sólo porque queremos que la red funcione ante un nuevo conjunto de datos que no ha sido presentado durante el aprendizaje, sino también porque es posible que haya datos que contengan ruido, que estén distorsionados o sean incompletos.

Si una red neuronal tiene un gran número de pesos (cada peso representa un grado de libertad), podemos correr el peligro de sobreajustar (*overfitting*) la red al conjunto de datos de entrenamiento. Esto provocaría la obtención de malos resultados al emplear el conjunto de test. Para evitar este problema podemos eliminar explícitamente conexiones o podemos asociar a cada peso una tendencia de disminución hacia cero. El método más simple consiste en el empleo de la siguiente expresión: $w_{ij}^{new} = (1 - \varepsilon)w_{ij}^{old}$ después de cada actualización de los pesos. Esto tiene la desventaja de que desfavorece el uso de grandes pesos ya que un único peso de gran valor decae probablemente más que muchos pesos pequeños. Hay otros métodos más complejos que favorecen la disminución de pequeños pesos [Haykin 1998].

Número de neuronas ocultas

La experiencia nos puede ayudar a determinar el número adecuado de neuronas en la capa oculta. En general, debe ser menor que el número de patrones de entrenamiento para evitar así problemas de memorización. Para la capa de entrada y salida, debemos tener en cuenta la naturaleza de cada conjunto de datos y el grado de precisión que se espera de la red.

El número de nodos ocultos tiene un particular efecto en la capacidad de generalización de la red. Redes con muchos pesos (muchos grados de libertad) tienden a memorizar el conjunto de datos mientras que redes con muy pocos pesos no son capaces de desempeñar las tareas que se les encomiendan. Por ello muchos algoritmos se diseñan para dar lugar a redes neuronales con un número menor de neuronas ocultas.

13.3.4 Variaciones de la retropropagación y otros MLP generales

El algoritmo de retropropagación es el primer algoritmo eficaz para el aprendizaje de MLP. Aunque hemos visto cómo se puede variar ligeramente el algoritmo mediante el cambio de parámetros y del número de neuronas ocultas, existen muchas otras aproximaciones para el aprendizaje de MLP que han ido apareciendo desde entonces.

Uno de los aspectos más complejos es determinar el número de neuronas ocultas, ya que pocas neuronas pueden hacer que el concepto a aprender no se pueda capturar mientras que demasiadas neuronas pueden sobreajustar los datos. Existen variantes del algoritmo de retropropagación que van añadiendo neuronas ocultas a medida que se van necesitando, o ajustan el tamaño por un análisis de los datos, o bien siguen reglas más o menos sencillas para hacerlo. Muchas implementaciones en paquetes de minería de datos incluyen estas variaciones, ya que son mucho más sencillas de cara al usuario.

Por ejemplo, el algoritmo de retropropagación (*WEKA.classifiers.neural.NeuralNetwork*) del sistema WEKA (véase el Apéndice A) permite, entre otras opciones, la selección del

momento ("momentum"), de la velocidad de convergencia (mediante los parámetros "learning rate" y "decay"), el número de épocas de aprendizaje ("trainingTime"), la inicialización de la semilla aleatoria ("randomSeed"), etc. Todos estos parámetros tienen valores por defecto y un usuario novel sólo debería modificarlos en el caso de que no obtenga buenos resultados con ellos. Quizá, lo más curioso son las reglas y opciones que utiliza para determinar el número de nodos en la capa oculta. Aunque este valor se puede especificar numéricamente de una manera directa, también se puede expresar según el número de atributos y de clases, ya que se ha comprobado que, en muchas ocasiones, la cantidad de nodos necesarios depende de estos factores. Por ejemplo, algunas de estas opciones especiales en la versión 3.2 de WEKA son: 'a', que genera un número de neuronas ocultas igual a $(at + cl) / 2$, donde at es el número de atributos (entradas) y cl el número de valores de la clase, 'i', que genera un número de neuronas ocultas igual a (at) , 'o', que genera un número de neuronas ocultas igual a (cl) y 't', que genera un número de neuronas ocultas igual a $(at + cl)$. Pueden parecer unas reglas muy triviales, pero la configuración por defecto, la 'a', suele dar buenos resultados en general.

Existen otras herramientas que facilitan todavía más la tarea de especificar los parámetros. Por ejemplo, en el sistema SPSS Clementine (véase el Apéndice A) el nodo "TrainNet" incluye cinco métodos de redes neuronales, entre ellas cuatro variantes de MLP, que son:

- "Quick" (rápido). Este método determina el número de nodos internos necesarios mediante una heurística y los deja fijos durante todo el aprendizaje.
- "Dynamic" (dinámico). Este método crea una topología inicial y va añadiendo o eliminando unidades ocultas según va avanzando el aprendizaje.
- "Multiple" (múltiple). Este método crea varias redes con topologías diferentes y las va entrenando simultáneamente. Finalmente se escoge la que tiene menos error.
- "Prune" (recortar). Este método comienza con una red con muchos nodos internos y va eliminando (recortando) las unidades internas (y también las de entrada) que son menos relevantes.

También se pueden modificar otros parámetros generales (velocidad de convergencia, tiempo de aprendizaje, semillas, etc.) u otros más específicos, mediante la opción "experto".

Estas son sólo dos muestras de que, aunque existen paquetes avanzados de redes neuronales con muchas variantes y parámetros para sacar todo el partido a los MLP, también existen otros sistemas que facilitan en gran medida la elección de topologías y parámetros, que son de capital importancia en el MLP.

13.3.5 Ejemplo

La principal diferencia en cuanto a aplicación entre el perceptrón simple y el MLP es que el primero sólo se puede aplicar a casos linealmente separables [Minsky & Papert 1969/1988]. En el caso de problemas no linealmente separables se emplea el MLP.

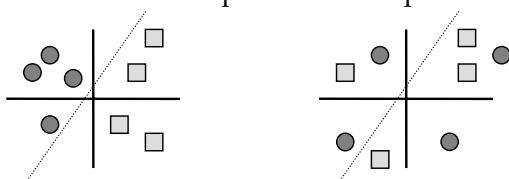


Figura 13.6. (Izquierda) Clases linealmente separables. (Derecha) Clases no linealmente separables.

Así podemos decir que las clases que aparecen en la Figura 13.6 (izquierda) son linealmente separables, mientras que las que aparecen en la Figura 13.6 (derecha) son no linealmente separables.

Como ejemplo hemos tomado una distribución formada por 100 puntos elegidos de manera aleatoria a partir de una distribución cuadrada uniforme. Aquellos puntos de la distribución situados por encima de la recta $X+Y > 0,6$ tienen como salida el valor “1”. El resto de puntos o casos tienen asociado el valor “0”. Una vez que el perceptrón simple es entrenado, es capaz de hacer una clasificación (separación lineal) de las dos clases como podemos ver en la Figura 13.7. Este experimento se ha implementado usando la “Toolbox” de redes neuronales de Matlab utilizando los siguientes parámetros: dos nodos en la capa de entrada y uno en la salida, un coeficiente de aprendizaje de 0,05 y el número de iteraciones fue 100.000. En esta figura aparecen las dos clases separadas por una línea. En la gráfica, el signo “+” representa a los unos y los círculos a los ceros en los datos de entrenamiento. En los datos de validación los cuadros representan los unos y los triángulos representan a los ceros. Como se puede observar, el separador lineal clasifica perfectamente los datos.

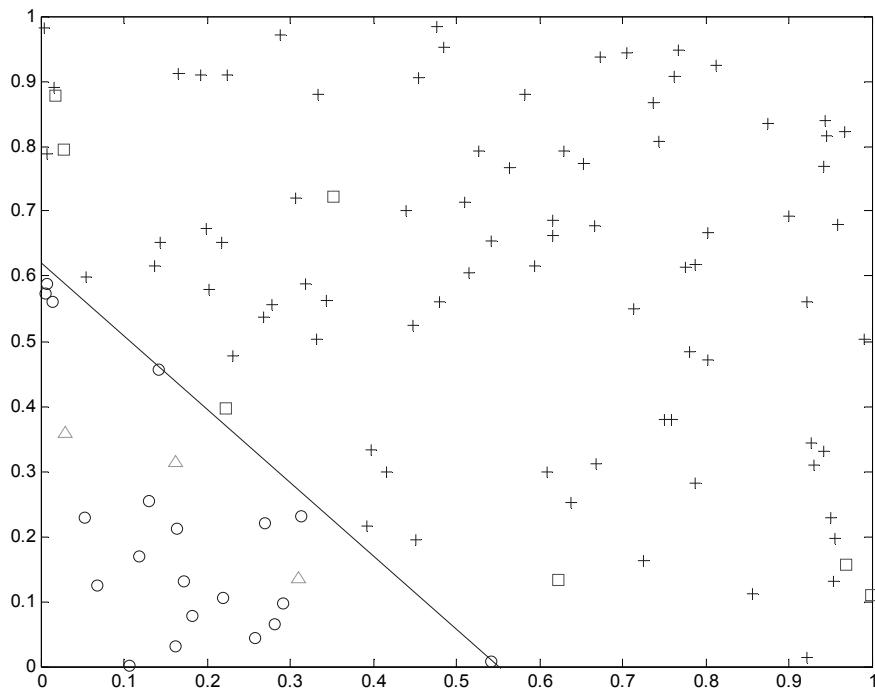


Figura 13.7. Resultados obtenidos empleando el Perceptrón simple.

Como en el caso anterior se seleccionan 100 puntos de manera aleatoria. En este caso aquellos puntos o casos que están en el límite marcado por $X+Y > 0,6$ y $X+Y < 1,6$ tienen asociados un valor de salida igual a “1”, mientras que para el resto su salida corresponde a “0”. Éste es un caso de separación no lineal, por tanto el Perceptrón simple no debe ser utilizado. Los resultados obtenidos empleando un MLP se muestran en la Figura 13.8, donde se puede observar la correcta clasificación. Este experimento se ha realizado empleando la “Toolbox” de redes neuronales de Matlab utilizando los siguientes parámetros: dos nodos en la capa de entrada, diez en la intermedia y dos de salida. El

coeficiente de aprendizaje fue de 0,05 y el número de iteraciones fue 1.000.000. En este ejemplo, de nuevo el signo "+" representa a los unos y los círculos a los ceros en el entrenamiento. En los datos de validación los cuadros representan los unos y los triángulos representan a los ceros.

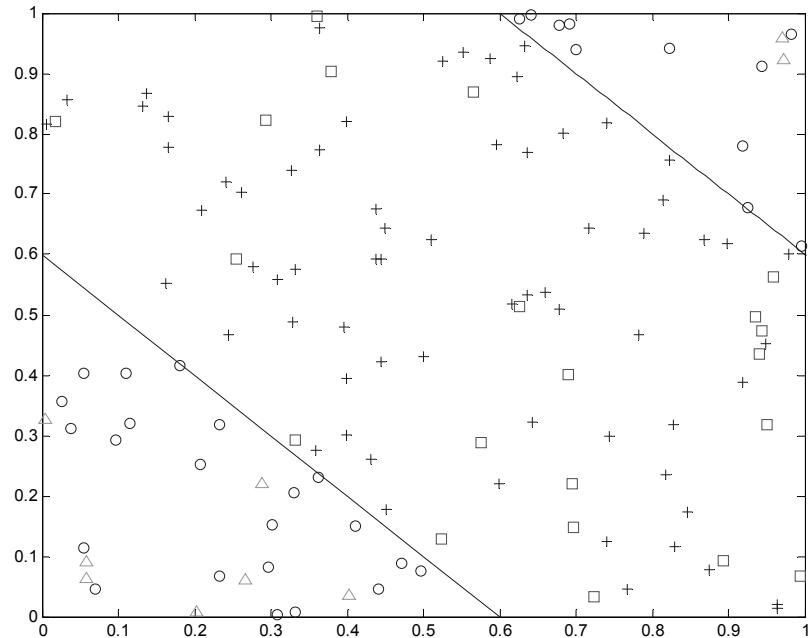


Figura 13.8. Clasificación obtenida empleando el MLP.

Como se observa, este ejemplo no lineal (en realidad formado por dos discriminantes lineales) sí que puede ser capturado por un MLP.

13.3.6 Funciones de base radial

En una red de Funciones de Base Radial (*Radial Basis Function*, RBF) típica, la capa de entrada actúa como receptor para el conjunto de datos de entrada. La característica más importante de las RBF es el uso de una función de cálculo (y no una función de activación tradicional) en las neuronas de la capa oculta, como se puede ver en la Figura 13.9.

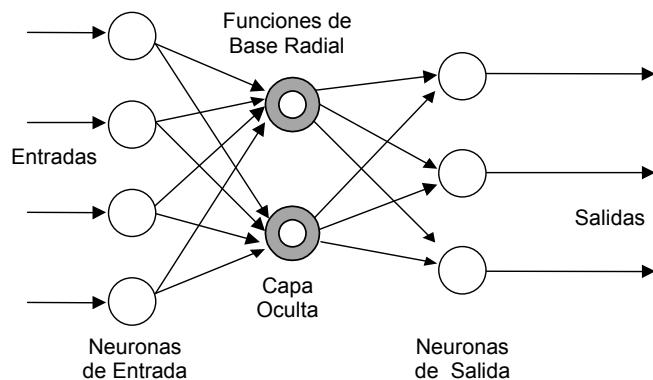


Figura 13.9. Ejemplo de Red de Funciones de Base Radial.

Esta función lleva a cabo una transformación no lineal desde el espacio de entrada al espacio de la capa intermedia. Las neuronas de la capa intermedia son las funciones base para los vectores de entrada y las neuronas de la capa de salida simplemente calculan una combinación lineal de las salidas de las neuronas ocultas. Las funciones que se usan con frecuencia en la capa intermedia son las funciones gaussianas. Su media y desviación estándar deben ser determinadas de alguna manera a partir del conjunto de datos de entrada. Si el patrón de entrada viene representado por x , y M es el número de neuronas ocultas, entonces $\phi(x)$ es el vector de las salidas de las neuronas ocultas, el cual viene representado por la siguiente expresión $\phi(x) = (\phi_1(x), \phi_2(x), \dots, \phi_M(x))^T$, en donde cada $\phi_i(x) = \exp(-\lambda_i \|x - c_i\|^2)$.

Los centros c_i de las gaussianas vendrán determinados por los datos de entrada. Los términos $\|x - c_i\|$ representan la distancia Euclídea entre las entradas y el centro i -ésimo. La activación de la red se calcula mediante la siguiente expresión:

$$y = w \cdot \phi(x) = w^T \phi(x)$$

donde w es el vector de pesos entre las neuronas ocultas y la neurona de salida.

Se puede entrenar la red usando la minimización del error cuadrático medio. Podemos actualizar los pesos después de presentar el patrón de entrada i -ésimo, x^i , mediante la siguiente expresión:

$$\Delta w_k = -\frac{\partial E^i}{\partial w_k} = e^i \cdot \phi_k(x^i)$$

donde E^i son los errores cuadráticos obtenidos. El método de minimización del error cuadrático medio se basa en minimizar el error existente entre la salida de la red y la salida esperada para un vector cualquiera de entrada. También se pueden actualizar los pesos mediante métodos rápidos de actualización en modo batería, ya que la operación que tiene lugar después de la activación de las funciones de base radial es totalmente lineal. [Haykin 1998] ha investigado muchísimo este tipo de redes.

Comparación con el perceptrón multicapa

Tanto las RBF como el MLP son excelentes aproximadores universales, es decir, pueden modelar funciones continuas de una manera muy próxima. Hay por supuesto algunas diferencias importantes:

- Todas las neuronas de un MLP generalmente calculan la misma función de sus entradas ponderadas, es decir, la misma función de activación, por ejemplo, la función logística. En las funciones de base radial, las neuronas ocultas realizan una proyección no lineal, mientras que la capa de salida es siempre lineal.
- La no linealidad en el MLP es generalmente monótona. En RBF usamos una función que decrece radialmente.
- En el MLP se emplea el vector producto, $w \cdot x$, de las entradas y los pesos, mientras que en el caso de RBF, el argumento es la distancia entre la entrada y el centro de la función de base radial, $\|x - w\|$.
- El MLP realiza un cálculo global mientras que las RBF obtienen la suma de las salidas locales. El MLP posee mayor capacidad para calcular soluciones en regiones del espacio de entrada donde hay escasez de datos en el conjunto de entrenamiento.

Si se requieren resultados precisos en todo el espacio de entrenamiento, es necesario el empleo de muchas funciones de base radial, es decir, muchas neuronas ocultas en una red RBF. Sin embargo, debido a la naturaleza local del modelo, las RBF son menos sensivas al orden en el que se le presentan los datos.

- El MLP debe retropropagar el error para cambiar los pesos progresivamente. Las RBF no hacen esto y por tanto son más rápidas de entrenar.

Selección de los centros para las RBF

La elección de los centros c_i de las gausianas es uno de los puntos importantes en el proceso de entrenamiento de la red. Si tenemos pequeños conjuntos de datos, no tenemos otra opción que situar los centros de nuestras funciones de base radial sobre los propios "puntos" del conjunto de datos. Sin embargo, esto puede dar lugar a una pobre generalización. Si el conjunto de entrenamiento es mayor hay varias posibilidades:

1. Elegir de forma aleatoria los centros de las funciones a partir del conjunto de datos de entrenamiento.
2. Asociar a cada centro una función de base radial de acuerdo con el algoritmo denominado *K medias* ("Kmeans"). Éste es un algoritmo no supervisado de agrupamiento donde k es el número de grupos que se desea encontrar y se corresponde en nuestro caso con el número de neuronas de la capa oculta. Este algoritmo se ve en el Capítulo 16.
3. Emplear el error cuadrático medio. En este caso no se garantiza la convergencia debido a que la función de error E no es convexa con respecto a los centros y puede haber algún mínimo local. La expresión utilizada sería:

$$\Delta c_i = -\frac{\partial E}{\partial c_i}$$

La solución 2 es tan frecuente que muchas veces las implementaciones de RBF asumen esta segunda opción.

13.3.7 Aplicaciones y ejemplo

Las RNA con aprendizaje supervisado son aplicadas a campos muy diversos y de forma creciente. Se utilizan para problemas de clasificación, regresión, compresión de datos, optimización, control, etc. De hecho, el MLP funciona igualmente bien para problemas de clasificación (con una función de activación en la salida) o de regresión (sin esta función de activación en la salida). Es precisamente para problemas de regresión con características no lineales donde las redes neuronales son una buena alternativa a los métodos de regresión generalizados y los no paramétricos vistos en los capítulos 7 y 8.

Respecto a los campos de aplicación son todos aquellos en los que puedan hacer falta modelos supervisados con alta expresividad. Por ejemplo, las redes neuronales se han utilizado en muy diversos campos, como la predicción de mercados financieros, la clasificación de dígitos escritos a mano, control de robots, la teledetección, etc.

Vamos a ver un ejemplo con el conjunto de datos *census-income* (también conocido como *adult database*, véase el Apéndice B) que está formado por 48.844 personas del censo estadounidense. El objetivo es determinar a partir de 14 atributos (nivel de estudios, país

de origen, tipo de empleo, raza, sexo, etc.) si el individuo gana más de 50.000 dólares anuales. Los datos están divididos en un conjunto de entrenamiento de 32.562 ejemplos y 16.282 de conjunto de test. Vamos a utilizar la herramienta Clementine (véase el Apéndice A para más detalles), que proporciona tanto versiones dinámicas del perceptrón multicapa, como de las RBF.

Uno de los problemas a abordar en este caso es el coste computacional debido al gran número de atributos y de ejemplos. Esto se agrava debido a que algunos atributos tienen muchos valores posibles y las redes neuronales obligan a su numerización (por ejemplo, el campo 14, el país de origen, tiene 43 valores posibles). Esto hace que este atributo genere 43 nuevos atributos. Conjuntamente, de los 14 atributos originales de entrada se pasa a 116 atributos numéricos. Por todo esto, el método del perceptrón multicapa, en el modo más eficiente de los que proporciona el Clementine ("rápido"), es relativamente lento. El RBF requiere todavía más tiempo y es preferible poner un límite de tiempo (por ejemplo, cinco minutos).

Los resultados de los modelos obtenidos se muestran en la siguiente tabla:

MLP ("RAPIDO")	RBF
Capa de entrada : 116 neuronas	Capa de entrada : 116 neuronas
Capa oculta nº1 : 11 neuronas	Capa oculta : 20 neuronas
Capa de salida : 3 neuronas	Capa de salida : 3 neuronas
Precisión pronosticada: 84,96%	Precisión pronosticada: 81,87%
Importancia relativa de las entradas	Importancia relativa de las entradas
campo14 : 0,27073	campo14 : 0,21017
campo4 : 0,16920	campo2 : 0,13608
campo7 : 0,16118	campo4 : 0,13548
campo8 : 0,14015	campo7 : 0,13463
campo9 : 0,13695	campo6 : 0,12746
campo6 : 0,13003	campo8 : 0,09937
campo2 : 0,11432	campo3 : 0,04874
campo1 : 0,10356	campo11 : 0,03242
campo10 : 0,09846	campo13 : 0,03237
campo12 : 0,05714	campo9 : 0,02744
campo5 : 0,03295	campo12 : 0,02617
campo3 : 0,02659	campo5 : 0,02383
campo11 : 0,01833	campo1 : 0,01402
campo13 : 0,00239	campo10 : 0,00896

Como podemos observar, el primer modelo, en este caso, parece mejor que el segundo (84,96 por ciento respecto a 81,87 por ciento). De hecho, si obtenemos la precisión con el conjunto de test, obtenemos un resultado del perceptrón multicapa de 84,53 por ciento de precisión, mientras que el RBF da una precisión de 82,25 por ciento.

Lo más interesante de la tabla anterior es que de los pesos de la red neuronal se derivan fácilmente la importancia relativa de las variables y es, por tanto, un modo de analizar dicha importancia (como los métodos de selección de variables vistos en la Sección 5.4.2). En el caso anterior vemos precisamente que el atributo más relevante es el campo14 (el país de origen), que es precisamente el que tiene más valores. Una solución para mejorar los resultados de este problema sería, por ejemplo, agrupar el país de origen por continentes, así tendríamos redes neuronales con muchas menos neuronas de entrada.

13.4 Aprendizaje no supervisado en RNA

El aprendizaje no supervisado se caracteriza por descubrir modelos o características significativas a partir únicamente de los datos de entrada. No existe, por tanto, como en el aprendizaje supervisado, un valor de salida o una clase para cada instancia, con la que poder contrastar la diferencia (obtener un error).

Los humanos son capaces de aprender sin supervisión explícita. El objetivo del aprendizaje no supervisado es imitar este aspecto de la capacidad humana. Este tipo de aprendizaje tiende a usar métodos biológicamente plausibles a diferencia de aquellos que emplean métodos basados en el descenso del gradiente como en las secciones anteriores. La red debe auto-organizarse y, para hacer esto, debe actuar ante algunos aspectos del conjunto de entrada como pueden ser la existencia de redundancia o grupos en el conjunto de datos. Por tanto, debe haber algún tipo de estructura en los datos ante la cual pueda responder.

Para el aprendizaje no supervisado con redes neuronales, se utilizan dos métodos principalmente:

- Aprendizaje de Hebb, que se usa para obtener proyecciones o compresiones óptimas de conjuntos de datos, de modo similar al análisis de componentes principales.
- Aprendizaje competitivo, que se utiliza principalmente para agrupar conjuntos de datos (*clustering*).

A continuación estudiaremos cada uno de ellos.

13.4.1 Aprendizaje de Hebb

El aprendizaje de Hebb se denomina así debido a Donald Hebb [Hebb 1949] quien en 1949 conjeturó: "Cuando el axón de una célula A está próximo a excitar una célula B y repetida o persistentemente toma parte en su activación, un proceso de crecimiento o cambio de metabolismo tiene lugar en una o ambas células de manera que la eficiencia de la neurona A como una de las células que participa en la activación de B, aumenta".

Si consideramos una red con propagación hacia delante, lo anterior puede ser interpretado como que el peso entre una neurona de entrada y otra de salida se fortalece cuando la activación de la neurona de entrada alcanza la neurona de salida y la activa. Podemos ver cómo la regla favorece al más fuerte: si los pesos entre las entradas y las salidas son grandes (de manera que la entrada tendrá un gran efecto en las salidas) la probabilidad de que los pesos crezcan es grande.

Más formalmente, consideremos el caso simple de una red neuronal artificial con propagación hacia delante, la cual posee un conjunto de neuronas de entrada con un vector de entrada asociado, x , y un conjunto de neuronas de salida con el vector de salida asociado, y . Esta red no tiene capa oculta, como se puede ver a partir de las siguientes expresiones que la definen. Por tanto tenemos:

$$y_i = \sum_j w_{ij} x_j$$

donde ahora la regla de Hebb se define mediante la siguiente expresión: $\Delta w_{ij} = \eta x_j y_i$. Por tanto, el incremento de los pesos entre cada neurona de entrada y salida es proporcional a la magnitud de la activación simultánea de estas neuronas.

Si sustituimos en la regla de aprendizaje de Hebb, $\Delta w_{ij} = \eta x_j y_i$, el valor de las salidas de la red, y_i , calculado mediante la primera de las dos ecuaciones anteriores, obtenemos la siguiente expresión para la regla de aprendizaje:

$$\Delta w_{ij} = \eta x_j y = \eta x_j \sum_k w_{ik} x_k = \eta \sum_k w_{ik} x_k x_j$$

Al escribir la regla de aprendizaje de esta forma se resaltan las propiedades estadísticas que aparecen en ella. La regla identifica la correlación entre diferentes partes de las componentes de los vectores del conjunto de datos de entrada. Esta regla tiene problemas de estabilidad debido al uso de la realimentación positiva. Los pesos crecen sin límite si no se toman algunas medidas preventivas. Éstas incluyen las siguientes:

1. Recortar los pesos, es decir, que haya un rango establecido por un máximo y un mínimo en los valores de los pesos, $[w_{\max}, w_{\min}]$ dentro del cual éstos deben permanecer.
2. Normalización específica de los pesos después de cada actualización, es decir, $\Delta w_{ij} = \eta x_j y_i$ y entonces:

$$w_{ij} = \frac{w_{ij} + \Delta w_{ij}}{\sqrt{\left(\sum_k w_{ik} + \Delta w_{ik}\right)^2}}.$$

3. Tener un término de reducción o “caída” de los pesos en la regla de aprendizaje para detener el crecimiento exagerado, es decir, $\Delta w_{ij} = \eta x_j y_i - \gamma(w_{ij})$, donde la función de caída $\gamma(w_{ij})$ representa una función monótona creciente de los pesos.
4. Crear una red con realimentación negativa.

Esta última red neuronal se utiliza en la siguiente sección.

Arquitectura con realimentación negativa

El Análisis de Componentes Principales (*Principal Component Analysis*, PCA) se vio en el Capítulo 4, como una técnica de reducción de dimensionalidad. El PCA se puede ver también como una técnica estadística de compresión lineal de un conjunto de datos. En [Fyfe 1993; Fyfe 1995a], durante los últimos años, se ha investigado el uso de una red neuronal con realimentación negativa para la implementación de PCA definida por las ecuaciones siguientes:

$$\begin{aligned} y_i &= \sum_{j=1}^N w_{ij} x_j \\ e_j &= x_j - \sum_{i=1}^M w_{ij} y_i \\ y_i &= \sum_{j=1}^N w_{ij} x_j, \forall i \end{aligned}$$

Partimos de un vector de entradas N -dimensional, x , y un vector de salidas M -dimensional, y , con w_{ij} siendo los pesos que unen la entrada j -ésima con la salida i -ésima. El coeficiente

de aprendizaje, η , es un pequeño parámetro que disminuye hasta cero a lo largo del período de entrenamiento de la red. La activación que pasa de las entradas a las salidas a través de los pesos se describe mediante la primera de las ecuaciones anteriores. Ésta se realimenta a través de los pesos desde las salidas, y el error e se calcula para cada dimensión de entrada, como se ilustra en la segunda ecuación. Finalmente los pesos son actualizados usando aprendizaje simple de Hebb, como se ve en la tercera ecuación.

Esta red se ha modificado a lo largo del tiempo para realizar agrupamientos en los que se conserva la topología [Fyfe 1995b; Corchado & Fyfe 2002a], para realizar Análisis Factorial [Charles & Fyfe 1998; Fyfe & Charles 1999] o para realizar Búsqueda de Proyecciones Exploratorias [Fyfe 1997; Fyfe & Baddeley 1995; Corchado & Fyfe 2002; Corchado et al. 2003]. Vamos a desarrollar este último punto.

La Búsqueda de Proyecciones Exploratorias (*Exploratory Projection Pursuit*, (EPP)) [Friedman 1987] es un nombre genérico para un conjunto de técnicas diseñadas para obtener proyecciones que revelen estructura “interesante” en conjuntos de datos multidimensionales a partir del estudio de momentos estadísticos de orden superior como el índice de *skewness* (orden tres) o el de *kurtosis* (orden cuatro). Ésta es la principal diferencia, por ejemplo, con las técnicas como el análisis de componentes principales (PCA) que se basa en el estudio del momento de orden dos o varianza de la distribución. La estructura se mantiene con frecuencia a través de los límites de las múltiples dimensiones de los conjuntos de datos. Una manera de revelar esta estructura es proyectar los datos en un espacio de menor dimensión y entonces estudiar este nuevo espacio. Este estudio se realiza a simple vista mediante el propio ojo humano. Por tanto debemos determinar cuál es el mejor subespacio para proyectar el conjunto de datos. La proyección típica de un conjunto de datos de alta dimensión viene caracterizada por una distribución gaussiana [Diaconis & Freedman 1984]. En ella la estructura existente es escasa. Esto ha permitido a los investigadores sugerir que se deben buscar proyecciones lo más alejadas posibles de la gaussiana. Entonces se define típicamente un índice de lo “interesante” que es una proyección en términos de lo alejado que ésta esté de la distribución gaussiana. Debido a que la distribución gaussiana está totalmente definida por sus dos momentos de orden uno y dos (la media y la varianza), se blanquean los datos. Al blanquear los datos se suprime la información relacionada con estos momentos para que no interfiera en el estudio de momentos de orden superior. Esto permite tener un “campo de juego” donde determinar lo alejada que puede estar una cierta proyección de una distribución gaussiana. El blanqueamiento consiste en trasladar el conjunto de datos inicial hasta que su media sea cero. Después se proyecta sobre las direcciones de los vectores propios o componentes principales y se multiplica por la inversa de la raíz cuadrada de sus valores propios para obtener un conjunto de datos que tiene media cero y su varianza es la unidad en todas sus direcciones. Esta técnica sólo se puede aplicar sobre el conjunto de datos cuando no estemos interesados en los dos momentos de orden inferior, es decir, la media y la varianza. Por tanto el blanqueamiento no se puede aplicar a un conjunto de datos si queremos realizar, por ejemplo, Análisis de Componentes Principales, debido a que éste está muy relacionado con el estudio de la varianza de la distribución como hemos indicado anteriormente.

Ejemplos y aplicaciones

Vamos a trabajar con el conjunto de datos denominado “iris” (véase el Apéndice B). Este conjunto de datos corresponde a las medidas en centímetros de la longitud y anchura del pétalo y sépalo de 150 flores de tres especies diferentes: Iris setosa (s), Iris versicolor (v) e Iris virginia (g). Vamos a utilizar una red con realimentación negativa para llevar a cabo Análisis de Componentes Principales. Esta red neuronal, cuya arquitectura se mostró en la sección anterior, está implementada usando Matlab, con un coeficiente de aprendizaje empleado 0,001 y el número de iteraciones es 10.000.

En la Figura 13.10 se puede ver la proyección obtenida por la red, donde se muestran los individuos sobre los ejes de los dos primeros componentes principales obtenidos por la red y no con los cuatro atributos originales. Con un umbral sobre la primera componente (en -1 por ejemplo), esta red es capaz de identificar claramente una de las especies, (s), en la proyección mostrada en la Figura 13.10. Las otras dos clases, (v, g) no son linealmente separables, por tanto esta técnica no da lugar a una buena identificación de ellas.

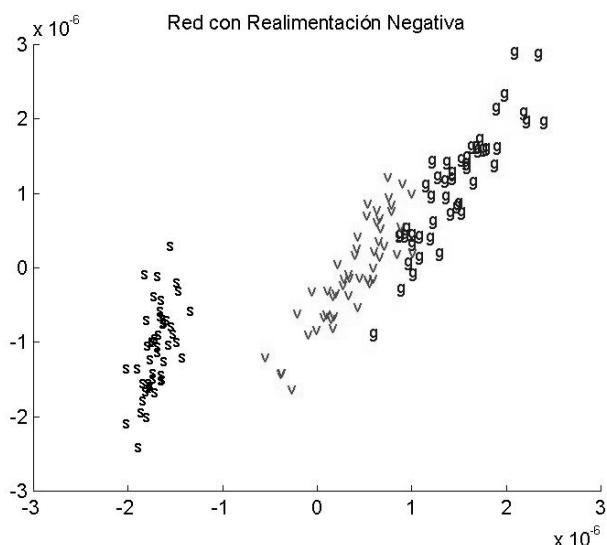


Figura 13.10. Proyección del conjunto de datos “iris” sobre los dos primeros componentes principales.

Este tipo de red neuronal fue utilizado para implementar EPP [MacDonald 2001] en teledetección, datos financieros y datos de censos. También se ha aplicado [Girolami & Fyfe 1996a; Girolami & Fyfe 1996b; Corchado et al. 2002] para resolver el difícil problema denominado *cocktail-party problem* o en castellano “el problema del cóctel”. Éste se trata de la extracción de una sola señal a partir de una mezcla de ellas cuando partimos de poca información sobre las señales, el ruido presente o el método empleado para la mezcla de las señales. En este último problema las señales deben ser no-gaussianas (o como mucho sólo una puede ser gaussiana) y el método de mezcla de las señales es lineal.

13.4.2 Aprendizaje competitivo

Uno de los aspectos menos biológicos de la regla básica de aprendizaje de Hebb es que no hay límite sobre la cantidad de fuentes que alimentan una sinapsis. Esto está reñido con el crecimiento de las neuronas reales ya que se cree que hay límite en el número y eficiencia

de las sinapsis de cada neurona. En otras palabras, hay un momento en el aprendizaje en el que si una sinapsis resulta fortalecida, otra debe ser debilitada. Esto se modela normalmente mediante un proceso de competición. En el aprendizaje competitivo tiene lugar una competición entre las neuronas de la capa de salida para decidir cuál se activa. Estas neuronas de la capa de salida se denominan con frecuencia *winner-take-all* o, en castellano, “la ganadora-toma-todo”.

El objetivo del aprendizaje competitivo es agrupar conjuntos de datos. En este tipo de aprendizaje, al igual que en la regla de Hebb, no se proporciona la respuesta correcta a la red (es decir, no hay información etiquetada), ya que estamos tratando de aprendizaje no supervisado. Se debe auto-organizar en base a la estructura existente en el conjunto de datos de entrada. El mecanismo básico del aprendizaje competitivo simple es el encontrar una neurona ganadora y actualizar los pesos para hacer que ésta tenga más probabilidad de ganar cuando una entrada similar se presente a la red. Despues de la competición entre neuronas de la capa de salida se calcula el incremento de pesos para cada iteración de la siguiente forma:

$$\Delta w_{ij} = \eta(x_j - w_{ij}), \text{ para la neurona ganadora } i.$$

Como se puede observar en la expresión anterior, la actualización de los pesos es una función de la diferencia entre los pesos y la entrada. Esta regla “desplazará” los pesos de la neurona ganadora directamente hacia la entrada. En términos de una distribución, los pesos tienden a la media de la distribución ya que $\Delta w_{ij} \rightarrow 0 \Leftrightarrow w_{ij} \rightarrow \bar{x}_j$, donde \bar{x} indica el promedio.

Probablemente, las tres variaciones más importantes del aprendizaje competitivo son:

- *Learning Vector Quantisation* (LVQ) [Kohonen 1984].
- Modelos ART (*Adaptive Resonance Theory*) [Gail & Grossberg 1987; Gail 1990].
- Mapas auto-organizados [Kohonen 1995].

LVQ es una técnica supervisada, es decir, para clasificación. Este modelo se estudia en el Capítulo 16. En cambio, las otras dos técnicas tienen como objetivo la identificación de grupos (*clustering*) y se aplican a campos como el reconocimiento de voz, reconocimiento de patrones, control, recuperación de documentos, visualización, etc.

A continuación, nos centraremos en el estudio del último de los tres casos.

13.4.3 Mapa de características o auto-organizados de Kohonen

El interés por este tipo de mapas de características, mapas auto-organizados o mapas auto-organizativos (*Self-Organizing Maps*, SOM) viene dado por su importancia biológica y por su utilidad práctica. El espacio de características se basa en la “disposición física” de las neuronas de salida para modelar algunas características del espacio de entrada. En particular, si dos entradas, x_1 y x_2 están próximas entre sí con respecto a alguna medida en el espacio de entradas, y causan la activación de las neuronas de salida y_a e y_b respectivamente, entonces y_a e y_b deben estar próximos entre sí respecto a algún tipo de composición o disposición de las neuronas de salida. Más aún, podremos decir que lo opuesto se debe mantener. Es decir, si y_a e y_b están próximas en la capa de salida, entonces las entradas que las producen deben estar también próximas en el espacio de entrada. Cuando se cumplen estas condiciones estamos ante un mapa de características. Estos

estas condiciones estamos ante un mapa de características. Estos mapas también se denominan **mapas que preservan la topología**.

Algunos ejemplos biológicos de estos tipos de mapas son:

- el mapa retinotópico, cuya entrada procede de la retina (en los ojos) y proyecta en el córtex visual (parte trasera del cerebro).
- el mapa somatosensorio, la entrada se corresponde con los centros de tacto de la piel y proyecta en el córtex somatosensorio.
- el mapa tonotópico, que proyecta las respuestas de nuestro oído en el córtex auditivo.

Se considera que cada uno de estos mapas está determinado genéticamente, pero modelado mediante el uso. Así por ejemplo, el mapa retinotópico es muy diferente si un ojo es excluido del proceso de visión durante períodos particulares del desarrollo.

El algoritmo de Kohonen [Kohonen 1995] es bastante sencillo. La red está constituida por dos capas y tiene lugar una competición entre las neuronas de la capa de salida, como se ilustra en la Figura 13.11:

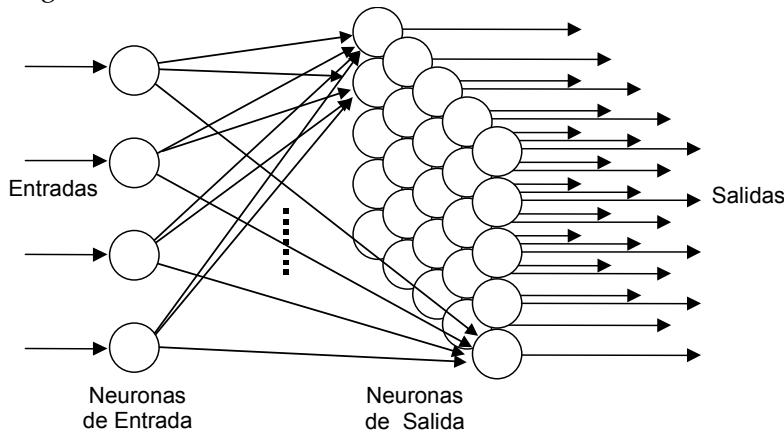


Figura 13.11. Ejemplo de un mapa auto-organizativo de Kohonen.

Sin embargo, ahora no sólo los pesos relacionados con la neurona ganadora son actualizados sino que también los pesos de las neuronas vecinas. Kohonen definió una función de vecindario, $f(i, i^*)$, para la neurona ganadora i^* . Esta función de vecindario es una función de la distancia entre la neurona i -ésima y la ganadora i^* .

Una función típica es la diferencia de gaussianas o, por su forma, también denominada jocosamente, Sombrero Mexicano, la cual se muestra en la Figura 13.12. Esta función es muy útil ya que permite incrementar los pesos de las neuronas próximas a la ganadora, (excitación), y reducir los pesos de aquellas que están a cierta distancia (inhibición). Los pesos de las neuronas que están más alejadas no sufren cambios, de ahí la forma de la función.

Como podemos ver en la Figura 13.12, el número de neuronas afectadas por esta función disminuye a medida que transcurre el aprendizaje.

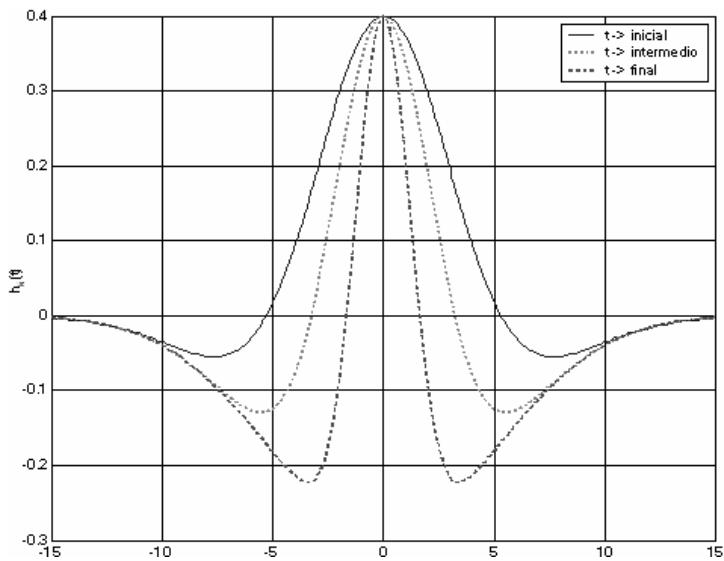


Figura 13.12. Diferencia de Gaussianas o Sombrero Mexicano.

Entonces si la unidad i -ésima está situada en el punto r_i en la capa de salida, entonces

$$f(i, i^*) = a \exp\left(\frac{-|r_i - r_{i^*}|^2}{2\sigma_1^2}\right) - b \exp\left(\frac{-|r_i - r_{i^*}|^2}{2\sigma_2^2}\right)$$

donde r_k es la posición del centro k -ésimo en el espacio de neuronas. Si el espacio de neuronas es de una sola dimensión, como en el caso de la Figura 13.13, una elección típica es que $r_k = k$; si el espacio es bidimensional, como en el caso de la Figura 13.14, $r_k = (x_k, y_k)$, se corresponde con el espacio de coordenadas cartesianas en dos dimensiones.

Ejemplos

Hemos realizado un primer experimento empleando un conjunto de datos artificiales. Éste consiste en el empleo de una red neuronal con dos entradas y 25 neuronas de salida, en una organización unidimensional en un primer momento y bidimensional (10×10) después. Las dos entradas, en cada iteración, se generan mediante una distribución uniforme a partir de un cuadrado, $[-1 \text{ a } 1]$, en sus dos dimensiones. El algoritmo empleado es el siguiente:

1. Seleccionar un punto aleatorio de ese “cuadrado”.
2. Se produce una competición entre las neuronas de la capa de salida. Aquella neurona cuyos pesos están más próximos a la entrada gana la competición:

$$\text{Neurona ganadora, } i^* = \arg \min \left(\|\mathbf{x} - \mathbf{w}_i\| \right)$$

3. Ahora actualizamos los pesos de todas las neuronas empleando la siguiente regla de aprendizaje:

$$\Delta w_{ij} = \alpha (x_j - w_{ij}) \cdot f(i, i^*)$$

donde $f(i, i^*)$ es el sombrero mexicano comentado anteriormente.

4. Volvemos al comienzo hasta que algún criterio de finalización sea alcanzado.

Kohonen normalmente mantiene el coeficiente de aprendizaje constante durante aproximadamente las primeras 1.000 iteraciones y, a partir de ahí, disminuye lentamente

hasta cero durante el resto del experimento (el número de iteraciones siempre depende de las características del conjunto de datos).

En la Figura 13.13 se muestra el mapa resultante al aplicar un mapa de una sola dimensión (25 neuronas) a un conjunto de datos bidimensional procedente de una distribución uniforme con forma de cuadrado, $[-1, 1], [-1, 1]..$ Entradas típicas podrían ser $(0,2, 0,6)$, $(-0,3, 0,8)$, $(0,6, -0,6)$ o $(-0,1, -0,5)$.

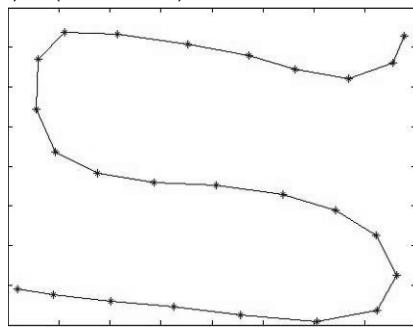


Figura 13.13. Mapa de una dimensión de un espacio de entrada de dos dimensiones.

En la figura anterior se observa cómo el mapa ha realizado una partición del espacio de entrada distribuyendo los 25 nodos de una manera aproximadamente uniforme a través del espacio, es decir cada nodo intenta cubrir una zona del espacio bidimensional original.

En la Figura 13.14 se muestra un típico ejemplo de los pesos asociados a un mapa auto-organizado cuando el espacio de neuronas es bidimensional, es decir, ahora organizamos las 100 neuronas en una estructura de 10×10 nodos. Los nodos están distribuidos en el espacio de entrada en función de la densidad del mapa. En este caso se emplea el mismo conjunto de datos artificial que en el caso anterior.

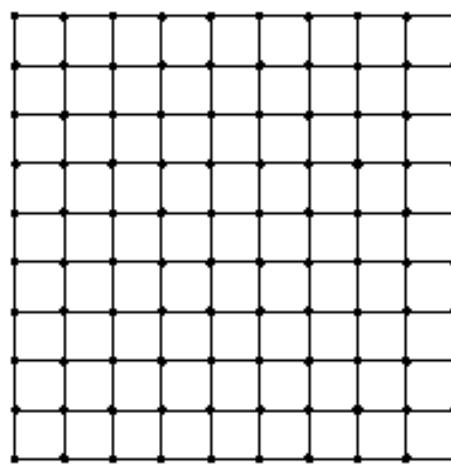


Figura 13.14. Pesos entrenados de un mapa de Kohonen de dos dimensiones.

En esta última figura se observa cómo los nodos del mapa autoorganizado se distribuyen de una manera aproximadamente uniforme para cubrir todo el espacio.

Quizás el interés de estos mapas se produce cuando los datos de origen tienen más de dos dimensiones y mediante un mapa de Kohonen conseguimos representarlo en dos dimensiones. Para verlo, pasemos a ver un ejemplo con datos reales. Hemos aplicado esta

misma red (aunque ahora con 225 neuronas de salida, organizadas en una matriz 15×15) para analizar el conjunto de datos denominado *Wisconsin Diagnostic Breast Cancer*. El resultado se puede observar en la Figura 13.15.

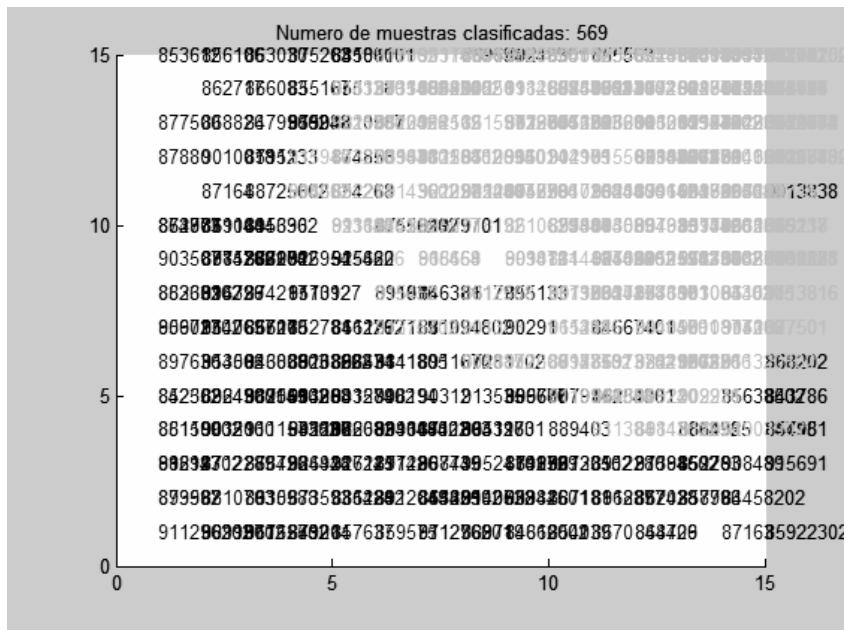


Figura 13.15. SOM aplicada sobre el conjunto de datos denominado “Wisconsin Diagnostic Breast Cancer”.

En la Figura 13.15 podemos ver cómo la red realiza un buen agrupamiento. En este caso lo podemos ver de una manera clara, porque estamos visualizando el atributo que determina si el cáncer es benigno o no (que no se ha utilizado en las entradas) en dos colores. Por tanto, podemos comprobar que se conserva la topología, ya que los casos benignos (en gris) aparecen, la gran mayoría, todos juntos en un lado mientras que los malignos (negro) aparecen en el otro.

En [Kohonen 1995], se dedica un capítulo entero a diferentes aplicaciones de los mapas auto-organizados. Algunos ejemplos se refieren a campos como la minería de datos, especialmente en lo relacionado con la recuperación de textos, páginas web, clasificación de variables financieras, control de robots y predicción, etc.

13.5 Sistemas, aplicabilidad y recomendaciones de uso

En este capítulo, hemos revisado algunos de los principales tipos de redes neuronales artificiales. Hemos sugerido que el principal rasgo diferenciador en el campo de las redes neuronales artificiales se encuentra entre aquellas redes que emplean aprendizaje supervisado, aplicables, por tanto, a las tareas de clasificación y regresión, y aquellas dedicadas al aprendizaje no supervisado, utilizables, fundamentalmente, para el agrupamiento, la reducción de dimensionalidad o la transformación de atributos.

Para ponerse manos a la obra, existen infinidad de paquetes y herramientas (muchas de ellas gratuitas) para utilizar redes neuronales. Por ejemplo, un simulador de redes neuronales muy conocido es el denominado SNNS (Simulador de Redes Neuronales de

Stuttgart) (<http://www-ra.informatik.uni-tuebingen.de/SNNS/>). Su objetivo es la creación de un entorno eficiente y flexible que permita simular la actuación de diferentes redes neuronales. También recomendamos la “Toolbox” de redes neuronales de Matlab (<http://www.mathworks.com/products/neuralnet/>) para el estudio, implementación y aplicación de diferentes arquitecturas neuronales. Otras herramientas específicas de redes neuronales son NeuralPlanner, NeuronalDiet o Easy NN (<http://www.easynn.com/>), que se comentan en el Apéndice A.

Respecto a las herramientas generales que incorporan técnicas de redes neuronales, hemos comentado anteriormente, WEKA y Clementine (también comentados en el Apéndice A). En realidad, la mayoría de paquetes genéricos de minería de datos (*suites*), incorporan al menos retropropagación y, muchos de ellos, como por ejemplo Clementine, también los mapas de Kohonen (SOM) o los RBF.

Estos paquetes genéricos son recomendables para los principiantes en redes neuronales, ya que permiten utilizar parámetros y topologías por defecto, con resultados muy aceptables. Las herramientas específicas son más recomendables cuando ya se tiene un mayor conocimiento y experiencia sobre redes neuronales y se desea obtener todo su potencial, mediante una elección precisa de los algoritmos, las topologías y los parámetros.

Para finalizar, hemos de dejar constancia que lo presentado en este capítulo es una introducción a toda una disciplina, la de las redes neuronales, en la que se pueden encontrar muchas más técnicas y variantes de las que hemos presentado aquí, que son, a nuestro parecer, las más usuales y útiles en minería de datos. Para ampliar información sobre redes neuronales artificiales, tanto de cómo sacar el máximo partido a las técnicas vistas aquí, como iniciarse en otras técnicas, se recomiendan los libros: [Isasi & Galván 2003] y [Haykin 1998].

Capítulo 14

MÁQUINAS DE

VECTORES SOPORTE

Xavier Carreras, Lluís Márquez y Enrique Romero

Los fundamentos teóricos de las máquinas de vectores soporte (SVM, del inglés *Support Vector Machines*) se encuentran en los trabajos de Vapnik y otros autores sobre la teoría del aprendizaje estadístico, desarrollados a finales de los años 70 y durante los 80. El modelo de SVM, tal como se entiende actualmente, fue presentado en la conferencia COLT (*COmputational Learning Theory*) del año 1992 por el mismo Vapnik, junto con Boser y Guyon [Boser et al. 1992] y descrito con más detalle posteriormente en [Cortes & Vapnik 1995; Vapnik 1998], posibilitando el salto de la formulación teórica a su aplicación práctica en problemas reales de reconocimiento de formas (*pattern recognition*). A partir de ese momento el interés por este modelo de aprendizaje no ha parado de crecer hasta el presente. Como consecuencia, tanto su desarrollo teórico (introduciendo conexiones con otros modelos de aprendizaje) como el campo de las aplicaciones reales han experimentado un avance muy importante, hasta el punto que, hoy en día, las SVM constituyen un referente completamente establecido para las disciplinas del aprendizaje automático y de la minería de datos. Los campos donde las SVM han sido aplicadas con éxito incluyen, entre otros, la visión por computador, la bioinformática, la recuperación de información, el procesamiento del lenguaje natural y el análisis de series temporales. Es interesante comentar que la comunidad científica que trabaja actualmente en el estudio y desarrollo de SVM es marcadamente interdisciplinaria puesto que incluye aspectos y técnicas del aprendizaje automático, optimización, estadística, análisis funcional, etc.

14.1 Introducción

Las máquinas de vectores soporte pertenecen a la familia de los *clasificadores lineales* puesto que inducen separadores lineales o hiperplanos en espacios de características de muy alta

dimensionalidad (introducidos por funciones núcleo o *kernel*) con un sesgo inductivo muy particular (maximización del margen).

En primer lugar, recordemos que todo hiperplano en un espacio D -dimensional, \mathbb{R}^D , se puede expresar como $h(x) = \langle w, x \rangle + b$, donde $w \in \mathbb{R}^D$ es el vector ortogonal al hiperplano, $b \in \mathbb{R}$ y $\langle \cdot, \cdot \rangle$ expresa el producto escalar habitual en \mathbb{R}^D . Visto como un clasificador binario, la regla de clasificación se puede expresar como: $f(x) = \text{signo}(h(x))$, donde la función signo se define como:

$$\text{signo}(x) = \begin{cases} +1 & \text{si } x \geq 0 \\ -1 & \text{si } x < 0 \end{cases}$$

En la terminología de clasificación, las $x \in \mathbb{R}^D$ son representaciones vectoriales de los ejemplos, con una componente real por cada atributo, y el vector w se suele denominar *vector de pesos*. Este vector contiene un peso para cada atributo indicando su importancia o contribución en la regla de clasificación. Finalmente, b suele denominarse *sesgo (bias)* y define el umbral de decisión. Dado un conjunto binario (es decir, con dos clases) de datos (ejemplos, vectores o puntos) linealmente separables⁴⁰, existen diversos algoritmos incrementales (*on-line*) para construir hiperplanos (w, b) que los clasifiquen correctamente. Podemos citar, por ejemplo: *Perceptron*, *Widrow-Hoff*, *Winnow*, *Exponentiated-Gradient*, *Sleeping Experts*, etc. (algunos de ellos vistos en el Capítulo 13). A pesar de que esté garantizada la convergencia de todos ellos hacia un hiperplano solución, las particularidades de cada algoritmo de aprendizaje pueden conducirnos a soluciones ligeramente distintas, puesto que puede haber varios (de hecho infinitos) hiperplanos que separan correctamente el conjunto de ejemplos.

Suponiendo que el conjunto de ejemplos es linealmente separable, ¿cuál es el “mejor” hiperplano separador en términos de generalización? La idea que hay detrás de las SVM *de margen máximo* consiste en seleccionar el hiperplano separador que está a la misma distancia de los ejemplos más cercanos de cada clase. De manera equivalente, es el hiperplano que maximiza la distancia mínima (o *margen geométrico*) entre los ejemplos del conjunto de datos y el hiperplano. Intuitivamente, este hiperplano está situado en la posición más neutra posible con respecto a las clases representadas por el conjunto de datos, sin estar sesgado, por ejemplo, hacia la clase más numerosa. Además, sólo considera los puntos que están en las fronteras de la región de decisión, que es la zona donde puede haber dudas sobre a qué clase pertenece un ejemplo (son los denominados *vectores soporte*).

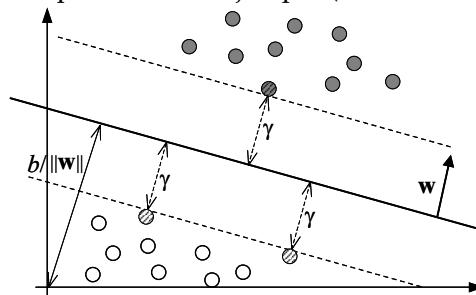


Figura 14.1. Hiperplano (w, b) equidistante a dos clases, margen geométrico (γ) y vectores soporte (puntos rayados).

⁴⁰ El concepto de separabilidad y, en particular, la separabilidad lineal, se ha visto en el Capítulo 6.

En la Figura 14.1 se presenta geométricamente este hiperplano equidistante (o de margen máximo) para el caso bidimensional.

Este sesgo inductivo de aprendizaje consistente en maximizar el margen se justifica dentro de la teoría del aprendizaje estadístico y se enmarca en el principio de *Minimización del Riesgo Estructural* [Vapnik 1995]. En esta teoría, maximizar el margen geométrico se ve como una buena heurística para minimizar la “complejidad” de la clase de hiperplanos separadores, que, a su vez, interviene directamente en las expresiones que acotan superiormente el error de generalización. A nivel práctico, el hiperplano separador de margen máximo ha demostrado una muy buena capacidad de generalización en numerosos problemas reales, así como una robustez notable frente al sobreajuste o *overfitting* (véase la Sección 6.4.1).

A nivel algorítmico, el aprendizaje de las SVM representa un problema de optimización con restricciones que se puede resolver usando técnicas de programación cuadrática (QP). La convexidad garantiza una solución única (esto supone una ventaja con respecto al modelo clásico de redes neuronales) y las implementaciones actuales permiten una eficiencia razonable para problemas reales con miles de ejemplos y atributos.

El aprendizaje de separadores no lineales con SVM se consigue mediante una transformación no lineal del espacio de atributos de entrada (*input space*) en un espacio de características (*feature space*) de dimensionalidad mucho mayor y donde sí es posible separar linealmente los ejemplos. El uso de las denominadas **funciones núcleo** (*kernel functions*), que calculan el producto escalar de dos vectores en el espacio de características, permite trabajar de manera eficiente en el espacio de características sin necesidad de calcular explícitamente las transformaciones de los ejemplos de aprendizaje. El aprendizaje en espacios de características vía transformaciones no lineales por medio de funciones núcleo no es exclusiva del paradigma SVM. Aunque se suele asociar los métodos basados en funciones núcleo con las SVM, al ser su ejemplo más paradigmático y más avanzado, hay muchos otros algoritmos que pueden “kernelizarse” para permitir el aprendizaje de funciones no lineales. Éste es el caso, por ejemplo, del perceptrón, de los discriminantes de Fisher, del análisis de componentes principales, etc., métodos que se han visto en otros capítulos de este libro.

Un requisito básico para aplicar con éxito las SVM a un problema real es la elección de una función núcleo adecuada, que debe reflejar el conocimiento a priori sobre el problema. El desarrollo de funciones núcleo para estructuras no vectoriales (por ejemplo, estructuras secuenciales, árboles, grafos, etc.) es actualmente una importante área de investigación con aplicación en dominios como el procesamiento del lenguaje natural y la bioinformática, entre otros.

A veces, los ejemplos de aprendizaje no son linealmente separables ni tan siquiera en el espacio de características. Otras veces no es deseable conseguir un separador perfecto del conjunto de aprendizaje, puesto que los datos de aprendizaje no están libres de errores (ejemplos mal etiquetados, valores de atributos mal calculados, inconsistencias, etc.), comportamientos excepcionales (*outliers*), etc. Focalizarse demasiado en *todos* los ejemplos de aprendizaje puede comprometer seriamente la generalización del clasificador aprendido por culpa del sobreajuste. En estos casos es preferible ser más conservador y admitir algunos ejemplos de aprendizaje mal clasificados a cambio de tener separadores más generales y prometedores. Este comportamiento se consigue mediante la introducción del

modelo de SVM con margen blando (*soft margin*). En este caso, la función objetivo a minimizar está compuesta por la suma de dos términos: el margen geométrico y un término de regularización que tiene en cuenta los ejemplos mal clasificados. La importancia relativa de los dos términos se regula mediante un parámetro, normalmente llamado C (véase la Sección 14.2.3). Este modelo, aparecido en 1995, es el que realmente abrió la puerta a un uso real y práctico de las SVM, aportando robustez frente al ruido.

14.1.1 Contenidos del capítulo

Dado el carácter introductorio del capítulo, los contenidos desarrollados son necesariamente parciales. Por ejemplo, sólo estudiaremos inicialmente las SVM para problemas binarios de clasificación, dejando para la Sección 14.5 una enumeración de posibles extensiones y temas avanzados para el lector interesado en profundizar.

El núcleo del capítulo se sitúa en la Sección 14.2, donde se describen los modelos básicos de SVM para clasificación binaria: SVM de margen máximo (o *hard margin*) en el caso linealmente separable, introducción de funciones núcleo para obtener clasificadores no lineales e introducción de los modelos de margen blando (*soft margin*) para tratar con robustez problemas reales, por ejemplo, con ruido en los datos de aprendizaje. La Sección 14.3 es complementaria y pretende dar una justificación, desde el punto de vista de la teoría del aprendizaje estadístico, del principio de *maximización del margen*. Es importante notar que el desarrollo que se hace de los modelos de SVM en la Sección 14.2 presupone un cierto conocimiento de la teoría de optimización con restricciones lineales. Los elementos de esta teoría usados en la sección se describen en un anexo al final de este capítulo. Finalmente, la Sección 14.4 introduce algunas aplicaciones de las SVM, poniendo énfasis en el problema de la clasificación de documentos textuales, lo que permite definir una función núcleo sobre secuencias para trabajar con una representación tipo tira de caracteres (o *string*) de los documentos. La Sección 14.5 trata de varias extensiones, entre ellas algunas para poder tratar problemas multiclas, problemas de regresión y de agrupamiento. La Sección 14.6 recoge unas breves referencias sobre dónde encontrar herramientas *software* de SVM y algunas recomendaciones de uso.

14.2 Máquinas de vectores soporte para clasificación binaria

En esta sección veremos los principales ingredientes de las SVM para clasificación binaria: la maximización del margen y el uso de funciones núcleo (*kernel functions*). La resolución del problema de optimización planteado siguiendo la idea de la maximización del margen tendrá como consecuencia la aparición de los vectores soporte, que dan nombre a las SVM.

Para fijar la notación, consideremos el problema de clasificación binaria dado por un conjunto de N datos $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, donde cada x_i pertenece a un cierto espacio de entrada X , y cada y_i pertenece a $\{-1, +1\}$ e indica la clase a la que pertenece x_i . A los elementos de X los denominaremos ejemplos o vectores. Para simplificar, supondremos que $X \subseteq \mathbb{R}^D$, donde D es la dimensión del espacio de entrada. En los casos en que los datos se presenten de manera diferente (habitualmente estructuras discretas de tipo secuencia, árbol o grafo), también es posible trabajar con SVM diseñando funciones núcleo específicas

para el problema (véanse el apartado dedicado a las SVM en el espacio de características y la Sección 14.4 este capítulo).

En primer lugar se describirán las SVM lineales con margen máximo que servirán como introducción a modelos más complejos y robustos. Los conocimientos básicos sobre problemas de optimización con restricciones lineales necesarios para seguir el desarrollo de esta sección se encuentran en el anexo, al final del capítulo.

14.2.1 SVM lineal con margen máximo

La SVM lineal con margen máximo (*maximal margin linear SVM*) es el modelo más sencillo e intuitivo de SVM, aunque también el que tiene condiciones de aplicabilidad más restringidas, puesto que parte de la hipótesis de que el conjunto de datos es linealmente separable en el espacio de entrada. Aun así, contiene muchas de las ideas subyacentes en la teoría de las SVM y es la base de todas las demás.

Supongamos que el conjunto de datos es linealmente separable en el espacio de entrada. Es decir, sin hacer ninguna transformación de los datos, los ejemplos pueden ser separados por un hiperplano de manera que en cada lado del mismo sólo hay ejemplos de una clase. En términos matemáticos, es equivalente a decir que existe un hiperplano $h: \mathcal{X} \rightarrow \mathbb{R}$ tal que $h(x) > 0$ para los ejemplos de la clase +1 y $h(x) < 0$ para los ejemplos de la clase -1. De manera más concisa, h cumple que $y_i \cdot h(x_i) > 0$ para todo i entre 1 y N , es decir, para todos los ejemplos.

Formulación original de las SVM

Recordemos que la idea que hay detrás de las SVM consiste en seleccionar el hiperplano separador que está a la misma distancia de los ejemplos más cercanos de cada clase (véase la Figura 14.1). Es muy fácil ver que la distancia de un vector x a un hiperplano h , definido por (ω, b) como $h(x) = \langle \omega, x \rangle + b$, viene dada por la fórmula $dist(h, x) = |h(x)| / \|\omega\|$, donde $\|\omega\|$ es la norma en \mathbb{R}^D asociada al producto escalar (es decir, $\|\omega\|^2 = \langle \omega, \omega \rangle$). Así pues, el hiperplano equidistante a dos clases es el que maximiza el valor mínimo de $dist(h, x)$ en el conjunto de datos. Además, dados dos puntos z_1 y z_2 equidistantes a un hiperplano, se cumple que $b = -(\langle \omega, z_1 \rangle + \langle \omega, z_2 \rangle) / 2$. Como el conjunto es linealmente separable, podemos reescalar ω y b de manera que la distancia de los vectores más cercanos al hiperplano sea $1/\|\omega\|$ (al multiplicar ω y b por una constante, la distancia no varía). Como consecuencia, los vectores z más cercanos tendrán $|h(z)| = 1$, mientras que para el resto $|h(z)| \geq 1$. De manera que el problema de encontrar el hiperplano equidistante a dos clases se reduce a encontrar la solución del siguiente problema de optimización con restricciones:

$$\begin{aligned} \text{Maximizar} \quad & \frac{1}{\|\omega\|} \\ \text{sujeto a:} \quad & y_i(\langle \omega, x_i \rangle + b) \geq 1 \quad 1 \leq i \leq N \end{aligned}$$

El *margen funcional* de un ejemplo (x, y) con respecto a una función f se define como $y \cdot f(x)$, mientras que el *margen geométrico* (o normalizado) de un hiperplano se define como $1/\|\omega\|$. Con estas definiciones, la solución de la SVM lineal con margen máximo es el hiperplano que maximiza el margen geométrico, restringido a que el margen funcional de cada ejemplo sea mayor o igual que 1.

La formulación más habitual de la SVM lineal con margen máximo en su *formulación original (primal form)*, equivalente a la anterior, es la siguiente:

$$\begin{aligned} \text{Minimizar} \quad & \frac{1}{2} \langle \omega, \omega \rangle \\ \text{sujeto a:} \quad & y_i(\langle \omega, x_i \rangle + b) \geq 1 \quad 1 \leq i \leq N \end{aligned}$$

Como se puede observar, es un problema de optimización convexa, consistente en minimizar una función cuadrática bajo restricciones en forma de desigualdad lineal. En esta formulación original, el hiperplano queda caracterizado por un vector de pesos ω con una componente por cada atributo que indica la importancia relativa de cada atributo en el hiperplano solución.

Formulación dual de las SVM

Es posible obtener una *formulación dual (dual form)* equivalente a la de la SVM de margen máximo en la que la solución se expresa como una combinación lineal de los vectores de aprendizaje. Su desarrollo se explica a continuación. Aplicando los resultados descritos en el anexo (al final de este capítulo) al problema de la SVM lineal con margen máximo tenemos:

$$L(\omega, b, \alpha) = \frac{1}{2} \langle \omega, \omega \rangle + \sum_{i=1}^N \alpha_i (1 - y_i(\langle \omega, x_i \rangle + b))$$

Derivando con respecto a las variables originales,

$$\begin{aligned} \frac{\delta L(\omega, b, \alpha)}{\delta \omega} = 0 \quad & \Rightarrow \quad \omega - \sum_{i=1}^N y_i \alpha_i x_i = 0 \quad \Rightarrow \quad \omega = \sum_{i=1}^N y_i \alpha_i x_i \\ \frac{\delta L(\omega, b, \alpha)}{\delta b} = 0 \quad & \Rightarrow \quad \sum_{i=1}^N y_i \alpha_i = 0 \end{aligned}$$

Sustituyendo las dos últimas relaciones en la función de Lagrange generalizada, se obtiene:

$$\begin{aligned} L(\omega, b, \alpha) &= \frac{1}{2} \langle \omega, \omega \rangle + \sum_{i=1}^N \alpha_i (1 - y_i(\langle \omega, x_i \rangle + b)) = \\ &= \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle + \sum_{i=1}^N \alpha_i = \\ &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \end{aligned}$$

De aquí, el siguiente resultado nos da una solución analítica y exacta del problema:

Teorema (SVM lineal con margen máximo). Supongamos que el conjunto de datos es linealmente separable en el espacio de entrada. Sea α^* una solución del problema dual

$$\begin{aligned} \text{Maximizar} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ \text{sujeto a} \quad & \sum_{i=1}^N y_i \alpha_i = 0 \\ & \alpha_i \geq 0 \quad 1 \leq i \leq N \end{aligned}$$

Entonces, el vector

$$\omega^* = \sum_{i=1}^N y_i \alpha_i^* x_i$$

es el vector ortogonal al hiperplano con margen (geométrico) máximo. La SVM lineal con margen máximo es

$$h(x) = \langle \omega^*, x \rangle + b^* = \sum_{i=1}^N y_i \alpha_i^* \langle x_i, x \rangle + b^*$$

donde

$$b^* = -\frac{1}{2} \left(\max_{y_j=-1} \{ \langle \omega^*, x_j \rangle \} + \min_{y_j=+1} \{ \langle \omega^*, x_j \rangle \} \right)$$

y su clasificador asociado es $f(x) = \text{signo}(h(x))$.

Una de las consecuencias más importantes del teorema, expresada en la relación

$$\omega^* = \sum_{i=1}^N y_i \alpha_i^* x_i$$

indica que el vector ortogonal al hiperplano se puede representar como una combinación lineal de los N ejemplos del conjunto de datos. Por otro lado, la condición de Karush-Kuhn-Tucker (véase el anexo al final de este capítulo) se traduce como

$$\alpha_i^* (1 - y_i (\langle \omega^*, x_i \rangle + b^*)) = 0 \quad 1 \leq i \leq N$$

De esta relación se desprende que sólo algunas de las α_i^* son diferentes de 0. Por tanto, en realidad sólo algunos de los ejemplos del conjunto de datos forman parte de la combinación lineal que da lugar a ω^* : son los llamados ejemplos o *vectores soporte*, que dan nombre a las SVM. Obsérvese que si todos los demás vectores fueran eliminados del conjunto de aprendizaje, la SVM lineal de margen máximo resultante sería exactamente la misma. Los vectores soporte son aquellos para los cuales $\alpha_i^* \neq 0$ y se cumple que están en la frontera de la región de decisión. Como consecuencia directa de la condición de Karush-Kuhn-Tucker, los vectores soporte tienen margen funcional 1 con respecto a la SVM lineal con margen máximo. El resto de ejemplos tiene margen funcional mayor que 1, puesto que cumplen las condiciones del problema original.

Otra de las ventajas de este teorema con respecto a la formulación original tiene que ver con su aplicación práctica: aunque en teoría son equivalentes, el problema de optimización dual es más fácil de resolver, con las técnicas actualmente conocidas, que el del problema original. Las implementaciones actuales basadas en este teorema son muy eficientes.

La expresión de la solución final (como suma de productos escalares con los vectores soporte) es una consecuencia de la manera en la que se resuelve el problema asociado, a diferencia de otros modelos cuyas soluciones también se expresan como sumas de funciones simples (los modelos clásicos de redes neuronales, por ejemplo, vistos en el Capítulo 13), en los que la expresión de la solución está prefijada a priori.

Una buena solución, especialmente para conjuntos de ejemplos voluminosos, es aquella en que la proporción de vectores soporte con respecto al número total de vectores N es pequeña. Sólo así se puede conseguir un factor de compresión elevado en la solución de la SVM (solución dispersa con respecto al número de vectores de aprendizaje). Nótese, sin

embargo, que en la formulación original de las SVM no es posible controlar la dispersión de la solución de manera explícita.

Ejemplo. En la Figura 14.2 podemos ver un ejemplo de SVM lineal de margen máximo. Para identificarlos mejor, cada punto del conjunto de datos tiene coordenadas enteras. En línea continua se indica la solución de margen máximo, mientras que en línea discontinua tenemos la solución de mínimos cuadrados, es decir la que minimiza la distancia media de *todos* los ejemplos al discriminante (véase la Sección 4.3.1). Los vectores soporte son: (1, 4), (10, 6) y (10, 3). Obsérvese que, de acuerdo con la teoría, los vectores soporte están a la misma distancia del hiperplano solución: $4x/27 - 2y/3 + 41/27 = 0$. Obsérvese también que el margen funcional de los vectores soporte es 1.

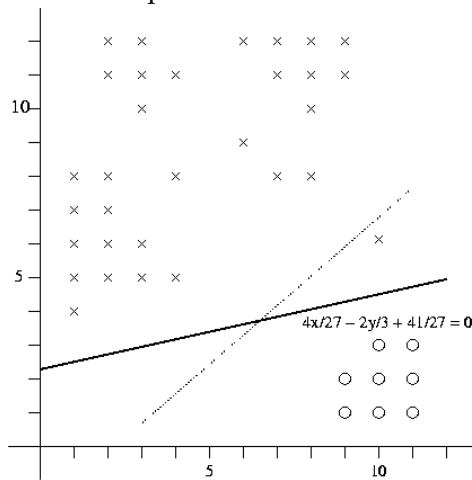


Figura 14.2. Ejemplo de SVM de margen máximo.

Restricciones

La SVM lineal con margen máximo tiene dos restricciones importantes. En primer lugar, el clasificador resultante es lineal. Es bien conocido que la mejor manera de representar muchos problemas no es un modelo lineal (véase, por ejemplo, el problema presentado en la Figura 14.3).

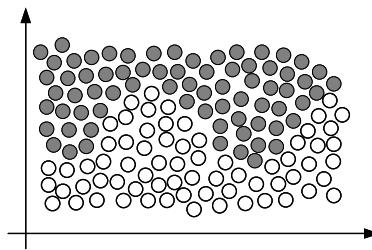


Figura 14.3. Conjunto no linealmente separable.

En segundo lugar, necesita que el conjunto de datos sea linealmente separable, cosa que no tiene por qué ser cierta o fácil de conseguir. De hecho, aunque el problema sea lineal, la existencia de ruido puede hacer que no sea separable linealmente o que no sea conveniente separarlo por el hiperplano de margen máximo (véase la Figura 14.5 más adelante). Las dos subsecciones siguientes están dedicadas a explicar cómo eliminar estas restricciones manteniendo la misma filosofía subyacente a la SVM lineal con margen máximo.

14.2.2 SVM con margen máximo en el espacio de características

En la Figura 14.3 se puede ver un conjunto de datos que no es linealmente separable, en el que la SVM lineal con margen máximo no es la mejor solución. Para evitar la restricción de linealidad de la solución resultante, observemos que todo lo desarrollado en la sección anterior para la SVM lineal con margen máximo sólo depende de la existencia de un producto escalar en el espacio de entrada: hiperplanos, distancias, derivadas del producto escalar, etc. La SVM (no lineal) con margen máximo en el espacio de características se basa en la idea de hacer una transformación no lineal del espacio de entrada a un espacio dotado de un producto escalar (lo que se conoce en álgebra como un espacio de Hilbert [Berberian 1961]). En este espacio se pueden aplicar los mismos razonamientos que para la SVM lineal con margen máximo. Dicho de otro modo, supongamos que existe una transformación no lineal del espacio de entrada en un cierto *espacio de características* \mathfrak{I} :

$$\Phi : \mathbb{R}^D \rightarrow \mathfrak{I}$$

$$x \rightarrow \Phi(x)$$

dotado de un producto escalar $\langle \Phi(x), \Phi(y) \rangle$ (\mathfrak{I} es un espacio de Hilbert). Si el conjunto de datos es linealmente separable en \mathfrak{I} (con los hiperplanos definidos a partir del producto escalar correspondiente), entonces la SVM con margen máximo en el espacio de características se puede obtener sustituyendo en la SVM lineal con margen máximo $\langle x, y \rangle$ por $\langle \Phi(x), \Phi(y) \rangle$.

La dimensión del espacio de características necesaria para poder separar el conjunto de datos puede ser arbitrariamente grande. Pero al aumentar la dimensión de \mathfrak{I} también se incrementa el coste computacional de cualquier algoritmo que calcule el producto escalar operando directamente con las componentes de $\Phi(x)$. Por ejemplo, supongamos que queremos transformar imágenes de 16×16 puntos al espacio de monomios de orden 5 (es decir, el espacio de todos los productos con cinco factores, repetidos o no) de los 256 puntos de la imagen. La dimensión de este espacio sería:

$$\binom{260}{5} \approx 10^{10}$$

intratable a nivel computacional (véanse [Müller et al. 2001], página 184 o [Burges 1998], página 158).

Afortunadamente, para ciertos espacios de características y ciertas transformaciones existe una forma muy efectiva de calcular el producto escalar usando las denominadas *funciones núcleo*. Una función núcleo (o, simplemente, núcleo) es una función $K : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ tal que $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$, donde Φ es una transformación de \mathbb{X} en un cierto espacio de Hilbert \mathfrak{I} . Es decir, el producto escalar se puede calcular usando la función núcleo, quedando implícita la transformación del espacio de entrada en el espacio de características. Por ejemplo, supongamos que definimos la siguiente transformación Φ de \mathbb{R}^2 en el espacio de características \mathfrak{R}^3 : $\Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$.

Entonces, el producto escalar $\langle \Phi(x), \Phi(y) \rangle$ se puede reformular como

$$\langle \Phi(x), \Phi(y) \rangle = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)^T = ((x_1, x_2) \cdot (y_1, y_2)^T)^2 = \langle x, y \rangle^2.$$

Por tanto, la función núcleo $K(x, y) = \langle x, y \rangle^2$ permite calcular el producto escalar $\langle \Phi(x), \Phi(y) \rangle$ en el espacio de características sin necesidad de utilizar (ni siquiera conocer)

la transformación Φ . La caracterización de las funciones núcleo se explica en las subsecciones siguientes.

Supongamos que tenemos definido un producto escalar por medio de una función núcleo en su correspondiente espacio de características. Adaptando los mismos razonamientos de la SVM lineal con margen máximo, tenemos el siguiente resultado:

Teorema (SVM con margen máximo en el espacio de características). Supongamos que el conjunto de datos es linealmente separable en el espacio de características definido implícitamente por una función núcleo $K(x,y)$. Sea α^* una solución del problema dual

$$\begin{aligned} \text{Maximizar} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ \text{sujeto a:} \quad & \sum_{i=1}^N y_i \alpha_i = 0 \\ & \alpha_i \geq 0 \quad 1 \leq i \leq N. \end{aligned}$$

Entonces, el vector

$$\omega^* = \sum_{i=1}^N y_i \alpha_i^* \Phi(x_i)$$

es el vector ortogonal al hiperplano con margen (geométrico) máximo en el espacio de características. La SVM con margen máximo en el espacio de características es

$$h(x) = \langle \omega^*, \Phi(x) \rangle + b^* = \sum_{i=1}^N y_i \alpha_i^* K(x_i, x) + b^*$$

donde

$$b^* = -\frac{1}{2} \left(\max_{y_j=-1} \{ \langle \omega^*, \Phi(x_j) \rangle \} + \min_{y_j=+1} \{ \langle \omega^*, \Phi(x_j) \rangle \} \right)$$

y su clasificador asociado es $f(x) = \text{signo}(h(x))$.

Observemos que todo lo comentado para la SVM lineal de margen máximo acerca de los vectores soporte sigue siendo válido ahora. En este caso, además, el número de vectores soporte afecta a la eficiencia en el cálculo de la solución resultante.

Como vemos, uno de los *parámetros* del modelo anterior es la función núcleo $K(x,y)$. A continuación enumeramos las funciones núcleo más usuales. En la Sección 14.4 se verán las implicaciones prácticas de escoger un núcleo u otro.

Algunas funciones núcleo

Las funciones núcleo de propósito general más comúnmente utilizadas en \mathbb{R}^D son:

Polinómica	$(\langle x, y \rangle + c)^d$	$c \in \mathbb{R}, d \in \mathbb{N}$
Gaussiana	$\exp\left(\frac{-\ x - y\ ^2}{\gamma}\right)$	$\gamma > 0$
Sigmoidal	$\tanh(s \langle x, y \rangle + r)$	$s, r \in \mathbb{R}$
Multicuadrática inversa	$\frac{1}{\sqrt{\ x - y\ ^2 + c^2}}$	$c \geq 0$

Una de las grandes ventajas de las funciones núcleo es que su aplicación no está limitada a ejemplos de tipo vectorial sino que son aplicables a prácticamente cualquier tipo de representación. Dado que los datos de los problemas reales presentan habitualmente estructuras discretas de tipo secuencia (*string*), árbol o grafo, es posible y conveniente diseñar funciones núcleo específicas para estos casos concretos con el fin de capturar mejor las propiedades del problema (véase la Sección 14.4 de este capítulo). Ésta es, actualmente, una de las áreas activas de investigación en los paradigmas de aprendizaje basados en núcleos. Para más información sobre el diseño de núcleos se puede consultar [Cristianini & Shawe-Taylor 2000; Schölkopf & Smola 2002].

Ejemplo. En la Figura 14.4 podemos ver un ejemplo de SVM con margen máximo en el espacio de características inducido por el núcleo gaussiano⁴¹. Cada clase está representada por un color diferente. Como se puede observar, el conjunto no es linealmente separable en el espacio de entrada, pero sí en el espacio de características.

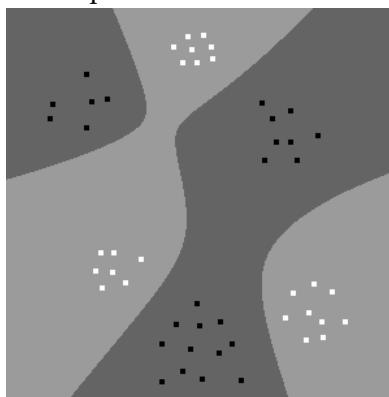


Figura 14.4. SVM con margen máximo en el espacio de características inducido por el núcleo gaussiano.

Caracterización de las funciones núcleo

Este apartado puede resultar algo técnico. El motivo de incluirlo aquí es el de hacer más autocontenido el texto, pero el lector puede omitirlo si lo prefiere sin que eso impida entender el resto del capítulo.

En espacios finitos, las funciones núcleo quedan caracterizadas por los valores propios de la matriz de productos escalares:

Teorema. Sea $\Delta = \{x_1, x_2, \dots, x_N\}$ un conjunto finito. Una función $K : \Delta \times \Delta \rightarrow \mathbb{R}$ simétrica es una función núcleo en Δ si y sólo si la matriz

$$\mathbf{K} = (K(x_i, x_j))_{i,j=1}^N$$

es semidefinida positiva (es decir, tiene valores propios no negativos).

El resultado en el que se basan las SVM con respecto a las funciones núcleo es debido a Mercer, desarrollado dentro del área del análisis funcional. El siguiente teorema es una versión simplificada del resultado que se puede encontrar en [Mercer 1909].

⁴¹ Esta figura y siguientes se han obtenido con el *software* de demostración 2D *on-line* de LIBSVM, disponible en la página web: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Teorema de Mercer. Sea Δ un subconjunto compacto de \mathbb{R}^D , y $K: \Delta \times \Delta \rightarrow \mathbb{R}$ una función continua y simétrica tal que

$$\forall f \in L_2(\Delta) \quad \int_{\Delta \times \Delta} K(x, y) f(x) f(y) dx dy \geq 0$$

Entonces K se puede expandir en una serie uniformemente convergente

$$K(x, y) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(y) \quad \lambda_j > 0 \quad \phi_j \in L_2(\Delta)$$

Como consecuencia del teorema de Mercer, dada una función K que cumpla las condiciones del teorema de Mercer, existe un espacio de Hilbert \mathfrak{I} y una transformación $\Phi: \mathbb{R}^D \rightarrow \mathfrak{I}$ tal que $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$. La transformación del espacio de entrada en el espacio de características es $\Phi(x) = (\phi_1(x), \phi_2(x), \dots)$, y el producto escalar K en \mathfrak{I} el expresado en el teorema.

Lo interesante de usar funciones núcleo con SVM es que el producto escalar se puede calcular implícitamente, sin conocer de manera explícita \mathfrak{I} ni la transformación Φ y evitando el coste computacional derivado de la posible alta dimensionalidad de \mathfrak{I} .

14.2.3 SVM con margen blando (*soft margin SVM*)

Desgraciadamente, no siempre es posible encontrar una transformación de los datos que permita separarlos linealmente, bien sea en el espacio de entrada o en el espacio de características inducido por alguna función núcleo (véase la Figura 14.3). A veces, incluso, no es conveniente Figura 14.5: la presencia de ruido (errores de medición, *outliers*, ejemplos mal etiquetados, etc.) puede forzar a que la SVM con margen máximo se fije demasiado en los ejemplos del conjunto de datos, provocando soluciones sobreajustadas a los ejemplos y que generalizan mal. La Figura 14.5(a) muestra un caso donde los ejemplos son linealmente separables, pero en el que el hiperplano de margen máximo (recta punteada) parece peor, en términos de generalización, que el hiperplano alternativo (recta continua) que clasifica mal uno de los puntos blancos. Las SVM con margen blando son modelos más robustos que las SVM con margen máximo, capaces de tratar conjuntos de datos no linealmente separables y con ruido.

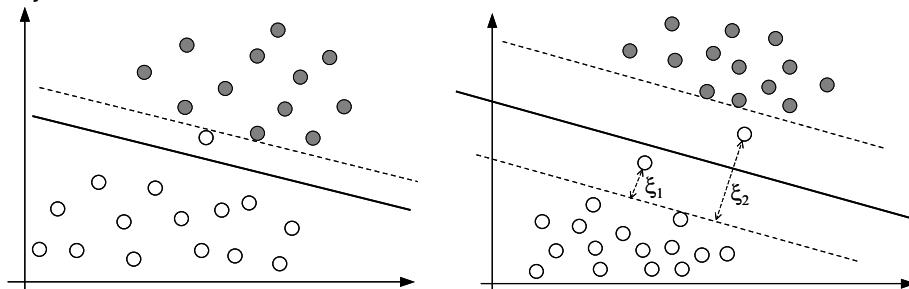


Figura 14.5. (a) Conjunto linealmente separable pero no deseable. (b) Variables de holgura.

La idea principal de las SVM con margen blando es la introducción de variables de holgura (del inglés *slack*, que se suelen denotar con la letra griega ξ), que permiten que las restricciones no se cumplan de manera estricta. El vector de variables de holgura tiene la misma dimensión que el conjunto de datos. Cada una de las componentes indica la máxima

diferencia posible entre 1 (el teórico margen funcional mínimo) y el margen real de cada ejemplo. El mínimo valor que puede tener una variable de holgura es 0. Como contrapartida, se introduce en la función a optimizar un término de regularización que depende de las variables de holgura, que establece un compromiso entre el margen y la magnitud de las mismas. Este término de regularización incluye una constante C , que determina la holgura del margen blando. Valores grandes de C obligan a tener pocas variables de holgura diferentes de 0 (o con valores grandes) en la solución final. Esta constante habrá que fijarla a priori y es, por tanto, un parámetro del aprendizaje.

SVM con margen blando y norma 1 de las variables de holgura

La SVM con margen blando más habitual es la que introduce la norma 1 de las variables de holgura, cuya formulación original es, en su versión lineal:

$$\begin{aligned} \text{Minimizar} \quad & \frac{1}{2} \langle \omega, \omega \rangle + C \sum_{i=1}^N \xi_i \\ \text{sujeto a:} \quad & y_i (\langle \omega, x_i \rangle + b) \geq 1 - \xi_i \quad 1 \leq i \leq N \\ & \xi_i \geq 0 \quad 1 \leq i \leq N \end{aligned}$$

donde C es una constante estrictamente positiva. Aplicando a este problema la teoría de optimización con restricciones lineales de forma similar a los problemas descritos en las secciones anteriores (las variables originales en este caso son ω, b y el vector de variables de holgura ξ), y usando funciones núcleo para permitir clasificadores no lineales, se obtiene el siguiente resultado:

Teorema (SVM con margen blando y norma 1 de las variables de holgura). Supongamos que tenemos un conjunto de datos y un espacio de características definido implícitamente por una función núcleo $K(x, y)$. Sea α^* una solución del problema dual

$$\begin{aligned} \text{Maximizar} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ \text{sujeto a:} \quad & \sum_{i=1}^N y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \quad 1 \leq i \leq N \end{aligned}$$

La SVM con margen blando en el espacio de características y norma 1 de las variables de holgura es

$$h(x) = \sum_{i=1}^N y_i \alpha_i^* K(x_i, x) + b^*$$

donde b^* es tal que $y_i \cdot h(x_i) = 1$ para todo i cumpliendo $0 < \alpha_i^* < C$. Su clasificador asociado es $f(x) = \text{signo}(h(x))$.

Curiosamente, la función a maximizar es exactamente la misma que en el caso de margen máximo. La única diferencia reside en la restricción $0 \leq \alpha_i \leq C$. Aunque hay versiones de SVM con margen blando donde esto no ocurre, este hecho nos permite entender mejor el comportamiento de los vectores soporte: la condición de Karush-Kuhn-Tucker implica que una variable de holgura sólo puede ser diferente de cero si $\alpha_i = C$. Por tanto, los ejemplos para los que $\alpha_i < C$ tienen el mismo comportamiento que en la SVM con margen máximo.

Dicho de otro modo, la SVM con margen máximo se puede obtener con $C = \infty$. Los ejemplos asociados a multiplicadores de Lagrange (véase el anexo de este capítulo) con valores altos pueden ser ejemplos con ruido. En las SVM de margen blando los vectores soporte se dividen en dos clases: los vectores soporte “normales” ($\alpha_i < C$ y margen funcional 1) y los vectores soporte “acotados” o *bounded* ($\alpha_i = C$ y margen funcional menor que 1). Algunos vectores soporte acotados pueden incluso estar mal clasificados por la SVM con margen blando, como se ve en la Figura 14.5.

SVM con margen blando y norma 2 de las variables de holgura

Otra versión de SVM con margen blando es la que introduce la norma 2 de las variables de holgura, cuya formulación original, en su versión lineal, es:

$$\begin{aligned} \text{Minimizar} \quad & \frac{1}{2} \langle \omega, \omega \rangle + C \sum_{i=1}^N \xi_i^2 \\ \text{sujeto a:} \quad & y_i (\langle \omega, x_i \rangle + b) \geq 1 - \xi_i \quad 1 \leq i \leq N \end{aligned}$$

donde C es una constante estrictamente positiva. Aplicando la teoría de optimización con restricciones a este problema, se puede obtener la versión dual de este problema de optimización (ver por ejemplo [Cristianini & Shawe-Taylor 2000]):

$$\begin{aligned} \text{Maximizar} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \left(K(x_i, x_j) + \frac{1}{C} \delta_{ij} \right) \quad \delta_{ij} = \begin{cases} 1 & \text{si } i=j \\ 0 & \text{si no} \end{cases} \\ \text{sujeto a:} \quad & \sum_{i=1}^N y_i \alpha_i = 0, \quad \alpha_i \geq 0 \quad 1 \leq i \leq N \end{aligned}$$

donde, como puede verse, el parámetro C juega un papel distinto al problema de margen blando con norma 1.

Ejemplo. En la Figura 14.6 podemos ver dos ejemplos de SVM con margen blando (norma 1 de las variables de holgura) en el espacio de características inducido por el núcleo gaussiano para el mismo conjunto de datos de la Figura 14.4.

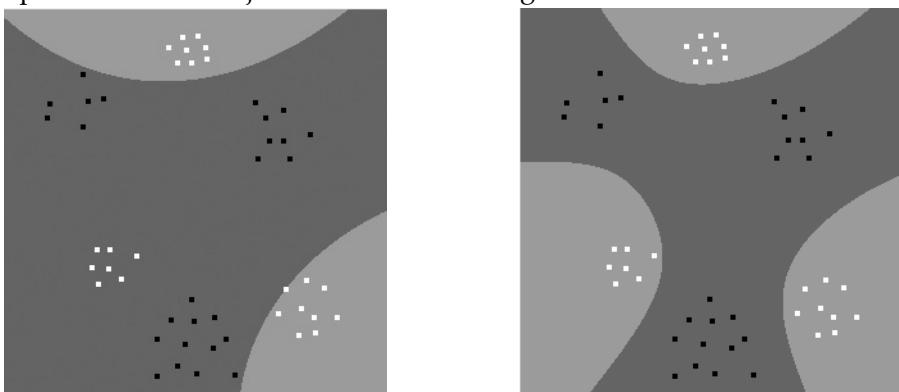


Figura 14.6. Dos ejemplos de SVM con margen blando usando el núcleo gaussiano. (a) $C=100$. (b) $C=1000$.

Cada clase está representada con un tono de gris diferente. Como se puede observar, con valores pequeños de C se permiten errores en el conjunto de aprendizaje (Figura 14.6a), mientras que valores más altos de C hacen que las clases queden completamente separadas (Figura 14.6b). Nótese, sin embargo, que la partición del espacio en clases de la Figura 14.6b es diferente a la obtenida con la SVM de margen máximo (Figura 14.4), a pesar de que

estamos en el mismo espacio de características. La decisión de qué modelo es mejor dependerá de su comportamiento en los puntos no vistos durante el aprendizaje.

Otros modelos de SVM con margen blando

La formulación original de SVM con margen blando corresponde a la de norma 1 y es debida a [Cortes & Vapnik 1995]. Recientemente han aparecido otras variaciones y extensiones con ideas similares a las explicadas hasta ahora. La variante que funciona mejor en la práctica depende del problema y puede estar influenciada por el tipo de ruido que haya en el conjunto de datos de aprendizaje. La explicación de las mismas va más allá de los objetivos de este capítulo.

14.3 Justificación teórica

A pesar de la aparente simplicidad de las ideas subyacentes a las SVM, los resultados teóricos que soportan la idea de las SVM y la maximización del margen son bastante complejos. Recordemos que el principal problema que se quiere resolver en un problema de aprendizaje supervisado es, a grandes trazos, el de aproximar la función objetivo de manera que se obtenga el menor error posible en *todo* el espacio de entrada (generalización). Para ello sólo disponemos del valor de la función objetivo en un conjunto finito de puntos. La elección de la familia de funciones con la que queremos aproximar la función objetivo es uno de los puntos clave que diferencia las diversas familias de métodos de aprendizaje.

Uno de los resultados fundamentales de la Teoría del Aprendizaje Estadístico (*Statistical Learning Theory*) [Vapnik 1998] acota el error de generalización por el error en el conjunto de datos más un término que depende de la complejidad de la familia de funciones con la que queremos aproximar la función objetivo. La complejidad de una familia de funciones está relacionada con la capacidad de aproximar un conjunto aleatorio de puntos. Otro resultado importante acota la complejidad de la familia de los hiperplanos por el margen geométrico invertido. La heurística resultante de combinar los dos resultados anteriormente comentados intenta maximizar el margen para acotar la complejidad de la familia de funciones, que a su vez acota el error de generalización. Para encontrar más información sobre las bases teóricas de las SVM se puede consultar [Burges 1998; Vapnik 1999; Müller et al. 2001] o [Vapnik 1995; Vapnik 1998].

14.4 Aplicaciones de las máquinas de vectores soporte

Como ya hemos dicho en la introducción, las SVM se han aplicado con éxito en los últimos años a numerosos problemas reales pertenecientes a áreas como la recuperación de información, reconocimiento y clasificación de imágenes, análisis de biosecuencias, reconocimiento de escritura, etc. Aparte de la solidez teórica de los modelos, el éxito empírico ha sido tal que algunos autores sugieren que las SVM pueden desplazar a las redes neuronales en una gran variedad de campos en un futuro próximo [Schölkopf & Smola 2002]. En esta sección aplicaremos, en primer lugar, el método de SVM a un problema artificial: la doble espiral. A continuación revisaremos dos enfoques de aplicación de SVM a un problema real de Categorización de Textos. Para encontrar más información

acerca de aplicaciones de las SVM se puede consultar [Cristianini & Shawe-Taylor 2000; Schölkopf & Smola 2002]⁴².

La aplicación de SVM como método de aprendizaje a un problema concreto requiere resolver dos cuestiones básicas de diseño. En primer lugar, el tipo de función núcleo que va a usar el método para explorar los datos. Existen las opciones estándar, como núcleos polinómicos o gaussianos, pero, tal y como veremos en el dominio de la categorización de textos, si el tipo de datos de la aplicación responde a estructuras de datos discretas, tales como secuencias, árboles o grafos, será posible diseñar funciones núcleo que expresen más eficazmente la similitud entre datos. En general, diseñar una función núcleo adecuada para un problema tiene que ver con: elegir una medida de *similaridad* entre ejemplos; elegir una *representación lineal* de los ejemplos y, por tanto, elegir el espacio de funciones de aprendizaje; introducir conocimiento a priori sobre la correlación de las observaciones; e introducir una distribución a priori sobre el conjunto de funciones de aprendizaje [Schölkopf & Smola 2002].

Una vez fijada la representación, la segunda cuestión a resolver es la dureza del margen que utilizará la SVM. Para ello, se deberá escoger entre el criterio de margen duro o una versión blanda controlada por el parámetro C . En ambos pasos, existirán parámetros del sistema que deberán optimizarse con los mismos datos, bien por validación cruzada (*cross-validation*) u otra metodología experimental de selección del mejor modelo.

14.4.1 La doble espiral

El problema de la doble espiral es un problema artificial en el que se trata de aprender a distinguir las dos áreas que definen dos espirales en un plano bidimensional, colocadas como muestra la Figura 14.7. Este problema, como vimos en el Capítulo 6, es bastante complejo e inabordable para muchas técnicas clásicas. El conjunto de aprendizaje consiste en tres vueltas de puntos de la espiral positiva (en tono oscuro y representado por cruces), y tres vueltas de puntos de la espiral negativa (en tono claro y representado por rombos). En total hay 194 puntos de aprendizaje contenidos dentro del cuadrado definido por los extremos $(-6,5, -6,5)$ y $(6,5, 6,5)$, tal y como muestra la figura.

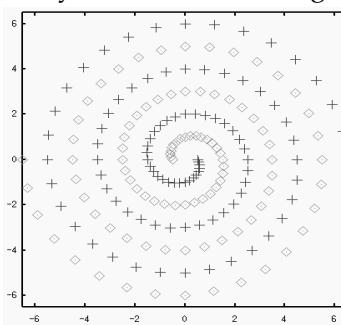


Figura 14.7. Conjunto de puntos de aprendizaje de la doble espiral.

En este problema también se incluye un conjunto de test de 194 puntos (definiendo dos espirales ligeramente desplazadas con respecto al conjunto de aprendizaje) que vamos a

⁴² También se puede consultar la página web de Isabelle Guyon: <http://www.clopinet.com/isabelle>.

usar para estimar el grado de acierto de las distintas variantes de SVM estudiadas. Vamos a usar el paquete de libre distribución $\text{SVM}^{\text{light}}$ para aprender un modelo que distinga entre las dos espirales⁴³. La entrada del problema consiste en dos dimensiones reales, que nos indican la posición de un punto en el espacio bidimensional, y el objetivo es predecir un valor real cuyo signo nos indique si el punto pertenece a la espiral oscura (positiva) o clara (negativa). Utilizaremos tres tipos de núcleos estándar: gaussianos, polinómicos y sigmoidales. Para cada uno de ellos deberemos encontrar la mejor configuración de parámetros del núcleo, junto con la mejor dureza del margen, controlada por el parámetro C .

Para los núcleos gaussianos, el único parámetro a optimizar es la anchura de la campana de Gauss (gamma). Dando un valor de margen duro ($C=1.000$) y probando varios valores de anchura, fácilmente se obtienen modelos que aproximan correctamente las dos espirales. La Figura 14.8 muestra el comportamiento de cuatro de estos modelos en el espacio de entrada entero, para cuatro valores del parámetro gamma ($0,01, 0,1, 1$ y 10) en los que paulatinamente se va cerrando la campana gaussiana. Los dos últimos modelos aprenden perfectamente el conjunto de aprendizaje y obtienen un grado de acierto estimado del 100 por cien, ya que definen perfectamente las espirales, a excepción de los extremos, dónde no hay puntos de aprendizaje.



Figura 14.8. SVM con núcleos gaussianos, con $\text{gamma} = \{ 0,01, 0,1, 1, 10 \}$ y $C=1.000$.

Para estos últimos modelos, la Figura 14.9 ilustra las funciones con que se aproximan las dos espirales. En el gráfico tridimensional, los dos ejes en el plano horizontal corresponden a las dimensiones de entrada, mientras que el eje vertical corresponde a la dimensión de salida. Con respecto a esta salida, los puntos negativos aparecen en tono claro, mientras que los positivos aparecen en tono oscuro. Los gráficos muestran claramente cómo las dos espirales se aproximan como composiciones de campanas de Gauss centradas en ciertos puntos de aprendizaje, correspondientes a los vectores soporte.

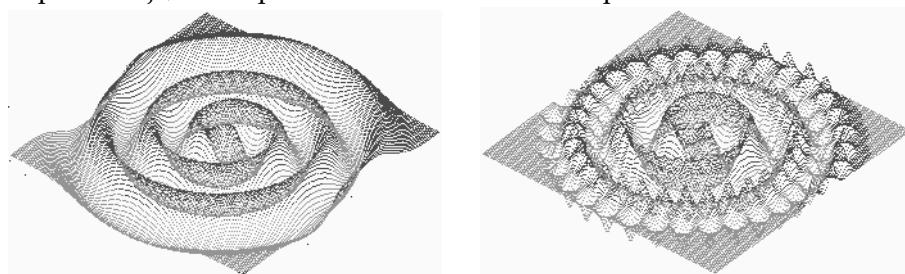


Figura 14.9. SVM con núcleos gaussianos, con $\text{gamma} = \{ 1, 10 \}$ y $C=1.000$.

⁴³ El software $\text{SVM}^{\text{light}}$, desarrollado por Joachims, se encuentra disponible en: svmlight.joachims.org. Véase también la Sección 14.6 para más información sobre este y otros paquetes de software.

También se aprecia el efecto del parámetro de anchura: en el primer modelo ($gamma=1$) las campanas son suficientemente anchas como para sobreponerse y definir las espirales con suavidad, mientras que en el segundo modelo de la Figura 14.9 ($gamma=10$) las campanas son más estrechas, y se ajustan mucho más a los vectores soporte, que quedan claramente delimitados en el gráfico.

Después de experimentar con los otros tipos de núcleos, polinómicos y sigmoidales, hemos obtenido resultados claramente insatisfactorios. De hecho es un resultado conocido la dificultad de aproximar las dos espirales como composición de funciones sigmoidales. Ello indica que las correspondencias que inducen estos núcleos no trasladan el problema a un espacio linealmente separable, y por tanto SVM fracasa. Debido a ello, en este caso será necesario trabajar con la versión de margen blando, controlando el parámetro C para encontrar el mejor compromiso entre ajuste a los datos y tolerancia al error. Para núcleos polinómicos, el parámetro a optimizar es el grado (d) del núcleo. Está claro que el problema de la doble espiral no es ni lineal ni cuadrático, por lo que estos dos casos dan resultados muy malos. La Figura 14.10 muestra el comportamiento para grados 4, 5 y 7, y distintos valores de C . El modelo con mejor grado de acierto estimado es el de grado 5, con 67,1 por ciento. Como se aprecia en el gráfico, las funciones aprendidas no definen la doble espiral, sino una relativa aproximación a ellas.



Figura 14.10. SVM polinómicas, con $[d=4, C=0,0001]$, $[d=5, C=0,001]$ y $[d=7, C=0,00001]$.

Asimismo, el uso de un núcleo sigmoidal da resultados insatisfactorios en este problema. Aquí, los parámetros a optimizar son la pendiente de la sigmoidal (s) y su desplazamiento (r), así como la dureza del margen C . Probando distintos valores, se obtienen aproximaciones que tienden a imitar una doble espiral, pero cuyo grado de acierto estimado es poco más del 50 por ciento. La Figura 14.11 muestra el comportamiento de tres modelos con núcleo sigmoidal, para $[s=1, r=18]$, $[s=2, r=18]$ y $[s=5, r=23]$, con $C=0,1$, siendo $[s=1, r=18]$ el mejor de ellos, con grado de acierto estimado del 59,8 por ciento.



Figura 14.11. SVM sigmoidales, con $C=0,1$, y $[s=1, r=18]$, $[s=2, r=18]$ y $[s=5, r=23]$.

14.4.2 Categorización de textos

El problema de la categorización o clasificación de textos (TC, del inglés *Text Categorization*) consiste, como su nombre indica, en etiquetar un texto o documento con una o varias categorías temáticas predefinidas. Este problema tiene aplicación en escenarios diversos del

área de la Recuperación de Información (IR, del inglés *Information Retrieval*), tales como la organización automática de documentos (en agencias de noticias, sitios web, cuentas de *e-mail*), filtrado de documentos (por ejemplo, filtrado de mensajes no deseados o *spam*), etc. La necesidad de tener herramientas automáticas para la gestión de este tipo de dominios se ha visto muy motivada en las últimas décadas, debido al creciente número de documentos accesibles digitalmente, por ejemplo, a través de la web. Por otro lado, los avances recientes en técnicas de aprendizaje automático han permitido elaborar sistemas para TC cada vez más efectivos, convirtiéndose, desde la perspectiva del aprendizaje, en un problema estándar para la evaluación de los distintos paradigmas y algoritmos⁴⁴. El método de las SVM ha demostrado, hoy por hoy, superioridad en este problema respecto a otros métodos, obteniendo los mejores resultados hasta la fecha en los conjuntos estándar de evaluación.

Definición formal. Sea X el dominio de todos los documentos, e Y un conjunto de categorías predefinidas. El objetivo de TC es aprender una función h , tal que dado un documento $x \in X$, nos devuelva el conjunto de categorías para tal documento, $h(x) \subseteq Y$. Para ello, dispondremos de un conjunto de aprendizaje $S=\{(x,y)\}_{i=1..m}$, formado por ejemplos (x,y) , donde x es un documento de X e y es un conjunto de categorías en Y . Para completar la experimentación, también dispondremos de un conjunto de validación o *test*, que se utilizará para evaluar la corrección de la función aprendida.

Metodología. Para modelar el problema de TC con las SVM de clasificación binaria, es necesario *binarizar* el problema. El método común es considerar cada posible categoría de Y como un problema binario por separado. Así, para cada categoría aprenderemos una SVM que decidirá si un documento pertenece o no a la categoría asociada. Para categorizar un documento aplicaremos cada SVM al documento, y devolveremos como resultado las categorías asociadas a las SVM que han clasificado el documento como positivo.

Medidas de evaluación. Para evaluar un sistema de TC se utilizan medidas de precisión y alcance (*precision and recall*), que son, como veremos en el Capítulo 17, medidas estándar del área de IR para tareas de reconocimiento o recuperación. La precisión mide la probabilidad de que si un sistema clasifica un documento en una cierta categoría, el documento realmente pertenezca a la categoría. Por otro lado, el alcance mide la probabilidad de que si un documento pertenece a cierta categoría, el sistema lo asigne a la categoría. Así, la precisión se puede ver como una medida de la corrección del sistema, mientras que el alcance da una medida de cobertura o completitud.

Para calcular estas medidas en un conjunto de *test*, se considera el problema de forma binarizada: cada documento con cada categoría del conjunto Y forma una decisión binaria. En este contexto, la siguiente tabla de contingencia resume el comportamiento de un sistema según la casuística de aciertos y errores:

	predicción positiva	predicción negativa
clase positiva	A	B
clase negativa	C	D

⁴⁴ El Capítulo 21 se dedica más extensamente a la minería de texto, la recuperación de la información y a la minería web. En este capítulo sólo se utiliza un ejemplo de categorización de textos para mostrar la flexibilidad para adaptar los núcleos a problemas que no son solamente del estilo atributo-valor.

En la tabla, cada celda representa el número de decisiones según el tipo de acierto o error. Así, A+D son los aciertos del sistema y B+D son los errores, y la suma de las cuatro celdas equivale al número total de decisiones binarias. Los valores de la tabla de contingencia permiten estimar las medidas de precisión y alcance según las siguientes expresiones:

$$\text{precisión} = \frac{A}{A+C} \quad \text{alcance} = \frac{A}{A+B}$$

Describir el comportamiento de un clasificador de textos con dos medidas no es práctico para comparar sistemas. Para ello, es común utilizar la medida F_β , que se define como:

$$F_\beta = \frac{(1+\beta^2) \text{precisión} \cdot \text{alcance}}{\beta^2 \cdot \text{precisión} + \text{alcance}}$$

que, para $\beta=1$, es la media armónica de la precisión y el alcance. En la medida, β es un parámetro que controla la importancia relativa entre las dos medidas. Es común usar el valor 1, que da igual importancia a las dos medidas.

Es importante notar que la categorización de un documento con una SVM se realiza teniendo en cuenta un umbral de decisión, que por defecto tiene valor 0 y por tanto hace que la categorización equivalga al signo de la predicción⁴⁵. Variando este umbral, podemos alterar el comportamiento de un clasificador en términos de precisión y *alcance*. Si aumentamos el umbral, requeriremos una magnitud de predicción (o confianza) mayor para clasificar un documento como positivo. Por tanto, el clasificador será más seguro y aumentará la precisión y bajará el *alcance*. Si, contrariamente, disminuimos el umbral, el clasificador será más permisivo, hecho que hará subir el *alcance* y bajar la precisión. Así, es posible ajustar el umbral de modo que precisión y *alcance* tengan el mismo valor. Este valor se conoce como *Break-even Point*, y se usa como medida combinada que resume el comportamiento de un sistema, alternativamente a la medida F_β . En la Sección 17.2.2 sobre evaluación también se introduce la evaluación basada en precisión y alcance, así como en la función de medida F_β .

En lo que queda de esta sección revisaremos un par de aproximaciones de la aplicación de SVM a TC, que varían la representación de un documento y su función núcleo.

14.4.3 Bag-of-words

En la bibliografía de IR, una técnica común para representar documentos es el modelo basado en vectores de palabras, conocido como *bag-of-words*⁴⁶. En este modelo, cada palabra de un diccionario prefijado corresponde a un atributo o dimensión del espacio de documentos. Con esto, un documento se representa dando valor a las dimensiones de las palabras que contiene. Existen varias aproximaciones para dar valor a los atributos: valor booleano indicando la presencia o ausencia de la palabra en el documento, el número de ocurrencias, o un valor real indicando el peso de la palabra en el documento. La distancia entre documentos se calcula mediante el producto escalar. En este modelo, por tanto, no se considera el orden que las palabras ocupan en el documento.

⁴⁵ Esto también es cierto para todos los métodos que devuelven un valor real indicativo de la confianza de la decisión.

⁴⁶ En el Capítulo 21 de minería de texto y web se desarrollan también los conceptos de *bag-of-words*, n-gramas, etc.

Para confeccionar el diccionario que forma el espacio de documentos, se consideran las palabras que forman una colección de documentos. Es habitual filtrar las palabras que no tienen contenido semántico (tales como determinantes, preposiciones, conjunciones, etc., llamadas *stop words*), y considerar lemas o raíces de palabras en lugar de sus formas (por ejemplo, “políticos” y “política” se representarían por su raíz “politic”). Un criterio estándar en IR para dar peso a cada atributo es considerar la medida TF-IDF, que se define como:

$$\phi_i(x) = tf_i \cdot \log(1/df_i)$$

donde tf_i es el número de ocurrencias de la palabra i en el documento x , y df_i es el ratio entre el número de documentos que contienen la palabra i -ésima y el número total de documentos de la colección.

Típicamente, una colección de documentos da lugar a decenas de miles de palabras o atributos. En este espacio el vector asociado a un documento es altamente disperso (*sparse*), con sólo unas decenas de atributos con valor distinto de cero. Dado que sólo es necesario calcular el producto escalar entre vectores documento, se pueden desarrollar representaciones muy eficientes con las que el producto escalar se calcula rápidamente.

Presentamos a continuación resultados del estudio de Joachims aplicando SVM para TC [Joachims 2002]. La experimentación se realizó sobre el conjunto de datos *Reuters-21578*, considerado estándar para el problema. Es una colección de documentos de la agencia de noticias Reuters, etiquetados según un esquema de categorías del ámbito económico. Muchos trabajos experimentales se concentran en las 90 categorías más frecuentes o incluso un subconjunto con sólo las diez más frecuentes, que cubren un total de 9.603 documentos de aprendizaje y 3.299 de *test*. En el trabajo de Joachims, el diccionario contemplaba 9.947 palabras, y los documentos tenían una media de 200 palabras aproximadamente. La Tabla 14.1 presenta los resultados de la experimentación en términos de *Break-even Point* para las diez categorías más frecuentes (filas superiores), y también la media de las 90 categorías (fila inferior). Se compara SVM contra cuatro métodos de aprendizaje estándar⁴⁷: *Naive Bayes*, *Rocchio*⁴⁸, árboles de decisión C4.5 y *k-Nearest Neighbour*. Los parámetros de cada método fueron optimizados adecuadamente. Para SVM, se muestran resultados para SVM lineales, con dos valores del parámetro de margen blando C , y un resultado usando un núcleo gaussiano. En casi todos los casos (menos para la categoría *interest*), los resultados de las SVM superan cualquiera de los otros métodos, evidenciando la superioridad de SVM sobre métodos convencionales en este dominio. Joachims argumenta que las SVM son particularmente eficaces porque con la estrategia de maximización del margen permite generalizar conceptos en espacios de gran dimensionalidad, mientras que las estrategias de los otros métodos fallan en estos casos. Los otros métodos no pueden tratar con espacios de dimensionalidad tan alta por motivos de generalización y coste computacional, y por tanto no son capaces de explorar el espacio con la misma eficacia. Para un estudio más detallado de la idoneidad de las SVM en el dominio de TC se puede consultar [Joachims 2002].

⁴⁷ Estos métodos se ven en otros capítulos de esta parte del libro.

⁴⁸ Rocchio es un algoritmo bastante popular en recuperación de información que utiliza las frecuencias de los términos y de los documentos [Rocchio 1971].

	Bayes	Rocchio	C4.5	k-NN	SVM lineal C=0,5	SVM gauss. gamma=0,01
<i>Earn</i>	96,0	96,1	96,1	97,8	98,0	98,2
<i>acq</i>	90,7	92,1	85,3	91,8	95,5	95,6
<i>money-fx</i>	59,6	67,6	69,4	75,4	78,8	78,5
<i>grain</i>	69,8	79,5	89,1	82,6	91,9	93,1
<i>trade</i>	81,2	81,5	75,5	85,8	89,4	89,4
<i>crude</i>	52,2	77,4	59,2	77,9	79,2	79,2
<i>interest</i>	57,6	72,5	49,1	76,7	75,6	74,8
<i>ship</i>	80,9	83,1	80,9	79,8	87,4	86,5
<i>wheat</i>	63,4	79,4	85,5	72,9	86,6	86,8
<i>corn</i>	45,2	62,2	87,7	71,4	87,5	87,8
90 categ.	72,3	79,9	79,4	82,6	86,7	87,5
						86,4

Tabla 14.1. Resultados de Break-even Point de distintos métodos en el corpus Reuters-21578 [Joachims 2002].

14.4.4 String kernel

En esta sección presentamos una técnica más sofisticada para tratar con datos de tipo *string* y, en particular, dominios textuales. La idea básica reside en comparar documentos, vistos como *strings*, analizando los *substrings* que contienen. Cuantos más *substrings* en común tengan dos documentos, más similares serán. La técnica permite que los *substrings* considerados no sean contiguos en los documentos, es decir, se permiten agujeros en la evaluación de *substrings* de un documento, que penalizarán proporcionalmente según su longitud. La técnica presentada, desarrollada por [Lodhi et al, 2002], permite evaluar eficientemente el producto escalar, o función núcleo, de dos documentos en el espacio de atributos formado por todos los *substrings* de los documentos. La complejidad computacional resultante es lineal respecto a la longitud de los documentos comparados y a la longitud de los *substrings* considerados. A continuación se describe la notación y la expresión del núcleo.

Sea Σ un alfabeto, y Σ^n el conjunto de todos los *strings* de longitud n . El conjunto de todos los *strings* finitos se expresa como:

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

La longitud de un *string* s se denota como $|s|$, y sus elementos como $s(1), \dots, s(|s|)$. La concatenación de dos *strings* s y t se denota como st . Dada una secuencia de índices $i = (i_1, \dots, i_{|u|})$, cumpliendo $1 \leq i_1 \leq \dots \leq i_{|u|} \leq |s|$, se define la subsecuencia u de s como $u = s(i) = s(i_1) \dots s(i_{|u|})$. La longitud $l(i)$ de una subsecuencia de s es $i_{|u|} - i_1 + 1$. Dado n , definimos el espacio de atributos como $F_n = \mathcal{R}^{\Sigma^n}$. Este espacio tiene una dimensión por cada *string* de Σ^n , y se indexa por los mismos *strings*. Podemos entonces definir la correspondencia de atributos ϕ de un *string* s hacia F_n definiendo la función que evalúa cada dimensión u de Σ^n :

$$\phi_u(s) = \sum_{iu=s[i]} \lambda^{l(i)}$$

En la función, el parámetro $0 \leq \lambda \leq 1$ es un factor de decalaje: cuanto mayor sea la longitud de la subsecuencia, menor será su contribución en el valor de $\phi_u(s)$ y, por tanto, en el cálculo de la similitud entre documentos. Estos atributos miden el número de ocurrencias de subsecuencias en el *string* s , ponderadas en función de sus longitudes. Por lo tanto, el

producto escalar para dos *strings* s y t computa la suma de todas las subsecuencias comunes en los *strings*, ponderadas según su frecuencia de ocurrencia y longitudes. La formulación del producto escalar, o función núcleo, es:

$$K_n(s, t) = \sum_{u \in \Sigma^n} \langle \phi_u(s) \cdot \phi_u(t) \rangle = \sum_{u \in \Sigma^n} \sum_{i: u=s[i]} \sum_{j: u=t[j]} \lambda^{l(i)} \lambda^{l(j)} = \sum_{u \in \Sigma^n} \sum_{i: u=s[i]} \sum_{j: u=t[j]} \lambda^{l(i)+l(j)}$$

Con esta formulación, el trabajo de [Lodhi et al. 2002] presenta un algoritmo de programación dinámica que calcula la función núcleo con una complejidad computacional de $n \cdot |s| \cdot |t|$. La Tabla 14.2 muestra los resultados experimentales para la categoría *earn* de la colección *Reuters-21578* obtenida por los mismos autores.

Núcleo	n	λ	precisión	alcance	F1
<i>palabras</i>	-	-	98,9	86,7	92,5
<i>n-gramas</i>	3	-	97,4	87,3	91,9
	5	-	99,2	90,3	94,4
	10	-	99,0	88,5	93,2
<i>substrings</i>	4	0,5	99,2	88,8	93,2
	5	0,01	99,2	90,5	94,6
	5	0,05	99,2	90,3	94,4
	5	0,1	99,2	90,3	94,4
	5	0,5	99,2	88,8	93,6
	7	0,5	99,2	90,0	94,0

Tabla 14.2 . Resultados para núcleos de substrings para la clase “earn” de Reuters-21578 [Lodhi et al. 2002].

Debido a limitaciones computacionales, en este trabajo los conjuntos de aprendizaje se consideraron más reducidos que en los de [Joachims 2002] (Tabla 14.1 anterior), de manera que los resultados no son directamente comparables. En la Tabla 2 se comparan tres tipos de núcleos: basado en palabras, correspondiente al modelo *bag-of-words*; basado en *strings* contiguos, conocidos como *n-gramas* en este tipo de representaciones⁴⁹; y basado en *substrings* no contiguos. Los resultados muestran que, en condiciones óptimas de parámetros, los núcleos basados en *substrings* (contiguos o no) superan el basado en palabras. Permitir la no contigüidad (*substrings*) se traduce en un ligero beneficio respecto a considerar sólo *substrings* contiguos (*n-gramas*), aunque el coste computacional es también mucho más elevado. Este compromiso entre eficiencia y eficacia deberá resolverse en base a las prioridades y restricciones de la aplicación final que estemos considerando.

En definitiva, el diseño de una función *kernel* específica, y la capacidad de las SVM de maximizar el margen a través de la función *kernel*, permiten elaborar sistemas más eficaces que los convencionales.

14.5 Extensiones y temas avanzados

Tal como avanzábamos en la introducción, el contenido del capítulo se ha circunscrito al estudio de los modelos de SVM para clasificación binaria y sus aplicaciones. Pero en realidad, la teoría actual en el marco de las SVM y la familia más general de aprendizaje basado en núcleos ha ido mucho más allá. Es un hecho indiscutible que ésta es un área de investigación especialmente activa, con contribuciones, resultados y aplicaciones nuevas

⁴⁹ Los *n-gramas* se tratan en el Capítulo 21.

apareciendo continuamente. El propósito de esta sección es enumerar algunas de las extensiones o temas avanzados para que el lector interesado en profundizar disponga de algunas referencias útiles.

El método de los vectores soporte también puede aplicarse para el caso de *regresión* manteniendo intactos los elementos básicos del algoritmo de maximización del margen: inducción de una función no lineal por medio de aprendizaje lineal en un espacio de características haciendo uso de funciones núcleo. Como en el caso de la clasificación el algoritmo de aprendizaje minimiza una función convexa y la solución es dispersa. Para ello se define una función de penalización (*loss function*) que ignora los errores a una distancia ε del valor verdadero⁵⁰ (a esta función se la suele denominar ε -insensible). La *regresión ε -insensible* se basa en encontrar una función no lineal de manera que todos los datos de aprendizaje estén a distancia menor o igual que ε (ε se toma como un parámetro del aprendizaje que mide la finura con que se va a tolerar el error en la muestra de aprendizaje). A pesar de que la función se ajuste bien a todos los puntos, la expresión de la función no lineal va a depender idealmente sólo de unos cuantos puntos o *vectores soporte*. Para ver detalles sobre el desarrollo se puede consultar [Cristianini & Shawe-Taylor 2000].

Una variante interesante de las SVM tanto para clasificación como para regresión son las denominadas ν -SVM, donde se sustituye el parámetro C de la optimización con margen blando por otro parámetro ν . Si el parámetro C establecía el compromiso entre dos medidas contrapuestas como minimizar el error en la muestra de aprendizaje y maximizar el margen, el parámetro ν pretende ser más intuitivo y controla el compromiso entre el número de *errores de margen* (puntos mal clasificados o que caen dentro del margen) y el número de vectores soporte. De esta manera se introduce una forma explícita de controlar la dispersión de la solución con respecto al número de vectores soporte. Véase [Schölkopf & Smola 2002] para más detalles.

Una gran variedad de métodos de aprendizaje, todos aquellos que admitan una expresión dual donde intervengan únicamente productos escalares entre vectores, pueden “kernelizarse” y estudiarse dentro de la familia de métodos de aprendizaje basados en funciones núcleo. Entre muchos otros, encontramos desarrollos y aplicaciones como por ejemplo: una versión no lineal del Análisis de Componentes Principales (PCA, véase la Sección 4.3.1) denominada Kernel-PCA y su aplicación en un entorno de selección de atributos relevantes [Shölkopf & Smola 2002]; una versión del discriminante de Fisher en el espacio de características (o Kernel-FD) que es competitivo con las SVM pero que tiene la ventaja de poder transformar fácilmente sus predicciones en probabilidades condicionales [Shölkopf & Smola 2002]; una variante del perceptrón (*Voted Perceptron*) usado dualmente con núcleos para resolver problemas de procesamiento del Lenguaje Natural [Collins & Duffy 2002]; etc.

La aplicación de las SVM a problemas de clasificación multiclase se suele plantear mediante los esquemas habituales de binarización (*one-vs-all*, *all-pairs*, ECOC, etc.), como vimos en la Sección 6.2.1 que convierten un problema multiclase en varios binarios, siendo, por tanto, independientes del método de aprendizaje utilizado (véase, por ejemplo,

⁵⁰ De la misma manera que se ignoran las diferencias entre predicciones de los ejemplos bien clasificados en el modelo de SVM para clasificación.

[Allwein et al. 2000]). Sin embargo, existen también variantes más elegantes de SVM donde una modificación de la función objetivo permite obtener simultáneamente el cálculo de un clasificador multiclas (véase [Schölkopf & Smola 2002]). Estas variantes han demostrado ser experimentalmente competitivas en términos de calidad con respecto a los esquemas de binarización. Sin embargo pueden ser computacionalmente más costosas y dar lugar a soluciones menos compactas. En definitiva, la elección de la mejor estrategia va a depender del problema y de los datos concretos que se manejen.

Tal como se han presentado en este capítulo, el modelo de SVM es un modelo de aprendizaje no incremental (*batch*). En algunos dominios reales es muy conveniente disponer de métodos incrementales (*on-line*) que puedan adaptarse a situaciones cambiantes de la función objetivo y a la llegada secuencial de ejemplos con un coste computacional aceptable. A pesar de que no ha sido un tema muy tratado en los inicios de las SVM, algunos trabajos recientes de investigación presentan estrategias incrementales efectivas para el aprendizaje de SVM tanto para clasificación como para regresión [Cauwenberghs & Poggio 2001].

Otra característica fijada a priori en este capítulo ha sido el trabajar con aprendizaje supervisado. El aprendizaje *no supervisado* (también conocido como *clustering*) o *agrupamiento*, consiste en encontrar agrupaciones “coherentes” y “útiles” de datos cuando las clases son desconocidas a priori. Con respecto a las SVM, encontramos un modelo general para aprendizaje no supervisado en [Ben-Hur et al. 2001]: se usa un núcleo gaussiano para encontrar en el espacio de características la mínima esfera que incluya todos los puntos. Cuando la esfera se vuelve a considerar en el espacio de entrada se separa en varios componentes englobando distintos *grupos* de puntos. La anchura del núcleo y la constante del margen blando son los parámetros que permiten controlar el *clustering*. Por otro lado, las denominadas *single-class* SVM son una extensión natural del modelo de SVM para clasificación en el caso no supervisado. En [Schölkopf & Smola 2002] se presenta su aplicación al problema no supervisado de *high-dimensional quantile estimation*, consistente en, dado un conjunto de datos S generado a partir de una distribución P , encontrar un subconjunto S' tal que la probabilidad de que un punto generado a partir de P caiga fuera de S' sea menor que un valor de probabilidad fijado a priori (simplificación del problema de *density estimation*, véase por ejemplo [Silverman 1986]).

Por otro lado, en ciertos problemas reales conseguir un número de ejemplos suficientemente grande para garantizar un buen aprendizaje supervisado es muy costoso (en tiempo y/o dinero), mientras que la obtención de grandes cantidades de ejemplos sin etiquetar es tarea fácil⁵¹ (por ejemplo, reorganización automática de documentos, filtrado de e-mail, etc.). En [Joachims 2002] se presenta un algoritmo de aprendizaje de SVM especializado para sacar partido de los ejemplos no etiquetados (denominado *transductive SVM learning*, por oposición a *inductive learning*) y se demuestran empíricamente sus ventajas en el dominio de la Clasificación de Documentos. En síntesis, la SVM aprende un separador que clasifique correctamente los ejemplos etiquetados tratando, a su vez, de maximizar la coherencia de la distribución de los ejemplos no etiquetados (por ejemplo, evitando en lo posible que queden muchos ejemplos no etiquetados cerca de la frontera de

⁵¹ En el Capítulo 18 se trata el uso generalizado de conjuntos de datos no etiquetados o parcialmente etiquetados para clasificación

decisión). En este proceso de aprendizaje, el algoritmo tiene que ir suponiendo incrementalmente las clases de los ejemplos no etiquetados para ir ajustando el hiperplano separador, por lo que el aprendizaje se vuelve bastante más ineficiente. Disponer de algoritmos de aprendizaje incremental sería pues muy importante para acelerar el aprendizaje en este caso.

Hay otros algoritmos de aprendizaje cuyo análisis teórico se basa en la maximización del margen. Por ejemplo, AdaBoost, que se verá en el Capítulo 18, es un algoritmo para construir una combinación lineal de clasificadores “simples” en una regla de clasificación con error de aprendizaje arbitrariamente pequeño [Schapire 2002]. Parte de sus buenas propiedades de generalización se basan en el análisis teórico del algoritmo que demuestra que es especialmente agresivo en la maximización del margen funcional sobre el conjunto de aprendizaje. Para encontrar más detalles sobre la relación de las SVM con AdaBoost y otras variantes se puede consultar [Smola et al. 2000; Rätsch 2001; Schapire 2002].

14.6 Paquetes *software* y recomendaciones de uso

A pesar de tratarse de un modelo de aprendizaje relativamente reciente y en continuo desarrollo, en la actualidad existen ya multitud de paquetes *software* de libre distribución para SVM y máquinas basadas en núcleos, incluyendo herramientas gráficas para demostraciones 2D. En el portal <http://www.kernel-machines.org> se puede acceder a toda esta información de forma muy sencilla. Por lo que a SVM se refiere, podemos encontrar *software* para múltiples modelos de regresión y clasificación, orientado a distintos tipos de usuario, para procesar eficientemente grandes conjuntos de datos, implementado usando lenguajes de programación como: C, C++, Java, Matlab, Fortran, etc., y funcionando sobre varias plataformas: SunOS, Solaris, Linux, Windows, Cygwin, etc.

Recomendamos LIBSVM, SVM^{light} o SVMTorch⁵² entre todos los paquetes de *software* de libre distribución para docencia e investigación. Todos ellos son robustos, fáciles de instalar y utilizar, además de estar frecuentemente actualizados. Soportan y permiten experimentar fácilmente con las funciones núcleo estándar y trabajar con miles de ejemplos y atributos, representando los ejemplos de manera dispersa (*sparse*). También soportan la mayoría de los modelos de clasificación y regresión (y sus extensiones) explicados en este capítulo y ofrecen el código abierto para futuras modificaciones y extensiones.

LIBSVM (desarrollado por C. Chang y C. Lin en la *National University of Taiwan*) es el paquete más recomendable para principiantes, ya que es muy completo y dispone de una guía y varios *scripts* de utilización para usuarios no expertos, además de una interfaz gráfica de demostración en 2D. Estas características se complementan con herramientas de ayuda en la selección de parámetros adecuados (*model selection*) y normalización de los atributos numéricos del conjunto de datos (*data scaling*) y de una lista de preguntas frecuentes (*FAQ*) completa y actualizada. LIBSVM dispone de versiones en C++ y Java y ofrece interfaces para varios lenguajes de programación. En cuanto al propio aprendizaje de SVM, LIBSVM es también muy completo y ofrece algoritmos para clasificación y regresión (incluyendo *v-SVM*) y *single-class* SVM. También se incluye una opción para

⁵² LIBSVM, SVM^{light} y SVMTorch están disponibles en las webs: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, <http://svmlight.joachims.org> y <http://www.idiap.ch/learning/SVMTorch.html>, respectivamente.

tratar automáticamente con problemas multiclas, de modo que el usuario no debe preocuparse de binarizar el problema (el esquema utilizado es un sencillo *one-vs-all*). De todas maneras, es conveniente saber que hay otros paquetes *software*, como M-SVM (disponible también en <http://www.kernel-machines.org>), para tratar específicamente los problemas de clasificación multiclas. Para finalizar, diremos que es muy destacable la eficiencia del algoritmo de optimización implementado por LIBSVM incluso trabajando con conjuntos de datos de miles de ejemplos con miles de atributos.

SVM^{light} (desarrollado por T. Joachims de la *Cornell University*) es probablemente el *software* más adecuado para un uso avanzado de las SVM, tanto por las funcionalidades que ofrece como por su robustez. La versión actual está implementada en C y ofrece versiones de instalación para muchas plataformas, e interfaces de conexión con Matlab y Java. El algoritmo de optimización implementado es muy completo e incluye varias heurísticas y técnicas de programación que permiten usar la herramienta de manera eficiente incluso con cientos de miles de ejemplos de aprendizaje. Las opciones principales de aprendizaje incluyen clasificación, regresión y *ranking* (donde el problema consiste en inferir, para cada ejemplo, una ordenación de las clases posibles). Para ayudar en la selección de parámetros ofrece, opcionalmente en el aprendizaje, el cálculo de estimadores del *error de generalización, precisión y alcance* (*precision and recall*). Destaca el cálculo, aproximado pero con un coste adicional mínimo, del estimador del *leave-one-out error*. Como se explica en la Sección 14.5, SVM^{light} ofrece la posibilidad de aprender simultáneamente a partir de ejemplos etiquetados y no etiquetados (mediante las denominadas *transductive SVM*). Otra opción interesante es la posibilidad aprender con un modelo de costes, donde se puede dar pesos distintos a los diferentes errores de clasificación. Finalmente, SVM^{light} soporta todos las funciones núcleo estándar y además permite al usuario definir y utilizar sus propias funciones núcleo, mediante la adición de archivos *kernel.h* (en LIBSVM por el contrario hay que tocar el código fuente directamente).

SVMTorch (desarrollado por R. Collobert y S. Bengio) es una implementación en C++ de los algoritmos de clasificación y regresión de SVM para grandes conjuntos de datos. Aunque mucho más limitada en cuanto a opciones, se puede considerar como alternativa a los dos paquetes anteriores. Paralelamente, SVMTorch forma parte de la librería genérica Torch de métodos de aprendizaje (desarrollada en el instituto de investigación IDIAP), por lo que se puede acompañar de otros algoritmos y herramientas no necesariamente ligadas a SVM. Éste sería también el caso de otros algoritmos para SVM integrados en plataformas generales de *software* de aprendizaje y minería de datos. Por ejemplo, en el entorno WEKA, uno puede beneficiarse de las interfaces gráficas para usar cómodamente los algoritmos para SVM conjuntamente con las herramientas de exploración/análisis de datos, selección de atributos, ajuste de parámetros, etc. De todas formas, para un uso intensivo del modelo SVM en problemas reales, grandes y complejos, recomendamos utilizar las herramientas específicas LIBSVM y SVM^{light} explicadas en los párrafos anteriores.

A pesar de que cada problema concreto es un mundo aparte, vamos a intentar sintetizar en este párrafo algunas de las recomendaciones de uso principales a la hora de enfrentarse a un problema concreto con SVM. Supongamos que ya tenemos los ejemplos representados en forma vectorial, es decir, en la forma tabular tradicional. Como en la mayoría de técnicas de aprendizaje, para evitar sesgos no deseados y mejorar la estabilidad numérica del aprendizaje, es conveniente normalizar los valores de los atributos numéricos

(por ejemplo, escalándolos a un intervalo [0,1] o normalizando a media=0 y variancia=1). Los atributos discretos es recomendable binarizarlos (cada par atributo-valor se convierte en un atributo binario, numerización “1 a n ”, vista en la Sección 4.4.2). A continuación, se debería fijar una función núcleo e intentar ajustar al mismo tiempo los parámetros propios del núcleo y del aprendizaje. Se recomienda empezar por un núcleo gaussiano (por ser más estable y tener pocos parámetros a optimizar) o bien por un núcleo lineal si sospechamos que el conjunto de datos es linealmente separable. Se recomienda hacer una exploración por intervalos de todas las combinaciones de parámetros, refinando la búsqueda en los intervalos donde mejor generalización se observa (la finura de la exploración dependerá fundamentalmente de los requerimientos computacionales y del tiempo disponible). Para alguna combinación de parámetros (por ejemplo, para valores del parámetro C muy grandes en problemas complejos con respecto al tipo de núcleo usado) es posible que el algoritmo de optimización no converja. Es conveniente, por tanto, predeterminar un tiempo máximo para cada experimento. La estimación del grado de acierto para cada parametrización se debería hacer, si es computacionalmente viable, por validación cruzada (*cross-validation*), como se ve en 6, o, alternativamente, utilizando los estimadores de error de generalización (proporcionados por $\text{SVM}^{\text{light}}$, por ejemplo). La selección de un buen modelo es fundamental para evitar el sobreajuste y conseguir buenos resultados de generalización. También se puede mirar el grado de compresión del modelo resultante (proporción del número de vectores soporte con respecto al número de ejemplos de aprendizaje), puesto que modelos dispersos (con pocos vectores soporte) tienen menos probabilidades de estar sobreajustados. Finalmente, habría que probar con otras funciones núcleo siguiendo el mismo procedimiento. Intentar seleccionar el núcleo más simple dentro de los que permiten generalizar adecuadamente sería el objetivo principal de la experimentación.

Joachims sugiere en [Joachims 2002] que las SVM son muy adecuadas para problemas del tipo *Clasificación de Documentos*, analizado en la Sección 1.4. En este tipo de problemas el número de dimensiones es muy elevado (miles de palabras = miles de atributos), cada ejemplo tiene una codificación muy dispersa (sólo unos pocos atributos son distintos de cero) y el concepto a aprender es denso (muchos de los atributos son realmente relevantes). Véase [Joachims 2002] para una descripción detallada del modelo estadístico que justifica la idoneidad de las SVM en la clasificación de documentos. En problemas donde se dispone de pocos ejemplos, las SVM presentan ventajas con respecto a otros métodos basados en maximización del margen, como AdaBoost (véase el Capítulo 18). Si además es posible disponer de una gran cantidad de ejemplos no etiquetados para complementar el conjunto inicial, entonces se abre la posibilidad de usar el modelo de SVM *transductiva* (comentada en la Sección 14.5). Desde el punto de vista práctico, el hecho de enfrentarnos a un problema con miles de ejemplos de aprendizaje y miles de atributos (donde incluso el número de dimensiones es superior a la de ejemplos) no ha de asustarnos a la hora de utilizar SVM, puesto que los algoritmos actuales permiten trabajar eficientemente en estos espacios de alta dimensionalidad y el sesgo de maximizar el margen, conjuntamente con un buen ajuste del modelo, permite evitar en lo posible el probable sobreajuste. Por último, en problemas donde tengamos la intuición que se puede sacar partido de la estructura no vectorial de los ejemplos para definir medidas de similitud que ayuden a discriminar entre

clases, es muy conveniente plantearse la programación de un núcleo propio y usarlo en el aprendizaje de SVM o de cualquier otro modelo basado en funciones núcleo.

En la preparación del material de este capítulo se han utilizado diversas fuentes que por su calidad, extensión y punto de vista merecen ser referenciadas. Entre toda la bibliografía básica sobre SVM, nos gustaría destacar: dos excelentes introducciones en formato de artículo [Burges 1998 ; Müller et al. 2001], el trabajo de Cristianini y Shawe-Taylor, en formato de libro de texto, muy completo y adecuado para una primera lectura sobre SVM [Cristianini & Shawe-Taylor 2000] y dos libros muy recientes que incluyen, aparte de una introducción general autocontenido, los últimos desarrollos sobre SVM y métodos de aprendizaje basados en núcleos [Schölkopf & Smola 2002; Herbrich 2002]. El primero de estos dos libros puede considerarse heredero del publicado en 1999 por los mismos autores [Schölkopf et al. 1999]. Por último, es de destacar que la Web creada por Schölkopf y Smola en abril del año 2000 (<http://www.kernel-machines.org>) es un portal muy completo y activo de acceso a las principales publicaciones, recursos, paquetes *software*, conferencias, etc., del mundo de los métodos de aprendizaje basados en núcleos y, en particular, de las SVM.

Anexo. Optimización con restricciones lineales

Las SVM necesitan, como hemos visto, resolver problemas de optimización con restricciones lineales. La manera habitual de resolver estos problemas es usando la teoría desarrollada en 1797 por Lagrange para resolver problemas de optimización con restricciones de igualdad, y extendida por Kuhn y Tucker en 1951 para restricciones en forma de desigualdad. En este anexo se resumen las ideas principales de la teoría de optimización con restricciones lineales necesarias para la resolución de los problemas asociados a las SVM.

Consideremos el *problema original o primal* de optimización con dominio $\Omega \subseteq \mathbb{R}^D$

$$\begin{aligned} \text{Minimizar} \quad & f(\omega) \quad \omega \in \Omega \\ \text{sujeto a:} \quad & g_i(\omega) \leq 0 \quad 1 \leq i \leq n \\ & h_i(\omega) = 0 \quad 1 \leq i \leq m \end{aligned}$$

Dada la función de Lagrange generalizada

$$L(\omega, \alpha, \beta) = f(\omega) + \sum_{i=1}^n \alpha_i g_i(\omega) + \sum_{i=1}^m \beta_i h_i(\omega),$$

el *problema dual* correspondiente se define como:

$$\begin{aligned} \text{Maximizar} \quad & \theta(\alpha, \beta) \quad \omega \in \Omega \\ \text{sujeto a:} \quad & \alpha_i \geq 0 \quad 1 \leq i \leq n \end{aligned}$$

donde $\theta(\alpha, \beta) = \inf_{\omega \in \Omega} L(\omega, \alpha, \beta)$. Los vectores α y β se denominan multiplicadores de Lagrange. Un punto de "silla" de la función de Lagrange generalizada es una tripleta $(\omega^*, \alpha^*, \beta^*)$ con $\omega^* \in \Omega$ y $\alpha_i^* \geq 0$ tal que $L(\omega^*, \alpha^*, \beta^*) \leq L(\omega, \alpha^*, \beta^*) \leq L(\omega, \alpha^*, \beta^*)$ para todo $\omega \in \Omega$ y $\alpha_i \geq 0$. Es decir, en un punto de silla L es mínimo con respecto a ω y máximo con respecto a α , β . Los resultados principales de la teoría se pueden resumir en:

Teorema. Una tripleta $(\omega^*, \alpha^*, \beta^*)$ es un punto de silla de la función de Lagrange generalizada si y sólo si sus componentes son soluciones óptimas de los problemas original y dual respectivamente, cumpliendo $f(\omega^*) = \theta(\alpha^*, \beta^*)$.

Teorema [Kuhn & Tucker 1951]. Supongamos que el dominio $\Omega \subseteq \mathbb{R}^D$ es convexo, $f \in C^1$ convexa y tanto las restricciones g_i como las h_i son funciones lineales. Un punto normal $\omega^* \in \Omega$ es solución del problema original si y sólo si existen α^*, β^* tales que:

$$\frac{\delta L(\omega^*, \alpha^*, \beta^*)}{\delta \omega} = 0, \quad \frac{\delta L(\omega^*, \alpha^*, \beta^*)}{\delta \beta} = 0$$

$$\alpha_i^* g_i(\omega^*) = 0, \quad g_i(\omega^*) \leq 0, \quad \alpha_i^* \geq 0 \quad 1 \leq i \leq n$$

La tercera condición es conocida como la condición de *Karush-Kuhn-Tucker*, y tiene como consecuencia que una solución sólo puede estar en una situación con respecto a cada restricción en forma de desigualdad: o bien está en el interior de la región definida por la desigualdad (restricción inactiva), en cuyo caso $\alpha_i^* = 0$, o bien está en la frontera (restricción activa), con $g_i(\omega^*) = 0$. La eliminación de una restricción inactiva no tiene ningún efecto en la solución del problema.

Los resultados anteriores permiten usar la descripción dual para resolver el problema original evitando trabajar directamente con las desigualdades originales. El problema original se puede transformar en dual simplemente igualando a cero las derivadas de la función de Lagrange generalizada con respecto a las variables originales, y sustituyendo las relaciones obtenidas de nuevo en la función de Lagrange generalizada. Esta idea corresponde a calcular $\theta(\alpha, \beta) = \inf_{\omega \in \Omega} L(\omega, \alpha, \beta)$. Como resultado, se elimina la dependencia de las variables originales. La función resultante sólo contiene variables duales, y debe ser maximizada bajo restricciones más simples. Ésta es la estrategia que se adopta en las secciones de este capítulo al tratar los problemas asociados a las SVM.

Capítulo 15

EXTRACCIÓN DE CONOCIMIENTO CON ALGORITMOS EVOLUTIVOS Y REGLAS DIFUSAS

María José del Jesus, Pedro González y Francisco Herrera

En este capítulo se describen dos técnicas integradas en el área de la computación flexible (*Soft Computing*), la Computación Evolutiva (*Evolutionary Computation*) y la Lógica Difusa (*Fuzzy Logic*), y su uso en el contexto de la minería de datos. Se exponen los algoritmos evolutivos y la lógica difusa desde un punto de vista eminentemente práctico, se explican sus componentes en función de la tarea de minería de datos a realizar y se incide en las posibilidades de su uso no sólo en problemas propios de minería de datos, sino también en problemas de otras etapas del proceso de extracción de conocimiento. Siguiendo la filosofía de la computación flexible, se hace hincapié en las posibilidades de hibridación de técnicas y se describen enfoques evolutivos para la obtención de reglas de asociación difusas.

15.1 Introducción

Tal y como se establece en los capítulos iniciales de este libro, el proceso de extracción de conocimiento en bases de datos ha evolucionado y continúa haciéndolo a través de la investigación interdisciplinar de áreas como las bases de datos, el aprendizaje automático, el reconocimiento de patrones, la estadística, la inteligencia artificial, el razonamiento con incertidumbre, la adquisición de conocimiento para sistemas expertos, la visualización de datos y la computación de alto rendimiento, entre otras. De hecho, cualquier sistema de extracción de conocimiento en bases de datos en general, y de minería de datos en particular, incorpora teorías, algoritmos y métodos de cualquiera de estas áreas.

Es habitual en cualquier proceso de minería de datos que el conjunto de datos a analizar se haya obtenido con un propósito distinto al de la extracción de conocimiento. En el campo médico, por ejemplo, cualquier proceso de extracción de conocimiento se enfrenta a un conjunto de datos en el que se recoge información sobre cualquier aspecto

considerado remotamente relevante aunque, finalmente, para el objetivo del proceso de extracción de conocimiento, no lo sea. No sólo eso, sino que también es muy habitual la presencia de información numérica junto con información textual, de ambigüedades por el uso de diferentes símbolos con igual significado, redundancia, términos perdidos, imprecisos o erróneos, etc. Todo esto hace que se necesiten sistemas robustos de preprocesamiento y una forma de organizar la información antes, durante o después el proceso de minería de datos que permita extraer cualquier tipo de conocimiento de un conjunto de datos de tamaño medio.

El término **Computación Flexible** (*Soft Computing*) agrupa a un conjunto de metodologías cuya característica principal es la tolerancia a imprecisión e incertidumbre, lo que le confiere una capacidad de adaptación que permite solucionar problemas en entornos cambiantes de forma robusta y con bajo coste.

De hecho, el principio que subyace en la computación flexible es la hibridación de técnicas para el desarrollo de métodos computacionales que obtengan una solución aceptable a bajo coste mediante la búsqueda de una solución aproximada a un problema formulado de forma precisa o imprecisa. Dentro de la computación flexible se incluyen metodologías como la lógica difusa, las redes neuronales y la computación evolutiva [Tettamanzi & Tomassini 2001]. Las redes neuronales se han visto en el Capítulo 13. En este capítulo nos ocupamos del resto de estas metodologías.

En el proceso de extracción de conocimiento en bases de datos y, en concreto, en el proceso de minería de datos existen distintas tareas o problemas que se pueden enfocar y resolver como problemas de optimización y búsqueda. Los algoritmos evolutivos imitan los principios de la evolución natural para formar procedimientos de búsqueda y optimización global y son aplicables para el desarrollo de algoritmos de minería de datos propiamente dichos, como algoritmos de pre o pos-procesamiento o como herramientas para la optimización de los parámetros de otros algoritmos [Freitas 2002].

En minería de datos uno de los objetivos a considerar, además de la precisión predictiva y el interés, es la comprensibilidad de los resultados para el usuario. En este aspecto, la lógica difusa constituye una herramienta de representación del conocimiento que permite modelar incertidumbre e imprecisión de una forma sencilla y directamente interpretable por el usuario.

Este capítulo se organiza de la siguiente manera: en la Sección 15.2 se describen características generales de los algoritmos evolutivos, haciendo especial hincapié en los dos tipos de algoritmos evolutivos más utilizados en minería de datos, los algoritmos genéticos y la programación genética. En la Sección 15.3 se muestra la aplicación de estos tipos de algoritmos evolutivos a procesos de minería de datos. En la Sección 15.4 se introducen conceptos generales e intuitivos de lógica difusa para describir, en la Sección 15.5, aspectos de los procesos de minería de datos en los que se puede utilizar la lógica difusa. En la Sección 15.6 se explica la combinación de ambas técnicas para la obtención evolutiva de reglas de asociación difusas y, en la Sección 15.7, se explican dos propuestas concretas que permiten ejemplificar algunos de los aspectos a considerar en el desarrollo de un algoritmo evolutivo para la extracción de reglas de asociación difusas. En la Sección 15.8 se describen algunos sistemas que utilizan algoritmos evolutivos y/o lógica difusa para tareas de minería de datos. Finalmente, en la Sección 15.9 se describen las conclusiones.

15.2 Computación evolutiva

La mayoría de los algoritmos de optimización y búsqueda clásicos utilizan un procedimiento determinístico para alcanzar una solución óptima: comienzan desde un punto aleatorio y se basan en una regla de transición especificada previamente para determinar la dirección de búsqueda. En definitiva realizan una búsqueda unidireccional para encontrar la mejor solución que será considerada como la nueva solución en la siguiente iteración para que el proceso continúe un determinado número de veces. En la Figura 15.1 se muestra este procedimiento. Las distintas propuestas difieren en la determinación de la dirección de búsqueda en cada paso intermedio.

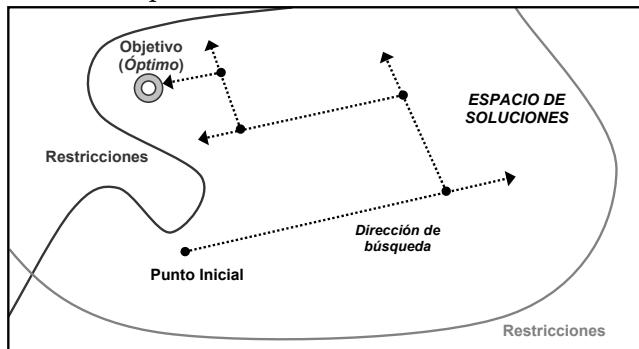


Figura 15.1. Enfoque “punto a punto” de métodos de optimización y búsqueda clásicos.

La mayoría de los métodos de optimización clásicos tienen las siguientes dificultades:

- La convergencia a una solución óptima depende de la solución inicial de la que se parte.
- Pueden quedar atrapados en óptimos locales, como puede observarse en la Figura 15.2.
- No son eficientes en el proceso de solución de problemas con un espacio de búsqueda discreto.
- No se pueden utilizar de forma eficiente en máquinas paralelas.

Los dos primeros problemas pueden solventarse mediante la inclusión de aleatoriedad en los procesos de búsqueda, dando lugar a los métodos estocásticos. No obstante, la mera inclusión de aleatoriedad en los procesos de búsqueda no resuelve los dos últimos problemas.

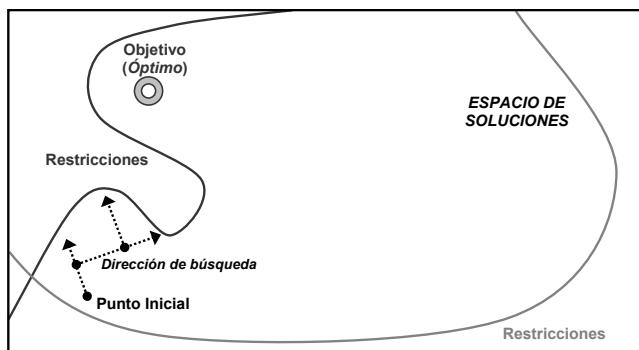


Figura 15.2. Problemas de convergencia y de “óptimos locales” del enfoque “punto a punto”.

La computación evolutiva es un conjunto de algoritmos estocásticos de búsqueda basados en abstracciones del proceso de la evolución de Darwin. Estos algoritmos tienen capacidad para solucionar las dificultades mencionadas anteriormente y se utilizan cada vez con mayor frecuencia para reemplazar a los métodos clásicos en la resolución de problemas reales en los que se presentan estos problemas. En este área se han propuesto distintos modelos computacionales denominados algoritmos evolutivos:

- **Algoritmos Genéticos** [Holland 1975; Goldberg 1989].
- **Estrategias de Evolución** [Schwefel 1995].
- **Programación Evolutiva** [Fogel 1988].
- **Programación Genética** [Koza 1992; Koza 1994].

Todas estas instancias o tipos de algoritmos evolutivos tratan de modelar la evolución y presentan las siguientes características comunes:

1. Utilizan el proceso de aprendizaje colectivo de una **población de individuos**. Normalmente cada individuo representa un punto dentro del espacio de búsqueda de todas las soluciones potenciales para un problema dado, es decir, codifica una solución candidata. Los individuos pueden incorporar adicionalmente otra información, como pueden ser los parámetros de la estrategia del algoritmo evolutivo. En el área de la computación evolutiva a la solución codificada se le denomina genotipo y a la solución decodificada (lo que realmente representa cada individuo en el contexto del problema) se le denomina fenotipo.
2. Los descendientes de los individuos se generan mediante procesos no determinísticos que tratan de modelar los **procesos de mutación y cruce**. La mutación corresponde a una auto-replicación errónea de los individuos, mientras que el cruce intercambia material genético entre dos o más individuos ya existentes. Ambos operadores son estocásticos, se aplican con probabilidades definidas por el usuario. Normalmente se establece una probabilidad de mutación muy inferior a la probabilidad de cruce, ya que una probabilidad de mutación muy elevada convertiría el proceso de búsqueda evolutivo en un proceso de búsqueda aleatoria. No obstante, la mutación es necesaria para incrementar la diversidad genética de individuos dentro de la población y para alcanzar valores de genes que no estén presentes en la población y que de otra forma serían inalcanzables, puesto que el operador de cruce sólo intercambia genes (ya existentes) entre individuos.
3. Se asigna una medida de calidad (denominada habitualmente **medida de adaptación o fitness**) a cada individuo mediante el proceso de evaluación. El **operador de selección** actúa en base a esta medida y favorece, en el proceso de reproducción, a individuos mejores respecto a aquellos con peor valor de la función de adaptación.

El funcionamiento de cualquier algoritmo evolutivo se puede describir de la siguiente forma: se mantiene una población de posibles soluciones para el problema, se realizan modificaciones sobre las mismas y se seleccionan, en función de una medida de adaptación del individuo al entorno, aquellas que se mantendrán en generaciones futuras y las que serán eliminadas. La población evoluciona a través de las mejores regiones del espacio de búsqueda mediante los procesos de modificación y selección. Las modificaciones sobre la población permiten mezclar información de los padres que debe pasar a los descendientes

(operador de cruce) o introducir innovación dentro de la población (operador de mutación). Este proceso se muestra de manera esquemática en la Figura 15.3.

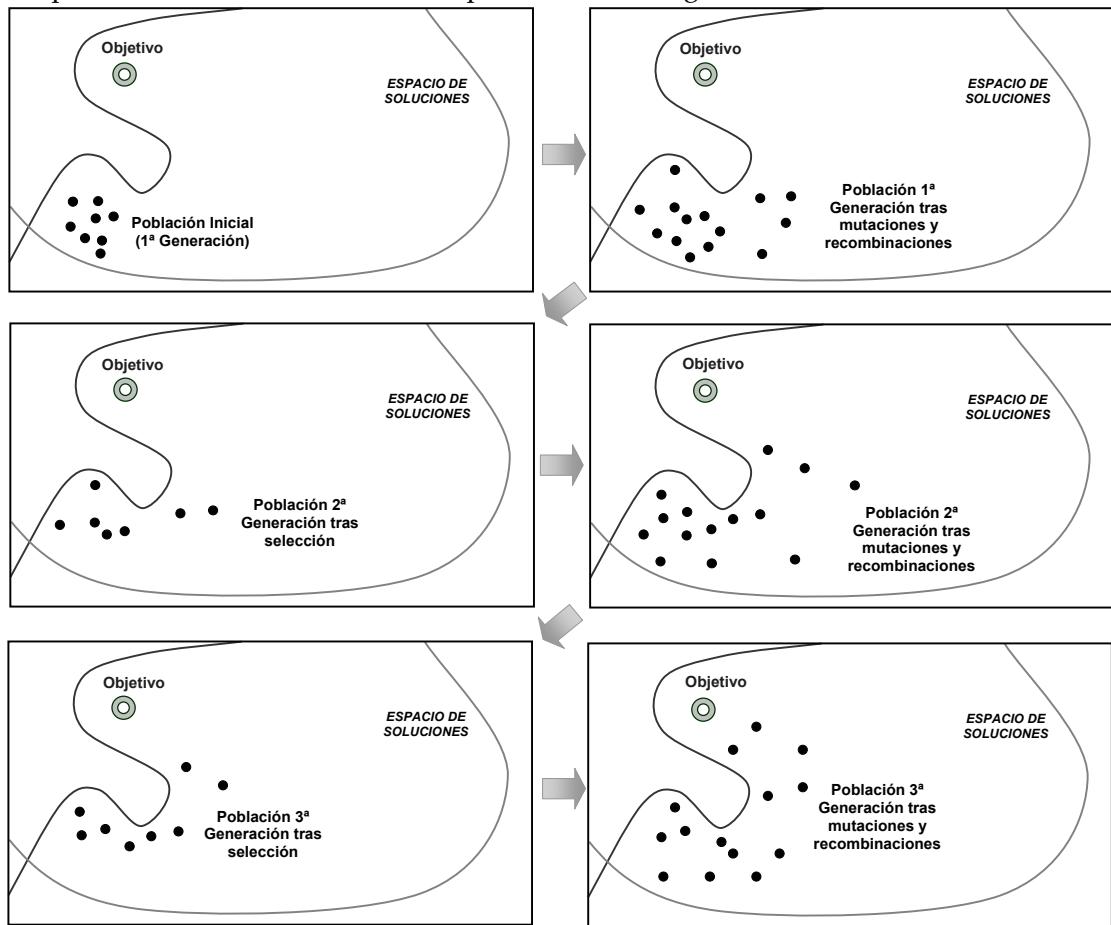


Figura 15.3. Comportamiento de la búsqueda en un enfoque evolutivo.

Una característica importante de los algoritmos evolutivos es que realizan un proceso de búsqueda global. El hecho de que trabajen con una población de soluciones candidatas más que una solución individual, junto con el uso de operadores estocásticos, reduce la probabilidad de caer un óptimo local e incrementa la probabilidad de encontrar el máximo global.

En [Bäck et al. 1997] se puede encontrar una descripción completa de las distintas instancias de algoritmos evolutivos. En los siguientes apartados se describirán las más utilizadas en la actualidad en el campo de la minería de datos: los algoritmos genéticos y la programación genética.

15.2.1 Algoritmos genéticos

Los algoritmos genéticos, en su propuesta original [Holland 1975], se distinguen de otros algoritmos evolutivos por tres características: el esquema de codificación binario, el método de selección (proporcional a la función de adaptación) y el método básico para producir variaciones en la población, el cruce. Es fundamentalmente esta tercera característica la que

hace a los algoritmos genéticos diferentes del resto de los algoritmos evolutivos. En propuestas posteriores a la de Holland se utilizan métodos alternativos de selección y se adoptan esquemas de codificación adaptados a los problemas a resolver y menos restrictivos que el esquema de codificación binario.

El funcionamiento básico de un algoritmo genético es el siguiente (véase Figura 15.4): el sistema parte de una población inicial de individuos que codifican, mediante alguna representación genética, soluciones candidatas al problema propuesto. Esta población de individuos (a los que se denomina cromosomas) evoluciona en el tiempo a través de un proceso de competición y variación controlada. Cada cromosoma de la población tiene asociada una medida de adaptación para determinar qué cromosomas serán seleccionados para formar parte de la nueva población en el proceso de competición. La nueva población se creará utilizando operadores genéticos de cruce y mutación. Este ciclo evolutivo continúa hasta que se verifique una determinada condición de parada: que se hayan realizado un determinado número máximo de evaluaciones de individuos, que la población haya evolucionado durante un número máximo de generaciones, que se haya alcanzado una solución con un determinado valor de la función de adaptación (o de parte de ella), que se estacione la población por no generarse individuos nuevos durante un determinado número de generaciones, etc.

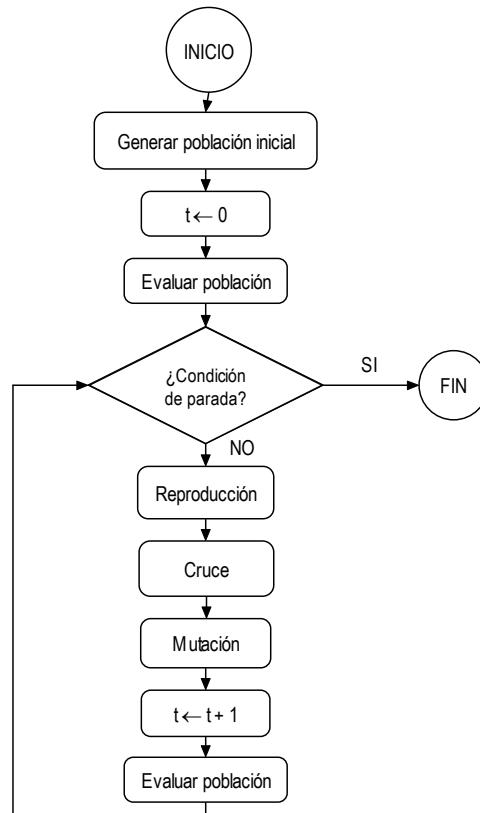


Figura 15.4. Algoritmo genético básico.

La aplicación de un algoritmo genético para resolver un problema debe determinar:

- Una representación genética de las soluciones del problema.

- Una forma de crear una población inicial de soluciones.
- Una función de evaluación que proporcione un valor de adaptación de cada cromosoma.
- Operadores que modifiquen la composición genética de la descendencia durante la reproducción.
- Valores para los parámetros que utilizan (tamaño de la población, probabilidades de aplicación de los operadores genéticos, etc.).

En las siguientes secciones, veremos ejemplos de estos aspectos.

15.2.2 Programación genética

La programación genética se puede definir como un algoritmo genético aplicado a *programas de computador* con el objeto de evolucionar los propios programas para obtener un programa resultado eficiente y significativo que resuelva una tarea [Koza 1992]. De igual forma que en un algoritmo genético se codifican variables de decisión, en la programación genética se codifica un programa que representa un procedimiento para resolver una tarea. Habitualmente la codificación se realiza en forma de árbol.

Los operadores de cruce y mutación son similares a los utilizados en algoritmos genéticos pero adaptados al esquema de representación empleado. En el caso del cruce (en una propuesta básica), para intercambiar un subprograma (una parte de material genético) se determina aleatoriamente el punto del árbol a intercambiar en ambos padres y posteriormente se intercambian los dos subprogramas. El operador de mutación se puede aplicar a un terminal o a una función (nodo no terminal). Si se aplica a un terminal, se cambia su valor actual por otro valor del conjunto de terminales. Si se aplica a un nodo, se elige una función diferente del conjunto de funciones.

De forma similar a los algoritmos genéticos, en la programación genética se utilizan los operadores genéticos durante un número pre-especificado de generaciones o evaluaciones, hasta que se alcance una solución con un valor de adaptación establecido o hasta que el algoritmo se estabilice. En la bibliografía especializada se han propuesto distintos operadores avanzados que definen de forma automática funciones, cruces con significado, etc. En secciones siguientes veremos algunos ejemplos de estos operadores.

15.3 Algoritmos evolutivos para la extracción de conocimiento

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

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

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

15.3.1 Algoritmos genéticos para el descubrimiento de reglas

Como hemos dicho anteriormente, cualquier propuesta de algoritmo genético de extracción de reglas, cuyo objetivo sea obtener reglas, debe determinar el esquema de representación utilizado para codificar cada una de las soluciones, los operadores genéticos y la función de adaptación. En las siguientes subsecciones explicaremos cada uno de estos aspectos y, por último, al final de esta sección, veremos un ejemplo sencillo de algoritmo genético aplicado al proceso de descubrimiento de reglas.

Esquema de representación

Los algoritmos genéticos siguen dos enfoques respecto a la forma de codificar reglas dentro de una población de individuos:

- El enfoque "*Cromosoma = Regla*", en el que cada individuo codifica una única regla.
- El enfoque "*Cromosoma = Base de Reglas*", también denominado enfoque *Pittsburgh*, en el que cada individuo representa un conjunto de reglas.

Dentro del enfoque "*Cromosoma = Regla*" existen dos propuestas genéricas:

- El enfoque *Michigan* en el que cada individuo codifica una única regla pero la solución final será la población final o un subconjunto de la misma. En este caso, es necesario evaluar el comportamiento del conjunto de reglas al completo y la aportación de la regla individual al mismo.
- El enfoque *IRL (Iterative Rule Learning)*, en el que cada cromosoma representa una regla, pero la solución del algoritmo genético es el mejor individuo y la solución global está formada por los mejores individuos de una serie de ejecuciones sucesivas.

La elección del esquema de representación depende, entre otros aspectos, de la tarea a realizar por parte del algoritmo de minería de datos y, por tanto, del tipo de regla a descubrir.

Si el objetivo es determinar un conjunto de reglas de clasificación, se debe evaluar el comportamiento del conjunto de reglas al completo más que la calidad de una regla individual. El esquema de representación que, a primera vista, parece más adecuado es el enfoque "*Cromosoma = Base de Reglas*" que considera la interacción entre las reglas. GABIL

[De Jong et al. 1993] y GIL [Janikow 1993] son ejemplos de algoritmos genéticos para clasificación que utilizan este esquema de representación. No obstante, este enfoque también tiene problemas puesto que implica el uso de individuos con una longitud superior (y a menudo variable), lo que provoca un incremento del coste computacional del algoritmo y la modificación de los operadores genéticos. Esto hace que dentro del campo de la clasificación también se hayan diseñado algoritmos genéticos con el enfoque “*Cromosoma = Regla*”, como COGIN [Greene & Smith 1993] y REGAL [Giordana & Neri 1995], que utilizan individuos con una sintaxis más reducida simplificando el diseño de los operadores genéticos. Este enfoque tiene dos inconvenientes: la dificultad del cálculo del valor de la función de adaptación, ya que cada regla se evalúa individualmente y es difícil determinar la calidad del conjunto de reglas al completo; y por otra parte, puesto que el objetivo es obtener un conjunto de reglas, el algoritmo genético no debería converger hacia un único individuo. Para evitar esto se necesita alguna técnica de *nichos* [Beasley et al. 1993] que fomente la existencia de individuos distintos dentro de la población.

Por el contrario, en procesos de descubrimiento de reglas de asociación es más adecuado el enfoque “*Cromosoma = Regla*” ya que el objetivo suele ser encontrar un conjunto reducido de reglas en las que la calidad de cada regla se evalúa de forma independiente al resto.

Una vez descritos los dos enfoques genéricos de codificación de reglas, explicaremos de forma detallada distintos esquemas de codificación del antecedente y consecuente en función del tipo de regla a obtener. Para ello, en gran parte de la descripción nos centraremos en la codificación de una única regla, de forma que los conceptos introducidos se puedan adaptar a los distintos enfoques de codificación mencionados.

Una regla se puede describir de forma genérica de la siguiente forma:

SI <condición 1> Y <condición 2> Y ... Y <condición N> ENTONCES <consecuente>
donde:

- Las condiciones expresadas en el antecedente pueden implicar atributos nominales o numéricos. Si los atributos son numéricos, se puede utilizar un proceso de discretización que establezca una correspondencia entre valores e intervalos, un proceso de agrupamiento, o un esquema de representación del conocimiento más avanzado (como el que se describe a partir de la Sección 15.4), entre otras soluciones.
- El número de condiciones que aparecen en el antecedente es variable.
- Cada una de las condiciones puede estar formada a su vez por una disyunción de condiciones individuales.
- Si la regla es de clasificación, en el consecuente sólo aparecerá una condición que implica a un atributo, el atributo de clase o atributo objetivo, y éste no puede estar contenido en el antecedente.
- Si la regla determina dependencias funcionales, se puede considerar como una generalización de una regla de clasificación, con la salvedad de que en el consecuente puede aparecer más de un atributo objetivo.
- Si la regla es de asociación no tiene porqué existir necesariamente una diferenciación entre atributos predictores y objetivo, y también puede existir más de una condición en el consecuente.

Dado que el antecedente de una regla está formado por un número de condiciones variable, la representación del mismo en un individuo lleva, de forma inmediata, a un esquema de codificación de longitud variable. Así, si restringimos el antecedente de las reglas a descubrir a una conjunción de atributos, un posible esquema de codificación puede estar formado por un número variable de parejas atributo-valor. Para cada una de estas parejas si el atributo es nominal el valor se codifica mediante un número entero. Si es numérico se puede codificar en binario el valor numérico correspondiente, o discretizar el dominio y asignar un entero a cada uno de los intervalos, entre otras soluciones. El uso de una representación no binaria permite el tratamiento uniforme de los atributos numéricos frente a los nominales.

En la Figura 15.5 se describe la codificación del antecedente de una regla para el problema *Wisconsin Diagnostic Breast Cancer* (véase el Apéndice B) en el que todas las variables son numéricas y se ha realizado una discretización del dominio.



Variable 1 (radio):	Variable 3 (perímetro):	Variable 9 (simetría):
1 ⇒ radio < 10	1 ⇒ perímetro < 100	1 ⇒ simetría < 0.2
2 ⇒ 10 ≤ radio < 15	2 ⇒ 100 ≤ perímetro < 165	2 ⇒ 0.2 ≤ simetría < 0.35
3 ⇒ 15 ≤ radio < 20	3 ⇒ perímetro ≥ 165	3 ⇒ 0.35 ≤ simetría < 0.5
4 ⇒ 20 ≤ radio < 25		4 ⇒ 0.5 ≤ simetría < 0.6
5 ⇒ radio ≥ 25		5 ⇒ simetría ≥ 0.65

Figura 15.5. Ejemplo de codificación entera de longitud variable para el antecedente de una regla.

El uso de reglas de longitud variable obliga a la modificación del operador de cruce para asegurar que los descendientes resultantes sean individuos válidos. Para evitar esto, se pueden codificar todos los atributos, pero estableciendo una marca especial que indique que la variable correspondiente no interviene en la regla. De esta forma, una posible estrategia de codificación puede representar el antecedente mediante una cadena de números enteros de longitud N , siendo N el número total de atributos, como se muestra en la Figura 15.6, donde se marcan con un 0 las variables que no intervienen en la regla. En ella se puede observar que, al estar las variables siempre alineadas, el operador de cruce produce descendientes válidos [Noda et al. 1999].



Variable 1 (radio):	Variable 3 (perímetro):	Variable 9 (simetría):
1 ⇒ radio < 10	1 ⇒ perímetro < 100	1 ⇒ simetría < 0.2
2 ⇒ 10 ≤ radio < 15	2 ⇒ 100 ≤ perímetro < 165	2 ⇒ 0.2 ≤ simetría < 0.35
3 ⇒ 15 ≤ radio < 20	3 ⇒ perímetro ≥ 165	3 ⇒ 0.35 ≤ simetría < 0.5
4 ⇒ 20 ≤ radio < 25		4 ⇒ 0.5 ≤ simetría < 0.6
5 ⇒ radio ≥ 25		5 ⇒ simetría ≥ 0.65

Figura 15.6. Ejemplo de codificación entera de longitud fija para el antecedente de una regla.

Si queremos incluir en el antecedente una condición compuesta por una conjunción entre atributos y varias disyunciones sobre los valores del atributo (es decir, que cada atributo pueda tomar varios valores dentro del antecedente de la regla), se puede optar por un esquema de codificación binario, en el que por cada atributo se almacene un bit para cada uno de los valores que puede tomar, de forma que si el bit correspondiente tiene valor 0

indica que no pertenece a la condición y si tiene valor 1 que sí pertenece [DeJong et al. 1993]. Si en un individuo todos los bits correspondientes a un atributo tienen valor 1 indican que dicha variable no es relevante para la información aportada en la regla (cualquier valor de la variable verifica la condición de la regla), por lo que esta variable se ignora. La Figura 15.7 muestra el antecedente de una regla y su codificación según este esquema.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	\dots	v_{30}
10001	000	010	00	000	00	000	00	10000	0	...	0
—————> SI (radio < 10 O radio \geq 25) Y 100 \leq perímetro < 165										Y simetría < 0.2	
Variable 1 (radio):				Variable 3 (perímetro):				Variable 9 (simetría):			
1 \Rightarrow radio < 10				1 \Rightarrow perímetro < 100				1 \Rightarrow simetría < 0.2			
2 \Rightarrow 10 \leq radio < 15				2 \Rightarrow 100 \leq perímetro < 165				2 \Rightarrow 0.2 \leq simetría < 0.35			
3 \Rightarrow 15 \leq radio < 20				3 \Rightarrow perímetro \geq 165				3 \Rightarrow 0.35 \leq simetría < 0.5			
4 \Rightarrow 20 \leq radio < 25								4 \Rightarrow 0.5 \leq simetría < 0.6			
5 \Rightarrow radio \geq 25								5 \Rightarrow simetría \geq 0.65			

Figura 15.7. Ejemplo de codificación entera de longitud fija para antecedentes de reglas DNF.

Como vemos, existen bastantes opciones para representar el antecedente y se optará por una u otra en función del problema y de las preferencias de implementación. Respecto al consecuente, si el objetivo del proceso de minería de datos es extraer reglas de clasificación, la codificación de la clase se puede realizar bajo alguno de estos enfoques:

- Codificarlo en el genoma del individuo [DeJong et al. 1993] de forma que también pueda evolucionar.
- Asociar todos los individuos de la población con la misma clase y ejecutar el algoritmo tantas veces como clases distintas existan [Janikow 1993].
- Elegir, de forma determinística, la clase más adecuada para el antecedente de la regla: la clase que tenga más representantes en el conjunto de ejemplos que verifican el antecedente de la regla [Giordana & Neri 1995] o la clase que maximiza la función de adaptación del individuo [Noda et al. 1999].

La primera y tercera opciones tienen la ventaja de permitir que los individuos de la población representen reglas de distintas clases y no sea necesario realizar múltiples ejecuciones del algoritmo evolutivo para descubrir reglas de las distintas clases.

Si el objetivo es descubrir reglas para expresar dependencias funcionales, es necesario representar en el consecuente no sólo una variable (el atributo objetivo) sino más de una, por lo que, en ocasiones, es conveniente optar por un esquema de codificación del consecuente que represente una conjunción de pares atributo-valor con longitud variable [Araujo et al. 2000].

Operadores genéticos

Además de los operadores genéticos clásicos (adaptados al esquema de representación utilizado), existen distintas propuestas de operadores genéticos diseñados específicamente para algoritmos genéticos cuyo objetivo es el descubrimiento de reglas.

Para algoritmos genéticos con el esquema de representación “Cromosoma = Regla”, es necesario evitar la convergencia de la población a un individuo particular, ya que un solo individuo no es una solución válida para el problema. Esto se resuelve en REGAL [Giordana & Neri 1995] mediante un operador de selección denominado *sufragio universal*

que elige los individuos que deben participar en la recombinación mediante una elección por sufragio de los ejemplos de entrenamiento. Para ello, cada ejemplo vota por la regla que determina la zona del espacio de búsqueda en la que está incluido, de forma proporcional a la medida de adaptación de la regla. Es una estrategia de *nichos* que potencia la evolución de diferentes reglas que cubren diferentes zonas del espacio de búsqueda.

Una de las operaciones más adecuadas en el proceso de descubrimiento de reglas es la operación de generalización-especialización que se lleva a cabo, básicamente, eliminando o añadiendo condiciones en el antecedente. En la bibliografía especializada existen propuestas en este sentido integradas en operadores de cruce, de mutación o bien operadores diseñados específicamente para ello.

En un algoritmo genético en el que evolucionan reglas con distintos consecuentes no es conveniente intercambiar material genético entre individuos (reglas) con consecuentes distintos, pues se considera que es información que no está relacionada. Esto se puede solucionar con un operador de cruce que opere sólo dentro de subpoblaciones, entendiendo que una subpoblación está formada por los individuos que representan reglas con igual consecuente [Araujo et al. 2000].

Función de adaptación

En el proceso de descubrimiento de reglas se intenta conseguir reglas con capacidad predictiva alta, comprensibles e interesantes. Esto se puede conseguir en el contexto evolutivo mediante una combinación lineal con pesos de estas tres medidas [Noda et al. 1999], mediante una matriz de confusión, o bien mediante un algoritmo genético con enfoque *multiobjetivo* [Deb 2001] que orienta el proceso evolutivo hacia la búsqueda de un conjunto de soluciones no dominadas en los distintos objetivos marcados. Las soluciones no dominadas son aquellas para las que no existe ninguna otra solución candidata que las supere en todos los objetivos marcados.

Algunas de las propuestas descritas en esta sección corresponden a algoritmos de descubrimiento de conocimiento con capacidad para trabajar sólo con bases de ejemplos de tamaño moderado, es decir, no son escalables respecto al tamaño del conjunto de ejemplos a explorar. Afortunadamente, surgen, de forma natural, propuestas que aprovechan el procesamiento paralelo en el contexto de los algoritmos genéticos para tareas de minería de datos que trabajen con grandes volúmenes de datos [Freitas & Lavington 1998].

En el contexto de la minería de datos, la etapa con mayor coste computacional de cualquier algoritmo evolutivo es la correspondiente a la evaluación de los individuos, por lo que es ésta la operación que se paraleliza cuando existe procesamiento paralelo. Freitas y Lavington clasifican en dos los enfoques que calculan de forma paralela la función de adaptación de los individuos (véase la Figura 15.8):

- Enfoque a): distribución de los individuos en los distintos procesadores y cálculo independiente de la función de adaptación. Los algoritmos que siguen este enfoque, si utilizan una arquitectura con memoria distribuida, están obligados a almacenar en cada procesador una copia completa del conjunto de datos para determinar el valor de la función de adaptación de los individuos que le corresponden y así redu-

- cir el tráfico de datos en la red entre los procesadores. Este aspecto dificulta la escalabilidad de los algoritmos resultantes para grandes bases de datos.
- Enfoque b): Distribución de los ejemplos entre los distintos procesadores y cálculo parcial de la función de adaptación de un individuo en cada procesador.

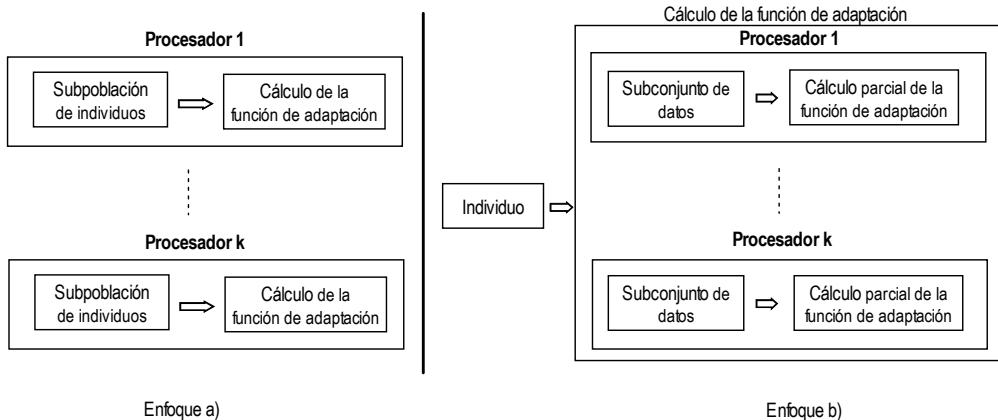


Figura 15.8. Dos formas distintas de paralelizar el cálculo de la función de adaptación.

Los sistemas GA-MINER [Flockart & Radcliffe 1995] y GA-PVMINER [Araujo et al. 2000] son ejemplos de algoritmos genéticos paralelos en minería de datos.

Algoritmo genético básico para el descubrimiento de reglas

Para mostrar de forma práctica los conceptos introducidos, describimos a continuación una propuesta sencilla para la extracción de reglas de asociación (véase el Capítulo 9) con un algoritmo genético.

El algoritmo genético que vamos a describir tiene como objetivo descubrir una regla de asociación para un atributo objetivo prefijado. Para ello codifica en un cromosoma el antecedente de una regla, es decir, sigue el esquema de codificación “Cromosoma = Regla”. El consecuente no lo codifica, sino que cada ejecución del algoritmo genético se realiza para cada uno de los valores del atributo objetivo.

Los componentes del algoritmo se detallan a continuación:

- **Representación.** Cada cromosoma codifica sólo el antecedente de la regla ya que el consecuente es fijo durante cada ejecución del algoritmo genético. El antecedente de la regla se representa en un cromosoma de longitud fija con un esquema de codificación entera (la posición i -ésima indica el valor que toma la variable i -ésima). Para las variables numéricas, en una etapa de pre-procesamiento, se determina un conjunto de intervalos, y la base de ejemplos se discretiza asignándole a cada valor numérico el número de intervalo al que pertenece. Existen múltiples propuestas para la discretización [Liu et al. 2002], como se vio en la Sección 4.4.1, pero en este ejemplo sencillo se ha optado por la más inmediata, la división uniforme (equidistante) del dominio de cada variable en un número de intervalos. El esquema de codificación incluye siempre un valor adicional para cada gen para indicar que la variable correspondiente no participa en la regla y corresponde al que se representa en la Figura 15.6.

- **Función de adaptación.** La función de adaptación es la media aritmética de la completitud (soporte) y la confianza de la regla representada en el cromosoma. La confianza determina la precisión de la regla, ya que refleja el grado con el que los ejemplos pertenecientes a la zona del espacio delimitado por el antecedente verifican la información indicada en el consecuente de la regla. Esta medida se calcula como el número de ejemplos de la clase que pertenecen a la zona determinada por el antecedente entre el número de todos los ejemplos (independientemente de su clase) que pertenecen a la misma zona. La completitud mide el grado de cobertura de la regla con respecto a los ejemplos de la clase, y se calcula como el cociente entre el número de ejemplos de la clase que pertenecen a la zona determinada por el antecedente y el número total de ejemplos de la clase.
- **Operadores genéticos.** El algoritmo genético utiliza un modelo de *reproducción* de estado estacionario. En éste, todos los individuos de población se mantienen en el proceso evolutivo salvo dos, los dos peores que serán sustituidos por los individuos resultantes de cruzar y mutar los dos mejores individuos de la población. El operador de *cruce* que se ha utilizado es el cruce en dos puntos que funciona de la siguiente forma: para cada pareja de cromosomas a cruzar se seleccionan aleatoriamente dos puntos en la longitud del cromosoma y se intercambia el material genético entre estos dos puntos dando lugar a dos descendientes. El operador de *mutación* que se utiliza es la mutación uniforme que selecciona aleatoriamente una posición del cromosoma (un valor de una variable del antecedente de la regla) y cambia su valor por uno obtenido aleatoriamente de entre los valores posibles de la variable.

Para ilustrar el funcionamiento del algoritmo, hemos realizado una experimentación sobre la base de ejemplos *Wisconsin Diagnostic Breast Cancer* (*wdbc*, véase el Apéndice B), en los que, a partir de 30 variables de entrada de un paciente se ha de determinar una variable de salida, que indica si el cáncer es benigno o maligno. Como todas las variables de entrada que intervienen son numéricas, se ha realizado un proceso de discretización en cinco y siete intervalos. Se ha utilizado una población de 100 individuos, y como condición de terminación del algoritmo genético se ha establecido que se alcancen 5.000 evaluaciones, es decir, el algoritmo genético terminará cuando se hayan evaluado 5.000 cromosomas.

El carácter estocástico de los algoritmos genéticos determina que para mostrar su buen funcionamiento deban realizarse varias ejecuciones, y evaluar el mejor, el peor y la desviación típica de los resultados, entre otras medidas. Para este ejemplo se han realizado cinco ejecuciones para cada una de las clases como atributo objetivo, aunque por las limitaciones de espacio y el carácter de este capítulo mostraremos sólo los mejores resultados de ese conjunto de experimentaciones. En la Tabla 15.1 se muestran los resultados obtenidos y la regla generada en cada caso.

El algoritmo genético permite obtener una regla para cada clase con un valor alto de confianza y un nivel de completitud adecuado. Con esta primera aproximación al problema se ha conseguido con un algoritmo genético sencillo obtener reglas de asociación adecuadas. Los resultados se podrían mejorar con un enfoque que permita obtener más de una regla, un esquema de codificación que permita representar reglas más flexibles, un mejor método de discretización, etc., así como algunas técnicas que veremos en el resto del capítulo.

Nº Interv.	Compleitud	Confianza	Regla
5	84,314	96,166	SI (0 <= Mean Concave Points < 0,024) Y (0 <= Radius SE < 0,6) Y (0 <= Area SE < 110) ENTONCES Benigno
	51,415	97,321	SI (1008 <= Worst Area < 1831) ENTONCES Maligno
7	89,356	94,379	SI (0 <= Area SE < 78,571) Y (185 <= Worst Area < 772,857) ENTONCES Benigno
	33,491	100,000	SI (21,428 <= Worst Radius < 26,071) ENTONCES maligno

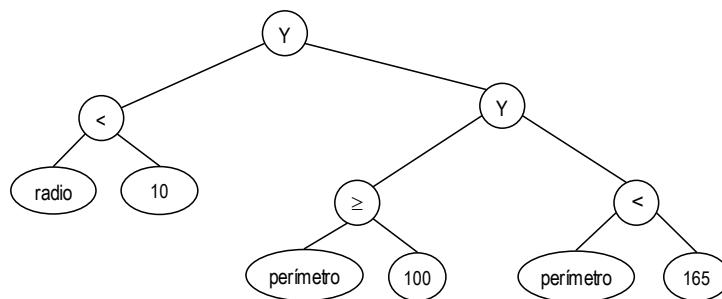
Tabla 15.1. Resultados obtenidos con el Algoritmo Genético Básico para la obtención de reglas intervalares.

15.3.2 Programación genética para el descubrimiento de reglas

La búsqueda que lleva a cabo la programación genética puede ser muy útil en tareas de predicción, puesto que el sistema puede producir combinaciones de atributos que no consideraría un algoritmo genético convencional.

Debemos tratar dos aspectos relacionados con el uso de programación genética para descubrimiento de reglas: la representación de los individuos y la expresión de la función de adaptación.

En programación genética, un individuo se suele representar mediante un árbol de expresiones sintácticas, donde los nodos hoja o terminales son condiciones de reglas y/o valores de atributos, y los nodos internos representan las funciones. La aplicación de la programación genética estándar a la tarea de extracción de reglas, por ejemplo de clasificación, es relativamente directa, si todos los atributos son numéricos. En este caso podemos incluir en el conjunto de funciones varios tipos de funciones matemáticas apropiadas para el dominio de la aplicación e incluir en un conjunto de terminales los atributos predictores. En la Figura 15.9 se muestra un ejemplo de codificación para el antecedente de una regla.



Si radio < 10 Y 100 ≤ perímetro < 165

Figura 15.9. Ejemplo de representación de un individuo en programación genética para descubrimiento de reglas.

En procesos de minería de datos en los que se combine información numérica con información nominal, la codificación de reglas comprensibles en un individuo no es un problema trivial, debido a la “propiedad de cierre” de la programación genética. Esta propiedad obliga a que la salida de un nodo pueda utilizarse como entrada de cualquier nodo padre en el árbol, para que así se puedan hacer los cruces. El problema es que esto no se verifica siempre si se combinan distintos tipos de variables. Para este problema, entre las soluciones que se ofrecen en la literatura especializada, destacan las siguientes:

- Utilizar una sintaxis limitada. Para ello, el usuario especifica para cada función disponible en el conjunto de funciones, el tipo de sus argumentos y de sus resultados, y los operadores de cruce y mutación se modifican para crear sólo árboles válidos, respecto a las restricciones indicadas en la sintaxis [Battacharyya et al. 1998].
- Utilizar una gramática definida por el usuario y dependiente del dominio que especifique la sintaxis de las reglas válidas. La gramática también se puede utilizar para incorporar conocimiento específico del dominio [Wong & Leung 2000].
- Modificar los datos para conseguir la propiedad de cierre. Un ejemplo de este enfoque consiste en expresar todos los atributos de los datos, de forma lógica (booleana) y utilizar operadores lógicos (AND, OR) en el conjunto de funciones [Eggermont et al. 1999]. De esta forma, la salida de cualquier nodo del árbol será un valor de tipo lógico, que se puede utilizar como entrada de cualquier operador lógico en el correspondiente nodo padre.

Al igual que con los algoritmos genéticos, la forma de codificar el consecuente de las reglas depende del tipo reglas a descubrir. Si son reglas de clasificación puede no ser necesario codificar el consecuente, bien porque se ejecute el programa genético tantas veces como clases distintas existan o porque estemos en el caso de un problema con sólo dos clases en el que una vez obtenida una regla que describa bien una clase, los datos que no verifiquen dicha regla serán de la clase opuesta. Si estamos en un problema de clasificación con múltiples clases y no se quiere ejecutar repetidamente el programa genético, se puede codificar (y evolucionar) el consecuente. En este caso será necesario incluir en el programa genético un enfoque de nichos que impulse la evolución hacia individuos distintos (con consecuentes distintos en este caso). De igual manera, el proceso de descubrimiento de reglas generalizadas, reglas de asociación, etc., implica la codificación del consecuente como se describió en la sección anterior.

El sistema LOGENPRO [Wong & Leung 2000] combina programación genética con Programación Lógica Inductiva (ILP, *Inductive Logic Programming*, véase el Capítulo 12) para evolucionar programas lógicos de primer orden y tiene una extensión que, utilizando una gramática adecuada, permite generar reglas con distintos formatos.

Respecto a la función de adaptación, en principio, en la programación genética para el descubrimiento de reglas se podrían utilizar funciones de adaptación similares a las utilizadas por los algoritmos genéticos para el descubrimiento de reglas, con algunas pequeñas diferencias. En particular, puesto que el tamaño de los árboles en la programación genética puede crecer mucho, es conveniente incorporar alguna medida de comprensibilidad de la regla en la función de adaptación. Un ejemplo de medida subjetiva de comprensibilidad es la que viene determinada por el tamaño de la regla y la cardinalidad del conjunto de reglas. Esto implica que estamos interesados en sistemas de programación genética que devuelvan un pequeño número de reglas pequeñas. La forma más sencilla de favorecer el descubrimiento de reglas pequeñas es incluir una penalización para las reglas largas en la función de adaptación [Bojarczuk et al. 2000].

En [Ryan & Rayward-Smith 1998] se describe un sistema híbrido que combina programación genética con árboles de decisión. El algoritmo utiliza la programación genética para evolucionar una población de árboles de decisión y un proceso de construcción de árboles

de decisión para generar la población inicial y modificar árboles individuales durante la evolución.

Como hemos visto, el problema de extracción de reglas en minería de datos se puede abordar tanto con un enfoque basado en algoritmos genéticos como en programación genética. Si se opta por la segunda opción deberá determinarse una gramática adecuada para generar finalmente el tipo de reglas deseado. En el caso de los algoritmos genéticos se debe especificar un esquema de codificación adecuado para generar de forma flexible y eficiente el tipo de reglas objeto del proceso de minería de datos.

15.3.3 Algoritmos evolutivos en otras etapas del proceso de extracción de conocimiento

La aplicación de un algoritmo de minería de datos a un conjunto de datos es la etapa central del proceso de extracción de conocimiento. No obstante, y como vimos en el Capítulo 2, en este proceso se incluyen otras etapas: comprensión del problema, comprensión de los datos, pre-procesamiento (o preparación) de datos, minería de datos y pos-procesamiento (evaluación e interpretación de los modelos). En esta sección describiremos algunas propuestas evolutivas para los procesos incluidos en la etapa de pre-procesamiento y mostraremos dos ejemplos de algoritmos genéticos sencillos para estas tareas.

Algoritmos evolutivos en la etapa de pre-procesamiento de datos

Gran parte de la investigación en extracción de conocimiento está centrada en el problema del tamaño de los datos a los que se enfrenta, ya sea por alta dimensionalidad o gran número de ejemplos. Este problema se afronta a través de dos vías: incrementando la escalabilidad de los algoritmos de minería de datos y/o reduciendo el propio tamaño de los datos.

Siguiendo por esta segunda vía, la reducción del tamaño de los datos se puede realizar de distintas formas, como vimos en los capítulos 4 y 5:

- Seleccionando variables y por tanto, reduciendo el número de columnas del conjunto de datos.
- Discretizando los valores de las variables, con lo que se reduce el número de valores posibles de cada una de ellas.
- Seleccionando instancias, es decir, reduciendo el número de ejemplos (filas) del conjunto de datos.

Los algoritmos evolutivos se han utilizado con éxito para resolver el problema de *selección de instancias*, entendiendo como tal la determinación del subconjunto de ejemplos más significativos dentro del conjunto total (no es, por tanto, un muestreo aleatorio). Para este problema, la codificación más inmediata de una solución candidata es a través de un cromosoma binario de longitud igual al número total de ejemplos. La función de adaptación debe ser una medida combinada de la reducción y una estimación de la precisión alcanzable con los ejemplos seleccionados. En [Cano et al. 2003] se realiza un estudio experimental de distintas propuestas evolutivas para el problema de selección de

instancias. Liu y Motoda editan el libro [Liu & Motoda 2001], en el que se puede encontrar un estado del arte en el tema.

La *selección de variables, características o atributos* consiste en la determinación de un subconjunto de atributos con máxima relevancia del conjunto total del que se dispone, para predecir el valor del atributo objetivo, si el objetivo es clasificar, o para extraer información interesante y relevante, si el objetivo es extraer reglas de asociación, por ejemplo. Los algoritmos evolutivos resuelven bien este problema porque tratan la interacción entre variables de forma más adecuada y global que los enfoques de búsqueda local que consideran la inclusión y/o eliminación de variables de forma individual.

En la literatura especializada se han desarrollado múltiples propuestas de algoritmos genéticos para la selección de atributos aplicadas principalmente a problemas de clasificación (y extensibles a otro tipo de problemas). Estas propuestas se diferencian fundamentalmente en la definición de los componentes principales del algoritmo genético:

- El *esquema de codificación* utilizado depende del objetivo de la selección de atributos. Así, si el objetivo es determinar el mejor conjunto de variables con una cardinalidad prefijada m , es adecuado el uso de cromosomas con codificación de m enteros en los que el i -ésimo gen representa la i -ésima variable seleccionada [Liu y Motoda 1998]. Por el contrario, si el proceso busca el conjunto mínimo de variables que maximice una función de adaptación dada, se utilizan cromosomas de longitud igual al número total de variables originales usando una codificación binaria, en la que un 1 en el i -ésimo gen indica la presencia de la i -ésima característica y un 0 su ausencia [Yang & Honavar 1998].
- La *función de adaptación* depende del enfoque del algoritmo de selección de atributos. De esta forma, se puede distinguir entre las funciones de adaptación incluidas en métodos de selección de características filtro, y métodos envolventes (“wrapper”) de selección de atributos, como vimos en la Sección 5.4.2.

Se han desarrollado propuestas genéticas con función de adaptación de método filtro basada en la medida de información mutua entre variables y en la ratio de inconsistencia introducido por la eliminación de variables. En [Liu & Motoda 1998b] se puede encontrar una revisión amplia de distintas medidas.

Respecto al enfoque envolvente, la definición de la función de adaptación como la estimación de la precisión proporcionada por la regla del vecino más cercano con cada subconjunto de atributos forma parte de distintas propuestas, fundamentalmente por la rapidez de su cálculo. También se han utilizado otros modelos de aprendizaje distintos al vecino más cercano como la ratio de clasificación correcta obtenido por una red neuronal [Yang & Honavar 1998].

La función de adaptación también puede combinar medidas independientes, por ejemplo, una medida de entropía sobre las variables, una estimación del coste de extracción de las variables seleccionadas y la ratio de clasificación obtenido por un árbol de decisión construido mediante C4.5 [Bala et al. 1996]. De esta forma, se puede obtener un algoritmo evolutivo de selección de atributos híbrido filtro-envolvente.

Los resultados obtenidos por estudios comparativos de propuestas de algoritmos genéticos frente a otros tipos de algoritmos de selección de características [Yang & Honavar 1998; Kudo & Skalansky 2000] en problemas de selección de características de gran escala

muestran que los algoritmos genéticos constituyen una de las principales opciones para obtener de forma razonable subconjuntos de variables adecuados.

En los procesos de *extracción o transformación de atributos* (véase el Capítulo 4) se busca una transformación, normalmente lineal, de las variables que ayude a eliminar información redundante o irrelevante, y muestre la importancia relativa de las variables seleccionadas, como el análisis de componentes principales. Los procesos de extracción de atributos engloban a los de selección de atributos en el caso en el que se asocie un peso a cada variable igual a 0 o 1. La opción más natural para representar pesos es la codificación real (el valor real del gen i representa el peso asociado a la variable i -ésima) [Punch et al. 1993]. En ocasiones se ha abordado la selección y extracción de atributos de forma conjunta codificando, con un esquema de codificación mixto (binario y real), en un mismo cromosoma tanto la presencia o ausencia de una variable como el peso asociado a la misma [Raymer et al. 2000].

Una generalización aún mayor es lo que se denomina *construcción de atributos* o *creación de características* (también visto en el Capítulo 4), tarea en la que el objetivo es construir de forma automática variables nuevas aplicando operaciones a las variables originales. En este problema el espacio de búsqueda se incrementa puesto que el número de combinaciones posibles de operaciones y operadores es muy elevado. Esto hace que sea un problema adecuado para la programación genética [Hu 1998]. También se ha abordado este problema desde una perspectiva híbrida de algoritmos genéticos y programación genética que realiza de forma simultánea selección y construcción de atributos [Vafaie & De Jong 1998].

Una propuesta evolutiva que afronte de forma conjunta la selección de atributos e instancias puede codificar cada solución candidata en un cromosoma binario de longitud $m+n$ (siendo m el número de variables y n el de instancias) [Liu & Motoda 2001]. El algoritmo de reducción de la dimensionalidad puede diseñarse con una función de adaptación envolvente, filtro o mixta de igual forma que los algoritmos de selección, extracción o construcción de atributos.

Algoritmos genéticos básicos para la selección de variables

En esta sección describiremos dos ejemplos de algoritmos genéticos cuyo objetivo es la selección de un conjunto de atributos que permitan reducir la dimensionalidad de los datos, manteniendo y/o mejorando su capacidad predictiva. Las dos propuestas difieren únicamente en el esquema de representación. Las describiremos a través de sus componentes:

- *Representación.* El objetivo del *algoritmo genético 1 (AG1)* es seleccionar un conjunto de variables de cardinalidad fija; así, si el algoritmo genético busca un conjunto de k variables, cada cromosoma representa un conjunto candidato mediante una lista de k enteros donde el i -ésimo gen representa la i -ésima variable seleccionada. En la mayoría de los problemas no se conoce a priori el número de variables a seleccionar. El *algoritmo genético 2 (AG2)* codifica un conjunto de atributos de longitud variable mediante una lista de N bits; así, si en el i -ésimo bit hay un 1, indica que la i -ésima variable forma parte de la solución candidata.
- *Operadores genéticos.* Ambos algoritmos genéticos utilizan el esquema de reproducción de estado estacionario, el cruce en dos puntos y la mutación uniforme.

- *Función de adaptación.* Las dos propuestas son algoritmos genéticos de selección de características tipo envolvente y utilizan como medida de evaluación de un subconjunto de variables una estimación de la capacidad de predicción de dichas variables. Esta medida se calcula utilizando *leaving one out* y la regla del vecino más cercano (1-NN). Sin embargo, el AG2 añade una penalización inversamente proporcional al porcentaje de reducción alcanzada.

En la Tabla 15.2 se muestran los resultados obtenidos con el conjunto de datos *Wisconsin*, visto anteriormente (que recordemos tenía 30 variables originariamente) con ambos algoritmos. Para el AG1 se han propuesto cinco tamaños distintos para el conjunto de variables: 2, 3, 5 y 7. Para AG2 la primera fila de la tabla muestra los resultados obtenidos sin penalización en la función de adaptación y en la segunda los correspondientes a la ejecución con un peso para la penalización de 0,3 que orienta el proceso evolutivo de búsqueda hacia subconjuntos pequeños.

Algoritmo	Nº Variables	%	Variables
AG1	2	91,916	21, 22
	3	92,794	2, 11, 20
	5	93,673	1, 2, 12, 21, 22
	7	93,849	1, 2, 11, 12, 20, 21, 22
	10	93,849	0, 1, 2, 6, 11, 12, 19, 21, 22, 25
AG2	19	93,849	1, 2, 4, 6, 7, 8, 10, 11, 12, 14, 18, 19, 21, 22, 25, 26, 27, 28, 29
	3	93,497	2, 21, 22

Tabla 15.2. Resultados obtenidos con dos algoritmos genéticos básicos para selección de características

Como se puede observar ambos algoritmos genéticos permiten seleccionar de forma sencilla conjuntos de variables con capacidad predictiva alta. El AG2 tiene la ventaja de no necesitar que el usuario determine a priori un tamaño fijo. No obstante, la limitación de la búsqueda a un determinado tamaño fuerza a una mayor reducción y es adecuada para algunos problemas.

Los procesos de selección de características se utilizan, como se ha mencionado, en la etapa de pre-procesamiento, antes del proceso de minería. En la Tabla 15.3 se muestran los resultados obtenidos al ejecutar el algoritmo genético de extracción de reglas (descrito en la página 395) a partir del conjunto de 19 variables obtenidos por los algoritmos genéticos de selección de características presentados en esta sección.

Nº Interv.	Comp.	Conf.	Regla
5	84,874	95,886	SI (0 <= Mean Concave Points < 0,042) Y (0 <= Perimeter SE < 5) ENTONCES Benigno
	40,566	98,851	SI (134 <= Worst Perimeter < 176) ENTONCES Maligno
7	75,630	95,070	SI (0 <= Mean Concavity < 0,061) ENTONCES Benigno
	37,264	98,750	SI (0,171 <= Worst Concave Points < 0,214) ENTONCES Maligno

Tabla 15.3. Reglas intervalares obtenidas a partir de 19 variables seleccionadas previamente.

En la Tabla 15.4 se muestran los resultados con el conjunto de cinco variables.

Nº Interv.	Comp.	Conf.	Regla
5	65,266	98,312	SI (0 <= Perimeter SE < 5) Y (50 <= Worst Perimeter < 92) ENTONCES Benigno
	49,057	92,857	SI (100 <= Mean Perimeter < 130) ENTONCES Maligno
7	54,902	95,610	SI (161,428 <= Mean Perimeter < 82,857) Y (0 <= Perimeter SE < 3.571) ENTONCES Benigno
	25,472	100,000	SI (125,714 <= Mean Perimeter < 147,142) ENTONCES Maligno

Tabla 15.4. Reglas intervalares obtenidas a partir de cinco variables seleccionadas previamente.

El proceso de extracción de reglas que parte de un conjunto de variables reducido es más eficiente, pero se puede observar en la Tabla 15.4 que las reglas obtenidas tienen niveles de confianza y completitud algo inferiores a los obtenidos con la generación de reglas partiendo del conjunto completo de variables. Esto se debe a que el algoritmo genético de extracción de reglas, por su funcionamiento, selecciona las variables más adecuadas para incluirlas en el antecedente de la regla y no requiere un proceso de selección de características previo.

15.4 Lógica difusa

La lógica difusa (del inglés, *fuzzy logic*⁵³) permite modelar conocimiento impreciso y cuantitativo así como transmitir, manejar incertidumbre y soportar, en una extensión razonable, el razonamiento humano de una forma natural. Desde que Zadeh la propuso en 1965 [Zadeh 1965] se ha aplicado en múltiples áreas de investigación, fundamentalmente por su cercanía al razonamiento humano y por proporcionar una forma efectiva para capturar la naturaleza aproximada e inexacta del mundo real.

Los sistemas basados en reglas que utilizan conjuntos difusos para describir los valores de sus variables se denominan sistemas basados en reglas difusas. Una regla difusa puede tener la siguiente descripción:

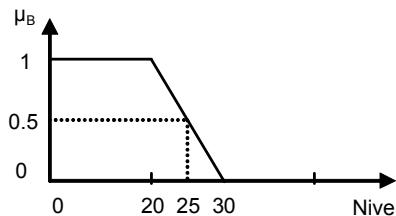
SI el radio es *pequeño* y el nivel de simetría es *alto*
ENTONCES el nivel de concavidad es *muy bajo*

donde los términos lingüísticos *pequeño*, *alto* o *muy bajo* se caracterizan mediante un conjunto difuso.

Un conjunto difuso puede definirse como una generalización de los conjuntos clásicos. En éstos, la función de pertenencia sólo puede tomar dos valores (0 o 1, pertenencia o no pertenencia). En cambio, en los conjuntos difusos, la función de pertenencia asigna a cada elemento un grado de pertenencia dentro del intervalo [0,1]. Esto permite representar conceptos con límites borrosos, mal definidos, pero su significado sí está definido de forma completa y precisa. En [Klir & Yuan 1995] se puede encontrar una descripción detallada sobre Teoría de Conjuntos Difusos. En la Figura 15.10 se puede ver un ejemplo de conjunto difuso definido con una función de pertenencia trapezoidal.

Un conjunto difuso es una entidad más compleja que un conjunto clásico pero constituye una representación con mayor precisión para conceptos reales. Esta expresividad permite simplificar reglas y los sistemas basados en ellos.

⁵³ Una traducción también muy frecuente es la de “lógica borrosa”.



$$\mu_B(x) = \begin{cases} 1, & 0 \leq x \leq 20 \\ \frac{30-x}{10}, & 20 < x \leq 30 \\ 0, & x > 30 \end{cases}$$

Figura 15.10. Ejemplo de conjunto difuso.

Una partición difusa de una variable determina una distinción de niveles en los valores de la misma mediante conjuntos difusos y permite solapamiento en las fronteras, al igual que ocurre en el razonamiento humano cuando trabajamos con valores lingüísticos [Zadeh 1975]. En la Figura 15.11, se muestra de forma gráfica la diferencia entre la división del dominio de una variable en intervalos y una partición difusa para la misma. Como se puede observar, la variable nivel se describe mediante cinco términos lingüísticos de los que se utilizan habitualmente en el razonamiento humano con variables numéricas: *Muy Bajo*, *Bajo*, *Medio*, *Alto* y *Muy Alto*. Cuando utilizamos estos conceptos, mentalmente pensamos en ellos con un cierto solapamiento entre algunos términos, ya que existen determinados valores de la variable nivel que se pueden considerar hasta cierto punto *Alto* o *Muy Alto*. De ahí que, cuando decidimos trabajar con una variable considerándola una variable *lingüística* que toma como valores términos *lingüísticos*, definimos una partición difusa que considere siempre una división del dominio en términos lingüísticos con cierto nivel de solapamiento. El significado de cada término viene especificado por un conjunto difuso y por tanto por una función de pertenencia. Esta función de pertenencia determinará –de forma precisa– para cualquier valor de la variable, el grado de pertenencia al conjunto difuso correspondiente.

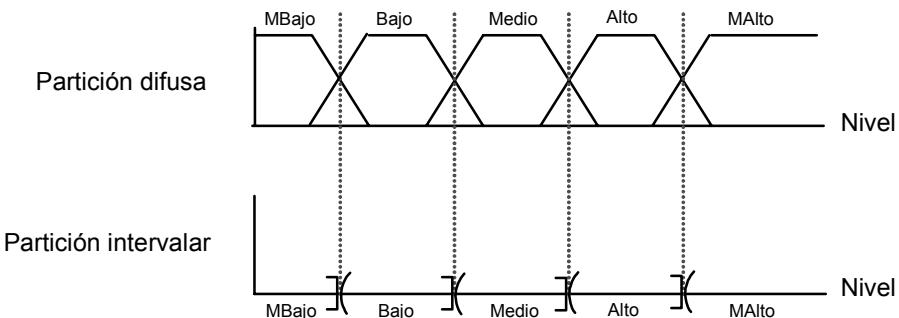


Figura 15.11. Partición difusa e intervalar de una variable.

Los sistemas basados en reglas difusas están formados por un conjunto de reglas difusas como la mostrada anteriormente. Ante una situación dada, la regla se podrá aplicar si el antecedente de la misma describe la zona del espacio a la que pertenece el ejemplo, es decir, si el grado de compatibilidad del antecedente de la regla y el ejemplo es mayor que cero. Este grado se calcula de la siguiente forma:

- Para cada variable se calcula el grado de pertenencia al conjunto difuso correspondiente. Si en la regla, para una variable se indica una disyunción de términos (Nivel = *Bajo* O *Muy Bajo*), esto se interpreta como la unión de los conjuntos difusos correspondientes y el grado de pertenencia se calcula con una t-conorma (operador

de unión difusa) que habitualmente es el máximo, aunque existen otras definiciones para el mismo. Según esto, el grado de pertenencia de un valor actual de la variable nivel a “*Bajo O Muy Bajo*” será igual al máximo entre el grado de pertenencia del valor actual del nivel a *Bajo* y el grado de pertenencia a *Muy Bajo*.

- Habitualmente el antecedente de una regla difusa está formado por una conjunción de condiciones para las distintas variables. En este caso, el grado de compatibilidad del ejemplo con la regla se calcula con una t-norma (operador de intersección difusa). Existen múltiples expresiones posibles para las t-normas, pero es frecuente el uso del operador mínimo o producto. Si utilizamos el mínimo, el grado de compatibilidad se calcula como el mínimo entre los grados de pertenencia de las variables implicadas en el antecedente a los conjuntos difusos correspondientes. Como se puede observar, si se elige la t-norma mínimo o producto, cuando una de las variables no pertenece al conjunto difuso implicado, el grado de compatibilidad del ejemplo con la regla es cero.
- Si en el antecedente las condiciones para las variables se combinan con el operador de disyunción (por ejemplo Nivel = *Bajo O Edad = Joven*), se utilizará el operador de unión difusa (t-conorma) para el cálculo del grado de compatibilidad.

Las reglas difusas se pueden considerar modelos locales simples, lingüísticamente interpretables y con un rango de aplicación muy amplio. Permiten la incorporación de toda la información disponible en el modelado de sistemas, tanto de la que proviene de expertos humanos que expresan su conocimiento sobre el sistema en lenguaje natural, como de la que tiene su origen en medidas empíricas y modelos matemáticos.

Es necesario destacar que no todos los sistemas difusos son sistemas basados en reglas difusas. Los conjuntos difusos se utilizan también, por ejemplo, en algoritmos de agrupamiento con el objetivo de obtener conjuntos difusos que permitan diferenciar los grupos con mayor expresividad, o también en algoritmos de modelado para predicción o regresión. Por ejemplo, se puede obtener un modelo que utilice conjuntos difusos para representar el conocimiento extraído, pero no necesariamente mediante reglas difusas.

En minería de datos, uno de los aspectos que determinan la calidad del conocimiento extraído, y por tanto del algoritmo utilizado, es la interpretabilidad del mismo y en este sentido los sistemas basados en reglas difusas son los más utilizados dentro del campo de la lógica difusa. En la siguiente sección se describen algunos de ellos.

15.5 Lógica difusa en minería de datos

En esta sección describiremos brevemente las razones que han potenciado el uso de lógica difusa en minería de datos y algunas de sus aplicaciones. Terminaremos la sección mostrando un ejemplo de proceso de generación de reglas difusas.

15.5.1 Aplicaciones de la lógica difusa en minería de datos

El uso de la lógica difusa en minería de datos viene determinado por los siguientes motivos:

- Cualquier proceso de minería de datos tiene como objetivo principal la identificación de patrones interesantes y la descripción de los mismos de una forma concisa

y con significado. Los modelos difusos representan una descripción de los datos orientada al usuario a través de un conjunto de reglas cualitativas que establecen relaciones significativas y útiles entre variables. Los conjuntos difusos permiten establecer límites *flexibles* entre los distintos niveles de significado, sin ignorar ni enfatizar en exceso los elementos cercanos a las fronteras, de igual forma que ocurre en la percepción humana. En todo proceso de extracción de conocimiento hay un componente de interacción humana y los conjuntos difusos permiten representar el conocimiento de forma lingüística, incorporar conocimiento previo de forma fácil y proporcionar soluciones interpretables.

- En muchos algoritmos de minería de datos y, especialmente, en la evaluación de los patrones extraídos, aparece el concepto de *interés* [Freitas 1999], que agrupa distintas características como validez, novedad, utilidad y simplicidad, y que puede cuantificarse de manera muy conveniente a través de conjuntos difusos. De forma adicional, algunos algoritmos incorporan, como parámetro adicional de interés, una medida de diferenciación difusa entre un patrón descubierto y un vocabulario definido por el usuario.

El papel de la lógica difusa en minería de datos se puede agrupar en función del tipo de tarea que realice el algoritmo de extracción de conocimiento:

1. *Agrupamiento*. En este campo, los algoritmos de minería de datos intentan diferenciar entre grandes volúmenes de datos mostrando información útil en forma de nuevos segmentos o grupos (*clusters*). Los conjuntos difusos proporcionan una búsqueda a través de los datos orientada y especificada en términos lingüísticos que ayuda a descubrir dependencias entre los datos en un formato cualitativo o semi-cualitativo [Turksen 1998]. Además, el uso de conjuntos difusos permite incluir fácilmente información contextual e incluso orientar el proceso de búsqueda con información lingüística [Pedrycz 1996].
2. *Clasificación*. De modo similar, los conjuntos difusos son muy apropiados para flexibilizar algunos algoritmos de clasificación clásica, utilizando regiones más flexibles para árboles de decisión, redes neuronales, máquinas de vectores soporte, etc. Por ejemplo, [Janikow 1998] presenta árboles de decisión donde las particiones se realizan a través de funciones de pertenencia difusas y no clásicas como ocurre en los árboles de decisión tradicionales. Esto permite que un ejemplo caiga por *varias* ramas del árbol, ponderándose las predicciones de las hojas de cada rama y dando una estimación de la clase más robusta.
3. *Reglas de asociación*. La afinidad de la lógica difusa con la forma en que se representa el conocimiento humano hace que su uso en las reglas de asociación con variables cuantitativas facilite la legibilidad de las mismas, su diseño, la incorporación de conocimiento cualitativo sobre el problema, el tratamiento de valores perdidos o faltantes y de clases con límites no muy bien definidos, así como el procesamiento del ruido en las variables que son resultado de medidas reales [Au & Chan 1998]. En [Chen & Wei 2002] se propone un algoritmo de extracción de reglas de asociación difusas con modificadores lingüísticos que hacen que el conocimiento extraído sea aún más comprensible y cercano al usuario, enriqueciendo la semántica de las reglas de asociación y haciendo que éstas sean más granulares. En algunas propuestas, se introduce conocimiento sobre el problema en la definición inicial de los

conjuntos difusos [Au & Chan 1999] y otras proponen un algoritmo de agrupamiento para establecer la definición de los mismos [Fu et al. 1998].

4. *Dependencias funcionales.* Dentro de este campo se ha utilizado la inferencia difusa para generalizar tanto la inferencia imprecisa como la precisa basada en dependencias funcionales entre variables. De forma similar las bases de datos relacionales difusas generalizan a las clásicas permitiendo el almacenamiento y recuperación de información difusa [Hale & Shenoi 1996]. En la bibliografía especializada existen distintas propuestas para extraer dependencias funcionales extendidas representadas mediante reglas difusas con variables lingüísticas [Bosc et al. 1999].
5. *Sumarización de datos.* En esta área el objetivo es proporcionar al usuario información comprensible para captar la esencia de una gran cantidad de información en una base de datos. La lógica difusa en esta tarea constituye una herramienta para representar conceptos difusos, jerarquías difusas [Lee & Kim 1997] y además facilita la interacción del usuario en sumarios lingüísticos y conceptos que involucran combinaciones complicadas de atributos.

15.5.2 Algoritmo para la generación de reglas difusas de Wang y Mendel

A continuación presentamos un ejemplo sencillo de utilización de la lógica difusa en un proceso de extracción de reglas de clasificación o asociación. Es una extensión del algoritmo iterativo propuesto por Wang y Mendel [Wang & Mendel 1992] que sigue el siguiente esquema:

1. Dividir el dominio de cada variable numérica en conjuntos difusos definiendo la función de pertenencia para cada uno de ellos. Esta partición difusa se puede realizar con información del experto (es decir, a partir de los niveles de distinción que él utiliza para razonar sobre el problema) o con conjuntos difusos distribuidos uniformemente a lo largo del dominio, en el caso de que no se disponga de esta información.
2. Para cada ejemplo, construir una regla de acuerdo a lo siguiente: elegir, para cada una de las variables el término lingüístico para el cual el valor correspondiente del ejemplo tiene mayor grado de pertenencia al conjunto difuso correspondiente. El antecedente de la regla estará formado por la conjunción de dichos términos lingüísticos. Asignar como consecuente de la regla la clase a la que pertenece el ejemplo. Si la regla no se ha generado previamente, incluirla en el conjunto de reglas.

Como se puede observar es un proceso iterativo que genera, para cada ejemplo, la regla que mejor lo describe. Por la forma en que genera las reglas y el tipo de reglas que obtiene es más apropiado para problemas en los que sea necesario extraer información poco general, específica.

Hemos aplicado este algoritmo de extracción de reglas difusas de nuevo al problema *Wisconsin Diagnostic Breast Cancer* (*wdbc*). En la Tabla 15.5 se muestran los resultados obtenidos utilizando cinco y siete términos lingüísticos (NTL en la tabla) para cada una de las 30 variables. Se muestra el número de reglas extraídas (NR) y como ejemplo, el nivel de confianza y completitud de las mejores reglas para cada clase.

NTL	NR	Mejor Regla	Comp.	Conf.
5	550	R1 (Clase 0)	75,070	96,585
		R2 (Clase 1)	36,321	93,500
5	569	R3 (Clase 0)	27,451	100
		R4 (Clase 1)	10,377	100

Tabla 15.5. Resultados obtenidos por el método de generación de reglas difusas de Wang y Mendel con 30 variables.

Como ejemplo, la expresión completa de la regla R1 es la siguiente:

Si *radius=Bajo* Y *texture=Bajo* Y *perimeter=Bajo* Y *area=Bajo* Y *smoothness=Medio* Y *compactness=Bajo* Y *concavity=Bajo* Y *concave_points=Bajo* Y *symmetry=Medio* Y *fractal_dimension=Medio* Y *radius SE=Muy bajo* Y *texture SE=Bajo* Y *perimeter SE=Muy bajo* Y *area SE=Muy bajo* Y *smoothness SE=Bajo* Y *compactness SE=Bajo* Y *concavity SE=Muy bajo* Y *concave_points SE=Bajo* Y *symmetry SE=Bajo* Y *fractal dimension SE=Bajo* Y *worst radius=Bajo* Y *worst texture=Bajo* Y *worst perimeter=Bajo* Y *worst area=Muy bajo* Y *worst smoothness=Medio* Y *worst compactness=Bajo* Y *worst concavity=Bajo* Y *worst concave points=Bajo* Y *worst symmetry=Medio* Y *worst fractal dimension=Bajo*
ENTONCES Benigno

Como se puede observar son reglas complejas y poco generales puesto que obligatoriamente incluyen todas las variables. Por tanto, son reglas mucho menos generales que las obtenidas por el algoritmo genético de extracción de reglas visto en la Tabla 15.1, aunque hay que destacar que son más generales que las que se obtendrían con un algoritmo de generación de reglas intervalares que forzase la aparición de todas las variables en el antecedente.

Con el objetivo de mejorar la inteligibilidad y generalidad de las reglas extraídas con el método de Wang y Mendel, se podría diseñar un proceso de generación de reglas en dos etapas:

1. Etapa de selección de variables relevantes. Para ello utilizamos los resultados obtenidos con el algoritmo de la página 401. En concreto, y por mostrar dos ejemplos, utilizaremos el conjunto de 19 variables resultado del algoritmo AG2 (el conjunto de variables con mayor poder de predicción) y el de cinco variables obtenido por el algoritmo AG1.
2. Etapa de generación de reglas difusas con las variables seleccionadas.

En la Tabla 15.6 se muestran los resultados obtenidos con los dos conjuntos de variables y cinco y siete términos lingüísticos.

NV	NTL	NR	Mejor Regla	Comp.	Conf.
19	5	501	R1 (Clase 0)	77,311	96,228
			R2 (Clase 1)	48,113	93,068
	7	562	R3 (Clase 0)	30,532	100
			R4 (Clase 1)	13,679	100
5	5	92	R5 (Clase 0)	88,515	91,158
			R6 (Clase 1)	78,302	96,248
	7	156	R7 (Clase 0)	59,384	95,470
			R8 (Clase 1)	38,679	87,490

Tabla 15.6. Resultados con el método de generación de reglas difusas de Wang y Mendel con 19 y 5 variables.

Las reglas obtenidas son más sencillas, especialmente las que se han generado con el conjunto de cinco variables. En la Tabla 15.7 se muestran las mejores reglas obtenidas con el conjunto de cinco variables.

R5	SI texture= <i>Bajo</i> Y perimeter= <i>Bajo</i> Y perimeter SE= <i>Muy Bajo</i> Y worst texture= <i>Bajo</i> Y worst perimeter= <i>Bajo</i> ENTONCES benigno
R6	SI texture= <i>Medio</i> Y perimeter= <i>Medio</i> Y perimeter SE= <i>Bajo</i> Y worst texture= <i>Medio</i> Y worst perimeter= <i>Medio</i> ENTONCES maligno
R7	SI texture= <i>Medio</i> Y perimeter= <i>Muy Bajo</i> Y perimeter SE= <i>Muy Bajo</i> Y worst texture= <i>Medio</i> Y worst perimeter= <i>Muy Bajo</i> ENTONCES benigno
R8	SI texture= <i>Medio</i> Y perimeter= <i>Alto</i> Y perimeter SE= <i>Muy Bajo</i> Y worst texture= <i>Alto</i> Y worst perimeter= <i>Medio</i> ENTONCES maligno

Tabla 15.7. Mejores reglas obtenidas con el método de Wang y Mendel con cinco variables.

El nivel de completitud de las mejores reglas obtenidas es siempre superior al que se alcanza con reglas que incluyen todas las variables manteniéndose la confianza a un nivel adecuado. Para este tipo de algoritmos de extracción de reglas que utilizan todas las variables tiene especial relevancia la reducción de la dimensionalidad previa. Si comparamos las características de las mejores reglas difusas obtenidas con 19 y 5 variables con las reglas intervalares obtenidas por el algoritmo genético (página 402, Tabla 15.3 y Tabla 15.4) vemos que por su propia definición las reglas difusas tienden a ser más generales que las intervalares.

15.6 Sistemas evolutivos difusos en minería de datos

Como hemos visto en la sección anterior, la lógica difusa es una herramienta que permite representar conceptos de forma expresiva y fácilmente entendible para el usuario, es decir, de forma lingüística. Esta herramienta se sitúa, por tanto, en el nivel de representación y se puede utilizar en combinación con otras técnicas y en esta sección estudiaremos la combinación de la lógica difusa con algoritmos evolutivos.

Los procesos de descubrimiento evolutivo de reglas de asociación difusas se pueden agrupar en torno a tres enfoques:

- Algoritmos evolutivos que sólo evolucionan las reglas difusas, y utilizan funciones de pertenencia definidas por el usuario. Este enfoque permite incorporar el conocimiento del dominio del usuario en la especificación de las funciones de pertenencia.
- Algoritmos evolutivos que evolucionan sólo aspectos relacionados con las funciones de pertenencia, como el número de funciones de pertenencia para cada atributo, la forma de estas funciones de pertenencia, etc.
- Algoritmos evolutivos que evolucionan tanto las reglas difusas como las funciones de pertenencia. Esto se puede realizar de dos formas distintas: utilizando individuos que incluyan los aspectos de las funciones de pertenencia junto con los valores de los atributos, o individuos distintos para los valores de los atributos y para los parámetros de las funciones de pertenencia, lo que implicaría la evolución paralela de ambas poblaciones de individuos.

Las características de las propuestas incluidas en el primer enfoque son similares a las descritas en la Sección 15.3, ya que no se evolucionan específicamente los aspectos

distintivos de las reglas difusas, los conjuntos difusos. En las siguientes subsecciones describiremos propuestas evolutivas haciendo especial hincapié en las que evolucionan los conjuntos difusos. En [Cordón et al. 2001] se puede encontrar una revisión exhaustiva de los sistemas evolutivos difusos encuadrados en los distintos enfoques.

15.6.1 Algoritmos genéticos para el descubrimiento de reglas de asociación difusas

Nos centraremos en varios aspectos importantes relacionados con la utilización de algoritmos genéticos para el descubrimiento de reglas de asociación difusas: el esquema de representación de los individuos, los operadores genéticos y la función de adaptación.

Esquema de representación

Para la representación de los individuos, si partimos de una partición difusa predefinida por el usuario se pueden utilizar los esquemas de codificación para reglas no difusas descritos en la Sección 15.3 de este capítulo. Si se aprenden también las funciones de pertenencia, el esquema de codificación debe incluir su representación. En este caso, podemos optar por incluir las características de las funciones de pertenencia en la codificación de los individuos, o codificarlas de forma independiente en otro individuo que evolucione de forma separada.

En [Mendes et al. 2001] se detalla una forma de codificar las funciones de pertenencia utilizando un individuo separado dividido en K elementos, donde K es el número de atributos que se tratarán como variables lingüísticas y de los cuales se van a aprender las funciones de pertenencia. Cada elemento consta de cuatro genes que definen colectivamente tres funciones de pertenencia (bajo, medio y alto) para el atributo correspondiente (véase la Figura 15.12). Cada gen se utiliza para especificar las coordenadas de dos vértices del trapecio que pertenecen a un par de funciones de pertenencia adyacentes.

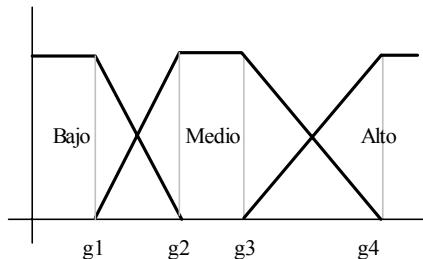


Figura 15.12. Definición de tres funciones de pertenencia trapezoidales mediante cuatro genes (g_1, g_2, g_3, g_4).

Esta representación tiene distintas ventajas: reduce el espacio de búsqueda del algoritmo genético, ahorra tiempo de procesamiento, fuerza a cierto solapamiento entre funciones de pertenencia adyacentes y garantiza que, para cada valor original del atributo continuo, la suma de sus grados de pertenencia a los tres valores lingüísticos sea 1.

Operadores genéticos

Para la tarea de descubrimiento de reglas de asociación difusas, se pueden utilizar operadores genéticos clásicos además de los descritos en la Sección 15.3, adaptándolos a las características de la codificación utilizada para la representación de los individuos.

Función de adaptación

A la hora de evaluar las reglas difusas obtenidas, se deben tener en cuenta aspectos como la precisión predictiva, la comprensibilidad y el interés. Para medir la precisión predictiva de una regla difusa debemos calcular el grado de pertenencia de cada ejemplo al antecedente de la regla utilizando la t-norma y t-conormas elegidas para representar los operadores de intersección y unión difusas. Una vez calculado el grado de pertenencia del ejemplo, la precisión predictiva del individuo (regla difusa) se puede calcular como [Romao et al. 2002]:

$$\text{Precisión} = \frac{\text{PredCorr} - \frac{1}{2}}{\text{TotPred}}$$

donde *PredCorr* (número de predicciones correctas) es la suma de los grados de pertenencia del antecedente de la regla para todos los ejemplos que tienen el valor predicho por la regla, y *TotPred* (número total de predicciones) es la suma de los grados de pertenencia al antecedente para todos los ejemplos.

Esta fórmula es una versión difusa de la medida de precisión utilizada por algunos algoritmos de minería de datos [Noda et al. 1999]. El objetivo al restar $\frac{1}{2}$ a *PredCorr* en el numerador es penalizar las reglas que son demasiado específicas sin tener una influencia significativa en reglas más generales.

La precisión predictiva de una regla difusa también se puede medir calculando el factor de confianza, *FC*, a partir de las medidas *PV* (Positivos Verdaderos, suma del grado de pertenencia al antecedente de los ejemplos pertenecientes a la clase indicada en el consecuente), *PF* (Positivos Falsos, suma del grado de pertenencia al antecedente de los ejemplos pertenecientes a otra clase distinta a la indicada en el consecuente), *NF* (Negativos Falsos, suma del grado de pertenencia de los ejemplos a la zona del espacio no delimitada por el antecedente, pero que satisfacen el consecuente) y *NV* (Negativos Verdaderos, suma del grado de pertenencia a la zona no delimitada por el antecedente de los ejemplos no pertenecientes a la clase indicada en el consecuente). El factor de confianza se calcula con la siguiente expresión:

$$FC = \frac{PV}{PV + PF}$$

También se puede extender la función de adaptación con una medida de comprensibilidad, que mida la simplicidad de la regla. Algunas funciones de adaptación que tienen en cuenta tanto la precisión predictiva como la comprensibilidad de una regla se describen en [Janikow 1993; Giordana & Neri 1995; Flockhart & Radcliffe 1995].

Otro criterio a considerar es el interés. En [Noda et al. 1999] se propone una función de adaptación que tiene en cuenta de forma objetiva el grado de interés de una regla mediante una suma ponderada cuyos pesos son especificados por el usuario. Una forma de considerar las medidas de interés subjetivas en la función de adaptación se puede encontrar en [Romao et al. 2002], donde se considera interesante una regla cuando es sorprendente para el usuario, en el sentido de representar conocimiento que no sólo es desconocido para el usuario, sino que también contradice las creencias originales del usuario.

15.6.2 Programación genética para el descubrimiento de reglas de asociación difusas

Las diferencias en las propuestas de programación genética para extracción de reglas de asociación difusas respecto a las realizadas para reglas no difusas residen fundamentalmente en la forma de representar los individuos.

Para generación de reglas de asociación difusas en la programación genética, las funciones serán operadores difusos y los nodos terminales serán términos lingüísticos [Mendes et al. 2001].

15.7 Ejemplos

En este apartado describiremos dos propuestas evolutivas para la extracción de reglas de asociación difusas. Las reglas a obtener se consideran reglas de asociación porque el objetivo de los algoritmos de minería de datos presentados no es tanto obtener un conjunto de reglas completo que maximice la precisión global en la predicción sino obtener un conjunto (posiblemente parcial) de reglas que maximicen la comprensibilidad e interés del conocimiento extraído [Freitas 2000].

15.7.1 Algoritmo genético iterativo con codificación entera: reglas simples

La primera propuesta diseñada para resolver el problema es un algoritmo genético iterativo de extracción de reglas de asociación difusas que intenta optimizar la precisión, generalidad e interés de las reglas de asociación difusas [Aguilera et al. 2003]. El algoritmo incluye una medida de interés objetiva e incorpora conocimiento del usuario y del dominio en la definición previa del conjunto de términos lingüísticos para las variables continuas.

El algoritmo genético tiene como objetivo descubrir reglas de asociación para un atributo objetivo prefijado, por lo que tendrá que ejecutarse tantas veces como valores distintos tenga el atributo objetivo y así obtener un conjunto de reglas para cada uno de ellos. La población inicial del algoritmo se genera de forma aleatoria. Se incorpora una etapa de pos-procesamiento que optimiza cada regla obtenida con el objetivo de incrementar la completitud, manteniendo el grado de confianza de la misma.

```

ALGORITMO Genético-Iterativo(E)
  Elegir un atributo objetivo AtOBJ
  Cto_Reglas ← Ø
  REPETIR
    Ejecutar el AG(E, AtOBJ) obteniendo la regla R
    BúsquedaLocal(R)
    CtoReglas ← CtoReglas ∪ R
    Modificar el conjunto de ejemplos E (penalizar los usados por R)
  MIENTRAS confianza(R) ≥ confianza_min Y R describa ejemplos nuevos
  FIN ALGORITMO

```

Figura 15.13. Algoritmo genético iterativo de reglas simples.

El proceso de minería de datos se realiza con un algoritmo iterativo que permite la obtención de varias reglas (invocando al algoritmo genético, AG) mientras las reglas generadas alcancen un nivel mínimo de confianza y describan información sobre zonas del espacio de búsqueda en las que aún queden ejemplos no cubiertos por las reglas generadas en las iteraciones anteriores. Para ello, se penalizan (una vez obtenida una regla) el conjunto de ejemplos representados por la misma para la generación de futuras reglas. El esquema completo del algoritmo se ve en la Figura 15.13.

A continuación se describen con más detalle los elementos que conforman esta propuesta.

Esquema de representación

Cada cromosoma codifica sólo el antecedente de la regla ya que el consecuente es fijo durante cada ejecución del algoritmo genético. El antecedente de la regla se representa en un cromosoma de longitud fija, utilizando codificación entera. Para ello se añade al conjunto de valores válidos de cualquier variable un valor especial indicador de la ausencia de dicha variable en la regla. Como vimos en la Figura 15.6 (pág. 392), hemos de considerar que, en este caso, el valor entero del gen i indica que la variable i toma como valor el término lingüístico i cuyo significado viene dado por la función de pertenencia correspondiente. Esta función puede ser definida previamente por el usuario o definida de forma uniforme, como se muestra en la Figura 15.14, en caso de que no se disponga de este conocimiento experto.

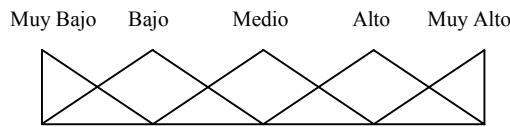


Figura 15.14. Ejemplo de partición difusa para una variable continua.

Función de adaptación

La función de evaluación combina, según la siguiente expresión, tres factores: la confianza, la completitud y el grado de interés de la regla.

$$fitness(c) = \frac{\omega_1 \cdot Completitud(c) + \omega_2 \cdot Interés(c) + \omega_3 \cdot Confianza(c)}{\omega_1 + \omega_2 + \omega_3}$$

1. *Confianza*. Determina la precisión de la misma ya que refleja el grado con el que los ejemplos pertenecientes a la zona del espacio delimitado por el antecedente verifican la información indicada en el consecuente de la regla. Para el cálculo de este factor utilizamos una expresión modificada de la definición de precisión aportada por Quinlan en [Quinlan 1987b] que se utiliza frecuentemente en la generación de reglas difusas de clasificación [Cordón et al. 1998]: la suma del grado de pertenencia de los ejemplos de la clase a la zona determinada por el antecedente dividido entre la suma del grado de pertenencia de todos los ejemplos (independientemente de la clase a la que pertenezcan) a la misma zona. Para calcular estos grados de pertenencia se utilizan funciones de pertenencia triangulares y la t-norma mínimo.
2. *Completitud*. Es una medida del grado de cobertura que la regla ofrece a los ejemplos de la clase. Se calcula como el cociente entre el número de nuevos ejemplos de la clase que cubre la regla y el número de ejemplos de la clase que quedaban por

cubrir. Esta forma de medir la completitud toma sentido al utilizarse el algoritmo genético dentro de un proceso iterativo, con el objetivo de potenciar la obtención de reglas distintas en cada ejecución del algoritmo genético. A partir de la segunda iteración se penalizan aquellas reglas que cubren ejemplos pertenecientes a las zonas delimitadas por reglas obtenidas previamente. Es una penalización que no utiliza ninguna función de distancia ya que penaliza diferencias a nivel fenotípico.

3. *Interés.* Para el cálculo del interés, en la práctica es adecuado utilizar los criterios objetivos como medidas que permitan seleccionar reglas potencialmente interesantes y los criterios subjetivos para que el usuario final determine reglas realmente interesantes [Freitas 2002]. En nuestra propuesta se sigue este enfoque y en el algoritmo genético el grado de interés se evalúa de forma objetiva. Para ello utilizamos el criterio de interés proporcionado en [Noda et al. 1999], que considera (en un proceso de modelado de dependencias) que el nivel de interés de una regla viene determinado por dos términos, uno referido al antecedente y otro al consecuente. Como el consecuente está prefijado, para el cálculo de interés sólo se utiliza el componente del antecedente, basado en una medida de información dada por la expresión:

$$\text{Interés} = 1 - \left(\frac{\sum_{i=1}^n \text{Gain}(A_i)}{n \cdot \log_2(|\text{dom}(G_k)|)} \right)$$

donde Gain es la ganancia de información [Noda et al. 1999], n es el número de variables que aparecen en el antecedente de la regla y $|\text{dom}(G_k)|$ es la cardinalidad de la variable objetivo (el número de valores posibles para la variable considerada como clase). Las variables con un valor alto de ganancia de información son adecuadas para predecir una clase, cuando estas variables se consideran de forma individual. Pero, desde el punto de vista del interés de una regla, se entiende que el usuario ya conoce o ha averiguado cuáles son las variables más predictivas para un dominio de aplicación concreto y por tanto las reglas que contienen dichas variables son menos interesantes (por ser menos sorpresivas y aportar menos información). Por eso se entiende que el antecedente de una regla es más interesante si contiene atributos con poca cantidad de información.

El objetivo global de la función de evaluación es orientar la búsqueda hacia reglas que maximicen la precisión y la medida de interés, minimizando el número de ejemplos negativos y no cubiertos.

Operadores genéticos

El algoritmo genético utiliza un modelo de reproducción de estado estacionario [Bäck et al. 1997]. La recombinación se realiza a través del operador de cruce en dos puntos y un operador de mutación uniforme pero sesgado, con el que la mitad de las mutaciones realizadas tienen el efecto de eliminar la variable correspondiente.

Etapa de pos-procesamiento del algoritmo genético: algoritmo de búsqueda local

La etapa de pos-procesamiento, que mejora la regla obtenida mediante un proceso de ascensión de colinas (*hill-climbing*), generaliza la regla mientras se incremente el grado de completitud. Para ello, en cada iteración se determina la variable tal que, al eliminarla, aumenta en mayor grado la completitud de la regla resultante, obteniendo así reglas más generales. Finalmente, la regla optimizada sustituirá a la original sólo si supera la confianza mínima.

15.7.2 Algoritmo genético iterativo con codificación binaria: reglas DNF

Este algoritmo es una ampliación de la propuesta anterior en la que se cambia el esquema de codificación para representar reglas de tipo DNF (forma normal disyuntiva), es decir, se incorpora la posibilidad de disyunción de valores para una variable. Para ello se utiliza un esquema de codificación binaria con tantos genes por variable como valores posibles existan para la misma. Se mantiene el esquema iterativo y la etapa de pos-procesamiento que optimiza cada regla obtenida con el objetivo de incrementar la completitud manteniendo el grado de confianza de la misma.

Esquema de representación

El esquema de representación adoptado es el que vimos en la Figura 15.7 (pág. 393) considerando que, en este caso, un 1 en un valor de una determinada variable no implica un intervalo sino un término lingüístico. Si para una variable hay más de un término lingüístico, se considera que la condición para esa variable está formada por la unión difusa de los conjuntos difusos asociados. De esta forma, se almacena para cada variable una lista de enteros (tantos como valores posibles puede tomar la variable) que pueden tomar el valor 0 o 1. En la última posición se guarda (a efectos de eficiencia) si la variable interviene o no en la regla (no interviene si están todos los valores o no está ninguno).

Función de adaptación

La función de adaptación tiene la misma expresión que la utilizada en el algoritmo genético con codificación entera visto anteriormente, salvo que para reglas difusas DNF no vamos a considerar el criterio de interés.

15.7.3 Experimentación

Para entender mejor los algoritmos anteriores y ver su capacidad, se han implementado y se ha realizado de nuevo una experimentación sobre la base de ejemplos *Wisconsin Diagnostic Breast Cancer* (véase Apéndice B). Los parámetros del algoritmo se muestran en la Tabla 15.8.

Los únicos parámetros que cambian de un algoritmo a otro son los pesos asociados a los objetivos de la función adaptación que toman los valores 0,5, 0,4 y 0,1 para la completitud, confianza e interés respectivamente en el caso del algoritmo genético con codificación entera y 0,4, 0,6 y 0 para el algoritmo genético con codificación binaria. La

razón de esta diferencia es que, en el segundo algoritmo, que evoluciona reglas DNF, se aumenta el peso de la confianza porque el tipo de representación tiende a generar reglas más generales y es conveniente potenciar la confianza de las mismas.

Número de evaluaciones	5000
Tamaño de la población	100
Umbral mínimo de confianza	0,9

Tabla 15.8. Valores para los parámetros de los algoritmos genéticos propuestos.

Se han ejecutado cinco veces ambos algoritmos para cada una de las clases del atributo predictor (0: Benigno, 1: Maligno), y los resultados se muestran en la Tabla 15.9 y la Tabla 15.10.

Clase	Regla	Comp	Conf	Interés
0	1	97,759	95,945	0,887
	2	87,395	91,704	0,869
1	1	76,415	99,275	0,909
	2	37,736	100	0,903
	3	77,830	99,298	0,896
	4	11,321	100	0,928
	5	6,604	100	0,849

Tabla 15.9. Resultados obtenidos con el algoritmo genético con codificación entera.

Como se puede observar el primer algoritmo genético (el de codificación entera) genera conjuntos de reglas que cubren casi la totalidad de los ejemplos con un nivel de confianza por encima de 0,9. Además la medida de interés objetiva determina un nivel de calidad adecuado.

Clase	Regla	Comp	Conf
0	1	99,440	85,472
1	1	100	67,211

Tabla 15.10. Resultados obtenidos con el algoritmo genético con codificación binaria.

Las reglas correspondientes a la Tabla 15.9 son las siguientes:

Clase 0 (Benigno):
 SI (*conc points SE* = Bajo) Y (*worst perimeter* = Bajo) Y (*worst conc points* = Bajo)
 ENTONCES benigno
 SI (*mean concave point* = Bajo) Y (*worst texture* = Bajo) Y (*worst perimeter* = Bajo)
 ENTONCES benigno

Clase 1 (Maligno)
 SI (*mean concave points* = Bajo) Y (*mean fractal dimension* = Bajo) Y
 (*worst concave points* = Alto)
 ENTONCES maligno
 SI (*worst radius* = Alto)
 ENTONCES maligno
 SI (*fractal dimension SE* = Bajo) Y (*worst concave points* = Alto)
 ENTONCES maligno
 SI (*worst compactness* = Alto) Y (*worst symmetry* = Alto)
 ENTONCES maligno
 SI (*worst perimeter* = Alto) Y (*worst compactness* = Alto)
 ENTONCES maligno

Si observamos las reglas obtenidas para cada una de las clases vemos que son conjuntos de reglas comprensibles, tanto por el reducido número de reglas como de condiciones en el antecedente de las mismas.

Las reglas correspondientes a la Tabla 15.10 son:

```

Clase 0 (Benigno):
  SI ((w.perimeter = Muy bajo) 0 (w.perimeter = Bajo) 0 (w.perimeter = Muy Alto) ) Y
    ( (worst compactness = Bajo) 0 (worst compactness = Muy Alto) ) Y
    ( (worst symmetry = Bajo) 0 (worst symmetry = Medio))
  ENTONCES benigno

Clase 1 (Maligno):
  SI (mean concavity = Bajo) Y
    ( (mean fractal dimension = Bajo) 0 (mean fractal dimension = Muy Alto) )
    ( (worst symmetry = Bajo) 0 (worst symmetry = Medio) )
  ENTONCES maligno

```

El algoritmo genético con codificación binaria extrae conjuntos formados por una sola regla muy general (con un elevado nivel de completitud) y un nivel de confianza más bajo, pero adecuado. El hecho de extraer una regla que representa a todos los ejemplos, hace que el proceso iterativo pare y no se generen más reglas.

15.8 Sistemas software

Las herramientas evolutivas y de lógica difusa todavía no se suelen integrar en muchas *suites* de minería de datos. Para utilizar estas herramientas es necesario, por tanto, utilizar sistemas específicos. Generalmente las herramientas vienen con una configuración óptima y permiten la variación de algunos parámetros. Si se desea ir más allá y cambiar la codificación de las reglas o variar los operadores en muchos casos es necesario tocar el código fuente.

A continuación se detallan implementaciones de algoritmos evolutivos y lógica difusa, tanto genéricas como algunas específicas en el campo de la minería de datos. Para los algoritmos evolutivos, tenemos:

- *Genetic Algorithms Archive* (<http://www.aic.nrl.navy.mil/galist/>) contiene información relacionada con la investigación en computación evolutiva. Desde esta página se puede acceder a contenidos teóricos o al código fuente libre de distintas implementaciones de algoritmos evolutivos.
- Herramientas para el MATLAB: existen dos paquetes, el EA_Matlab (http://www.systemtechnik.tu-ilmenau.de/~pohlheim/EA_Matlab/ea_matlab.html), un conjunto de herramientas para implementar un amplio número de métodos de la computación evolutiva en MATLAB, y el *Genetic Algorithm Toolbox for MATLAB* (<http://www.shef.ac.uk/uni/projects/gaipp/gatbx.html>) desarrollado en la Universidad de Sheffield para el uso de algoritmos genéticos dentro de la herramienta MATLAB. Estas herramientas son genéricas y no específicas para minería de datos.
- *Genalytics Predictive Suite* (<http://www.genalytics.com/>) es una herramienta comercial mucho más orientada a la minería de datos. Es un paquete integrado, muy orientado al uso de técnicas evolutivas en CRM (*Customer Relationship Management*).
- *Yale* (<http://yale.cs.uni-dortmund.de>) es una *suite* de minería de datos que incorpora algoritmos genéticos.

En cuanto a la lógica difusa:

- Entorno Xfuzzy (http://www.imse.cnm.es/Xfuzzy/index_sp.html): desarrollado en el Instituto de Microelectrónica de Sevilla, perteneciente al Centro Nacional de Microelectrónica. Xfuzzy es un entorno libre de programación de sistemas basado en lógica difusa, que combina un conjunto de herramientas que facilitan las distintas etapas del proceso de diseño de sistemas de inferencia basados en lógica difusa, desde su descripción inicial hasta la implementación final. También contiene un libro sobre lógica difusa (<http://www.imse.cnm.es/Xfuzzy/Fleb/Fleb.htm>).
- Winrosa (<http://esr.e-technik.uni-dortmund.de/winrosa/en/winrs.htm>) es una herramienta *software* para la generación automática de reglas difusas basada en el método *Fuzzy-ROSA*, desarrollado en la Univ. de Dortmund (Alemania).
- DataEngine (<http://www.dataengine.de/>) es una herramienta de minería de datos desarrollada por la empresa alemana MIT GmbH, para análisis inteligente de datos que une métodos estadísticos con redes neuronales y lógica difusa.

Algunas técnicas irán incorporándose en sistemas de minería de datos genéricos. No obstante, también es conveniente consultar sitios sobre el área de algoritmos evolutivos o lógica difusa, como por ejemplo la página principal de EvoNet (*the European Network of Excellence in Evolutionary Computing*), (<http://evonet.inria.fr/>), que contiene información sobre computación evolutiva, o la página Fuzs (<http://www.abo.fi/~rfuller/fuzs.html>) de Robert Fullér, con enlaces interesantes sobre lógica difusa.

15.9 Conclusiones

El área de la computación flexible proporciona un conjunto de herramientas que de forma independiente o conjunta se han comenzado a utilizar con éxito en tareas de extracción de conocimiento.

Además de las redes neuronales, que se tratan en el Capítulo 13, y de las técnicas consideradas en este capítulo, existen otras técnicas relacionadas (o incluidas) en la computación flexible que no se han tratado: métodos estocásticos no evolutivos, como el algoritmo de metrópolis o el *simulated annealing* [Kirkpatrick et al. 1987]. Éstos, en cierta manera, se consideran precursores de los métodos evolutivos y su uso en minería de datos es limitado hoy en día.

La lógica difusa permite al usuario incorporar conocimiento lingüístico al proceso de minería de datos, de forma directa, mezclarlo con información no lingüística y tratar de forma adecuada datos incompletos y/o con ruido. Pero quizás una de las características que más potencia su uso en este tipo de algoritmos es su capacidad de representar conocimiento de forma lingüística y directamente interpretable, a través de las reglas difusas.

La programación genética utiliza un esquema de representación con gran poder expresivo, por lo que se aplica de forma natural para descubrir reglas de predicción. Además tiene capacidad para extraer reglas sorpresivas e interesantes para el usuario por su habilidad para descubrir interacciones importantes entre variables; es muy posible que aunque un atributo por sí solo no sea relevante para la predicción, cuando se combina con otros mediante alguna función, sí lo sea. Hay que destacar que en ocasiones se señala que una desventaja del uso de la programación genética en descubrimiento de reglas es su

tendencia a producir resultados con capacidad de generalización baja. No obstante, tal y como se pone de manifiesto en la bibliografía especializada, la programación genética para minería de datos tiene un gran potencial de uso aún por abordar.

Los algoritmos genéticos realizan una búsqueda global e independiente del dominio, aspecto que los convierte en una herramienta robusta, escalable y aplicable en distintas etapas del proceso de extracción de conocimiento. Un argumento adicional que justifica su escalabilidad es que su funcionamiento facilita las implementaciones paralelas. Hay que destacar además que una parte importante de los procesos de extracción de conocimiento tienen un enfoque multiobjetivo, puesto que tratan de obtener soluciones cuya calidad depende de más de un criterio (por ejemplo, reglas que sean precisas, con alto nivel de interés y simplicidad) y los algoritmos multiobjetivo [Deb 2001] proporcionan soluciones adecuadas a este tipo de problemas. Una de las principales dificultades de las propuestas genéticas en la actualidad es el proceso de evaluación de individuos, aspecto en el que se continúa investigando, especialmente en técnicas de muestreo sobre grandes bases de datos que reduzcan el conjunto de aprendizaje sin pérdida de precisión predictiva.

Por último, aunque aquí hemos tratado hibridaciones entre la lógica difusa y la computación evolutiva, se ha trabajado extensamente también en otras hibridaciones, como los sistemas “*neuro-fuzzy*”, “*neuro-genetic*”, “*fuzzy-svm*”, etc.

Capítulo 16

MÉTODOS BASADOS EN

CASOS Y EN VECINDAD

Pedro Isasi

Este capítulo se centrará en métodos de aprendizaje automático basados en casos y vecindad. Este tipo de métodos trata de resolver un problema a partir de información extraída de un conjunto de ejemplos existentes previamente. Los ejemplos son los que aportarán la información necesaria para poder predecir el comportamiento de un nuevo dato no perteneciente al conjunto de ejemplos. Los métodos basados en vecindad reciben su nombre del hecho de que la predicción se basa fundamentalmente en la utilización del conjunto de ejemplos “vecinos” al dato que hay que procesar, o en el caso más general, porque la *distancia* entre cada ejemplo y el dato en cuestión es esencial en el proceso.

El capítulo se va a dividir en métodos para la tarea de clasificación, en la que los ejemplos tienen información de la clase a la que pertenecen, y métodos para la tarea de agrupamiento, en la que los ejemplos no contienen información de la clase a la que pertenecen (aprendizaje no supervisado). El capítulo incluye técnicas diversas: vecinos más próximos, discriminantes de Fisher, agrupamiento jerárquico, mapas auto-organizativos de Kohonen para agrupamiento y clasificación (SOM y LVQ), K medias, así como métodos evolutivos basados en vecindad más avanzados y una breve reseña al razonamiento basado en casos. La característica conductora de todas estas técnicas es que se basan en los conceptos de distancia o en conceptos derivados de ella, como es el de cercanía o vecindad.

Dentro del problema supervisado, se va a abordar, a modo de ejemplo, la tarea de clasificación, pero todo lo comentado es fácilmente extrapolable a otros problemas supervisados como son la regresión, la interpolación o la aproximación de funciones.

16.1 Introducción

Cuando se aprende de ejemplos, casos o datos conocidos, generalmente lo que se intenta es poder tomar una decisión sobre nuevos casos. Parece de sentido común pensar que ante

una nueva situación se deberá actuar como se hizo en situaciones anteriores *parecidas o similares*, si en éstas se tuvo éxito. En consecuencia, parece una regla bastante clara y natural que puede ser aplicada a multitud de tareas de minería de datos.

Por ejemplo, en la tarea de clasificación, podremos asignar una clase a un nuevo caso, observando las clases de casos similares. Igualmente, en la tarea de agrupamiento, asignaremos un nuevo ejemplo al grupo donde estén los individuos más similares. En el caso de la regresión, el valor predicho para un nuevo ejemplo no puede distar mucho de los valores obtenidos para ejemplos similares.

A partir de esta idea surgen dos nociones muy importantes. La primera, y la más obvia, es determinar qué se entiende por “similitud”, dando lugar al concepto matemático equivalente (o más bien, inverso) de “distancia”. La segunda es determinar cuándo se va a explotar dicha similitud: si se va a realizar con un preprocessamiento anticipativo o no retardado (*eager*) respecto a un preprocessamiento demorado o retardado (*lazy*). Esta cuestión surge porque los métodos que buscan instancias similares se ajustan muy bien a esta segunda perspectiva, a la de esperar hasta que se nos plantea una cuestión sobre un nuevo ejemplo. En ese momento, se buscará de entre los ejemplos anteriores que se encuentran almacenados, el más similar, y se actuará como se actuó en aquella ocasión. Esto supone, en muchos casos, la no creación de un modelo general para la tarea en cuestión, sino que se predice *al vuelo*, en función de cada caso concreto. Esto hace que muchos de estos métodos reciban nombres diferentes según la perspectiva: aprendizaje basado en instancias [Aha et al. 1991] o en casos, métodos retardados o perezosos (*lazy*), métodos basados en similitud o en distancias.

Los métodos que se describen en este capítulo deben procesar el conjunto de ejemplos existentes para poder hacer comparaciones con los nuevos casos y los casos pasados. De hecho, existen tres maneras de procesar los ejemplos: la memorización de todos los ejemplos (dando lugar a métodos retardados), la memorización de parte (seleccionando ejemplos significativos, siendo, en cierto modo, un híbrido) y la creación de *prototipos*, es decir, representantes ficticios de un conjunto de datos (dando lugar a métodos no retardados).

A continuación se describe en más detalle las dos nociones anteriores: el concepto de distancia como función inversa de la similitud y la separación entre métodos retardados y no retardados.

16.1.1 Medidas de distancia

Como se ha mencionado, la manera de formalizar el concepto de similitud es a través de métricas o medidas de distancia. En realidad, si se quiere saber la similitud entre dos instancias o individuos, es necesario elegir una función de distancia y calcular con ella la distancia entre los dos individuos. Y éste es precisamente uno de los aspectos más interesantes de los métodos que se describen en este capítulo: el mismo método puede funcionar de maneras diferentes, variando la métrica de distancia. Dicho de otra manera, se pueden adaptar distintos métodos de aprendizaje que tienen una función de distancia como parámetro para trabajar con distintos tipos de problemas.

Una pregunta lógica ante esta situación es la siguiente: ¿cuántas funciones de distancia hay? Como se verá a continuación, además de la distancia euclídea o clásica, existen otras

muchas funciones que cumplen los requisitos de una función de distancia (llamada entonces métrica) y que pueden funcionar mejor en ciertos contextos. Más aún, dependiendo de la aplicación, se pueden definir funciones *ad-hoc* que actúen como métrica de distancia.

Las medidas de distancias más tradicionales son aquellas que se aplican sobre dos instancias o ejemplos, tales que todos los atributos son numéricos. Por ejemplo, se pueden definir las siguientes distancias entre dos vectores, puntos o instancias x e y de dimensión n de muy distintas formas.

- **Distancia Euclídea.** Es la distancia clásica, como la longitud de la recta que une dos puntos en el espacio euclídeo:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- **Distancia de Manhattan** o distancia por cuadras (*city-block*). Como su nombre indica, hace referencia a recorrer un camino no en diagonal (por el camino más corto), sino zigzagueando, como se haría en Manhattan:

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- **Distancia de Chebychev.** Simplemente calcula la discrepancia más grande en alguna de las dimensiones:

$$d(x, y) = \max_{i=1..n} |x_i - y_i|$$

- **Distancia del coseno.** Si se considera que cada ejemplo es un vector, la distancia sería el coseno del ángulo que forman:

$$d(x, y) = \arccos\left(\frac{x^T y}{\|x\| \cdot \|y\|}\right)$$

- **Distancia de Mahalanobis.** Todas las distancias anteriores asumen, en cierto modo, que los atributos son independientes (es decir, consideran cada atributo una dimensión ortogonal a las demás). Una distancia más robusta es ésta, que utiliza la matriz de covarianzas S :

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

Existen otros casos donde cada ejemplo o instancia está formado también por atributos numéricos, pero representan una secuencia. Por ejemplo, si una instancia tiene doce atributos, $x = (x_1, x_2, \dots, x_{12})$, donde cada x_i indica las ventas en el mes i , puede ser más interesante tener en cuenta si la secuencia de variables tiene o no tiene la misma forma. Existen distancias apropiadas para secuencias, como son la distancia de Spearman (véase por ejemplo [Gibbons 1985]), o incluso algunos coeficientes de correlación.

En la mayoría de las métricas anteriores, desde la euclídea hasta la de Spearman, es muy conveniente normalizar todos los atributos (entre cero y uno por ejemplo), como vimos en la Sección 4.5 (página 93). Si no normalizamos, alguna dimensión (atributo) puede tener una magnitud media muchísimo mayor que otras y pesará mucho más a la hora de calcular las distancias. La detección de valores anómalos o atípicos es también

importante, ya que una buena normalización puede verse frustrada por estos valores, como se comentó también en el Capítulo 4.

El concepto de distancia no sólo existe para valores numéricos, también se puede utilizar cuando los ejemplos están formados por atributos nominales, cosa que es habitual en minería de datos. Por ejemplo, para atributos nominales se suele utilizar la función delta, es decir $\delta(a,b)=0$ si y sólo si $a=b$ y $\delta(a,b)=1$ en caso contrario. Con esta función, podemos definir la distancia de manera sencilla:

$$d(x, y) = \omega \sum_{i=1}^n \delta(x_i, y_i)$$

donde ω es simplemente un factor de reducción (por ejemplo $1/n$ es un valor usual). Este factor se puede elegir convenientemente cuando los atributos son nominales y numéricos, se puede utilizar esta función de distancia para el subconjunto de atributos nominales, utilizar una función de distancia de las anteriores para el subconjunto de atributos numéricos y combinar ambos valores utilizando un factor adecuado.

Del mismo modo se pueden definir distancias para otros tipos de datos más complejos. Por ejemplo, cuando se trabaja con tiras de caracteres, una distancia muy habitual es la *distancia de edición*, en la que se ponderan inserciones, borrados y sustituciones. Por esta razón la palabra “jamón” está más cerca de “amor” que de “monja”.

Si los datos son conjuntos más que vectores, entonces se puede definir la distancia entre conjuntos de la siguiente manera:

$$d(x, y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$$

En este caso, por ejemplo, el conjunto $\{ j, a, m, o, n \}$ está más cerca de $\{ m, o, n, j, a \}$, en realidad la distancia sería 0, que de $\{ a, m, o, r \}$.

Finalmente, se pueden definir distancias específicas para documentos de texto o hipertexto, grafos, árboles y cualquier otra estructura de datos que represente los ejemplos.

La relevancia de todas estas medidas de distancia es que la mayoría de métodos que se describen en este capítulo se pueden “parametrizar” para trabajar con distintos tipos de datos y actuando de diferentes formas, con sólo cambiar la función de distancia que utiliza el algoritmo.

16.1.2 Métodos retardados frente a no retardados

Como se ha comentado, existe una división de métodos que tiene que ver con la manera de realizar el aprendizaje. Es lo que se conoce como métodos retardados y no retardados.

Los métodos retardados, también llamados perezosos (del inglés *lazy*), se llaman así porque retrasan la decisión de la generalización del conjunto de entrenamiento, hasta el instante en que se recibe un nuevo dato a procesar. Por el contrario, los métodos anticipativos o no retardados (del inglés *eager*) construyen un modelo de generalización, antes de tener que realizar dicha tarea de generalización, a partir del conjunto de ejemplos. Una vez que el modelo ha sido realizado, los ejemplos son reemplazados por el modelo, y nunca se vuelven a utilizar (se pueden liberar de la memoria), a la hora de realizar la generalización.

Las diferencias entre ambos tipos de métodos son fundamentales [Mitchell 1997]:

- Los métodos retardados necesitan menos esfuerzo de computación durante el entrenamiento; en la mayoría de los casos la fase de entrenamiento ni siquiera existe, mientras que los métodos no retardados pueden requerir de un gran esfuerzo de computación para desarrollar el modelo en la fase de entrenamiento.
- Por el contrario, los métodos retardados desplazan el grueso de su tarea a la fase de generalización, y es en esta fase donde requieren un esfuerzo computacional muy elevado. Los métodos no retardados suelen realizar la generalización en muy poco tiempo y con poco esfuerzo computacional.
- Los métodos retardados esperan hasta la aparición de un nuevo dato a generalizar para construir su modelo de generalización, y el modelo construido es sólo válido para el dato en cuestión.
- Los métodos no retardados construyen su modelo sin tener en cuenta el dato a generalizar, lo construyen en una fase previa, de forma global, y el mismo modelo es utilizado para todos los datos futuros a generalizar.

Estas diferencias se pueden agrupar en una diferencia fundamental, y es que los métodos retardados realizan una aproximación local al dato a generalizar, por lo tanto el modelo resultante podría verse como una combinación de aproximaciones locales, que da mayor versatilidad a la solución. Por otra parte los métodos no retardados construyen una aproximación global utilizando la totalidad del conjunto de ejemplos. El problema de los métodos retardados es que se basan en la selección de un conjunto de ejemplos apropiados, o en algún tipo de ponderación de cada uno de los elementos del conjunto de ejemplos, para la construcción de la aproximación local. Tanto los criterios de selección de ejemplos apropiados, como la ponderación de los mismos, puede ser una labor difícil y en algunos casos distorsionar, al no haber criterios claros, los resultados de las aproximaciones locales, y por extensión de la generalización producida.

A continuación se describirán brevemente, a modo de ejemplo, un método no retardado, los discriminantes lineales de Fisher; y otro retardado, el método de k vecinos, que será explicado en más detalle más adelante.

Ejemplo: discriminantes lineales

Los discriminantes lineales consisten en dividir el espacio de estados, es decir el conjunto de todos los posibles puntos en los que pueden ser ubicados los ejemplos en regiones, definidas mediante ecuaciones lineales. Las funciones discriminantes (paramétricas y no paramétricas) se vieron en los capítulos 7 y 8. En este punto sólo se mostrará el más simple de los discriminantes, el lineal, para ilustrar un método no retardado y para ilustrar que, en cierto modo, es un discriminante que se basa en distancias, en este caso la euclídea. En un discriminante lineal sobre un espacio n-dimensional, las fronteras de las regiones vendrán determinadas por las intersecciones de los hiperplanos descritos en las ecuaciones que se obtienen como solución al problema. Estas regiones han de tener la particularidad de contener ejemplos únicamente de una clase.

El discriminante lineal más conocido y más utilizado por su sencillez y rapidez es el discriminante de Fisher [Fisher 1936]. Este procedimiento consiste en dividir el espacio en hiperplanos, como se ha dicho anteriormente. Se parte de un conjunto de datos, que se

representarán en el espacio como puntos, de los cuales se conoce la clase a la que pertenecen. Cada conjunto de puntos pertenecientes a la misma clase forma un grupo. Se determina en primer lugar el centroide de cada grupo, a partir de los centroides se determina un nuevo hiperplano de forma que divida el espacio en dos partes biseccionando la línea que une los centros de cada clase.

Para ilustrar estas ideas se incluirá un ejemplo sobre diferentes variedades de la flor del lirio (“iris”, este conjunto de datos se explica en el Apéndice B). Existen tres variedades del lirio: Setosa (etiquetada con una *S*), Versicolor (etiquetada con una *E*) y Virginica (etiquetada por una *A*), que pueden distinguirse por el ancho y longitud de sus pétalos (aquí, por simplicidad, no se considerarán el resto de los atributos). Además, supóngase que se dispone del conjunto de muestras ya contrastadas (etiquetadas) que aparece en la Figura 16.1.

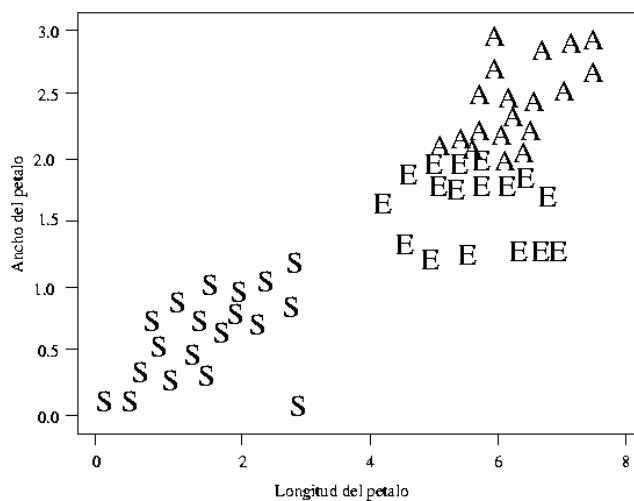


Figura 16.1. Conjunto de datos para clasificación.

Se desea obtener una regla que discrimine, en función de la longitud y el ancho, las flores futuras. En este caso, al haber sólo dos características discriminantes, el espacio de estados es bidimensional, y se puede representar en un plano como se aprecia en la figura; al ser un espacio bidimensional las superficies discriminantes serán hiperplanos de dimensión 1, es decir rectas. La existencia de un mayor número de dimensiones no varía la manera de resolver el problema.

El discriminante de Fisher calcula los centroides (el punto que minimice la distancia media de los elementos de esa clase) de cada una de las clases *S*, *E* y *A*, a continuación, se unen los centros de las clases adyacentes con sendas rectas, y se calcula la mediatrix de los elementos anteriores. Estas mediatrixes serán las reglas discriminantes obtenidas por el método de Fisher (Figura 16.2), definidas por las ecuaciones correspondientes a dichas rectas. Si se varía la función de distancia utilizada para calcular los centroides o la función de distancia utilizada para calcular las mediatrixes se podría tener una clasificación diferente.

Como se aprecia en la Figura 16.2, el método construye una aproximación global al problema mediante la división del espacio en tres regiones delimitadas por rectas,

utilizando el conjunto de ejemplos. Las *rectas* son el modelo y éste ya puede ser utilizado tantas veces como haga falta.

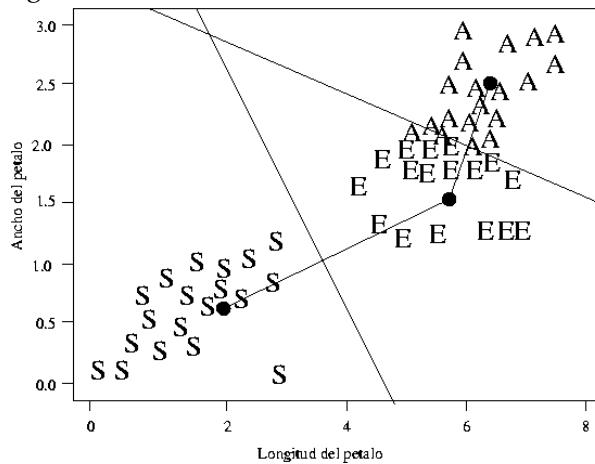


Figura 16.2. Clasificación mediante discriminantes de Fisher.

Una vez creado el modelo de generalización, los ejemplos se pueden despreciar, no van a volver a ser utilizados, y los nuevos datos serán predichos utilizando el modelo; en este caso un nuevo dato se etiquetará como Virginica si está dentro de la primera región, como Versicolor si está en la segunda y como Setosa si está en la tercera.

Ejemplo: K vecinos

En este caso se determina para cada región del espacio la probabilidad de que un elemento que esté situado en ella pertenezca a cada una de las clases existentes. Este tipo de clasificadores será objeto de estudio en la Sección 16.3.2. En este caso no hay reglas prefijadas como en los casos anteriores, sino que la clasificación se irá haciendo para cada caso nuevo en particular. Cuando un caso nuevo aparece, se genera un círculo con centro en dicho punto y un radio prefijado como parámetro del sistema. Se calcula el número de ejemplos que caen dentro del círculo y se etiqueta al nuevo caso como perteneciente a la clase más numerosa dentro del círculo. En la Figura 16.3 se aprecia cómo en un primer caso, con un radio reducido, dentro del círculo hay dos ejemplos de la clase A y uno de la clase E, luego el nuevo dato se etiqueta como perteneciente a la clase A.

En la figura se aprecia también que el radio elegido influye en la clasificación. Si se aumenta o disminuye el radio, el número de ejemplos que caen dentro del círculo puede variar, y la predicción realizada también. Por lo tanto, este radio será un parámetro crítico a tener en cuenta. También podemos observar que el concepto de círculo viene (o puede venir) determinado por el concepto de distancia. Si se varía la función de distancia se pueden generar "bolas", que son un área cerrada donde la distancia es menor o igual que un determinado valor. Variando la función de distancia, por tanto, también podrá variar la clasificación realizada por el método.

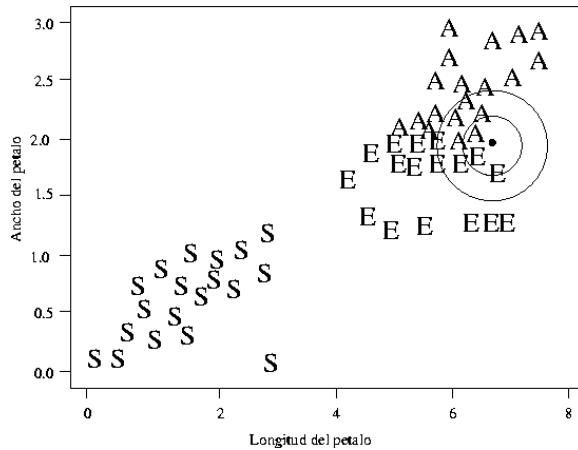


Figura 16.3. Clasificación mediante vecino más cercano.

El método espera hasta la aparición de un nuevo dato a clasificar para la utilización del conjunto de ejemplos. Cuando el dato está disponible se recurre a los ejemplos para realizar la clasificación, se crea una regla local al dato que acaba de llegar, se realiza la clasificación, y se abandona dicha regla. Si ahora hubiera que clasificar otro dato más, los cálculos realizados para la clasificación del dato anterior serían inservibles y habría que realizar de nuevo todo el proceso. Este es un típico ejemplo de aprendizaje retardado. En este caso hay que almacenar los ejemplos de entrenamiento siempre, ya que son utilizados una y otra vez, no como en el ejemplo anterior, en el que una vez construido el modelo los ejemplos pueden ser eliminados.

16.2 Técnicas para agrupamiento

El agrupamiento (*clustering*) es, como se ha visto a lo largo de este libro, una de las tareas más frecuentes en la minería de datos. Se trata de encontrar grupos entre un conjunto de individuos. El concepto de distancia puede jugar un papel crucial, ya que individuos similares (cercanos) deberían ir a parar al mismo grupo.

Muchos de los métodos de agrupamiento que se describirán a continuación (sobre todo los jerárquicos y los K medias) se suelen incluir en el área que se conoce en estadística como análisis de conglomerados (*cluster analysis*), aunque hoy en día este uso es más restringido, debido a la hibridación de técnicas.

16.2.1 Mapas auto-organizativos de Kohonen

El modelo de mapas autoorganizados de Kohonen fue realizado por un científico finlandés llamado Teuvo Kohonen [Kohonen 1982; Kohonen 1984]. Fue modelizado en un principio como una red neuronal de dos capas, tal y como se muestra en la Figura 16.4 y tal como se vio en el Capítulo 13.

El modelo consta de dos capas, una capa de entrada, donde se introducen los ejemplos, y otra de competición, en el que cada célula representa a un *prototipo*. Como se vio, el objetivo del entrenamiento es que los prototipos *capturasen* ejemplos similares.

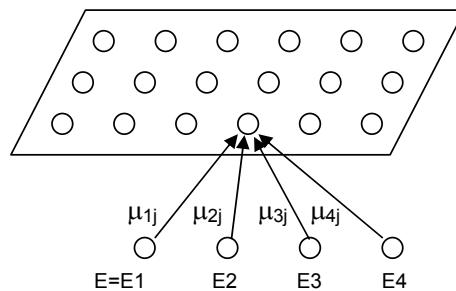


Figura 16.4. Arquitectura del mapa de Kohonen.

Sin embargo el método puede ser descrito como un sistema de aprendizaje a partir de ejemplos basado en distancia. En este caso, cada ejemplo de entrenamiento viene definido por un vector de n componentes que puede representarse mediante un punto en un espacio de n dimensiones. Si n es igual a 2, cada ejemplo sería un punto en un plano.

Un aspecto muy interesante de este tipo de redes es que los prototipos pueden ser también representados como puntos en el plano, gracias a los valores de sus conexiones entre la capa de entrada y la de competición (de hecho tienen tantas conexiones como entradas, con lo que esta coincidencia ocurre para cualquier dimensionalidad de los ejemplos). En la parte izquierda de la Figura 16.5 se muestra una red de Kohonen con dos entradas y tres salidas (prototipos) C_1 , C_2 y C_3 . En la parte derecha de la Figura 16.5 se representan los ejemplos y los prototipos. Éstos hacen, en cierto modo, la función de "centroides".

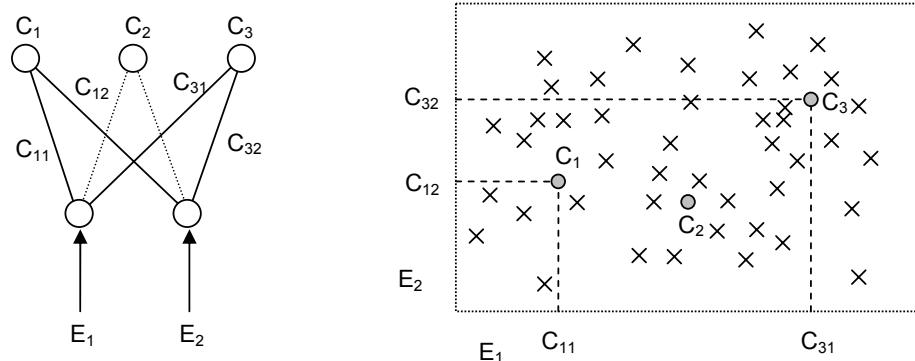


Figura 16.5. Representación de prototipos y ejemplos.

El método se divide en dos fases. En primer lugar, se selecciona a un prototipo como ganador de una competición entre todos ellos, y después el prototipo ganador es desplazado para que represente mejor a los puntos que le pertenecen. En la primera fase se calcula una distancia entre los prototipos y un punto del conjunto de ejemplos. Para el cálculo de este valor se pueden utilizar varias medidas de distancia de las descritas con anterioridad. La distancia euclídea es una de las más empleadas:

$$d(\varepsilon, \mu_j) = \sqrt{\sum_{i=1}^n d(\varepsilon_i, \mu_{ij})^2}$$

donde ε es el atributo i -ésimo de la entrada y μ_{ij} es el componente i -ésimo del prototipo j . La función producto escalar también se usa frecuentemente como distancia:

$$d(\varepsilon, \mu_j) = \sum_{i=1}^n \varepsilon_i \cdot \mu_{ij}$$

En este caso la distancia mide el ángulo que forman los dos vectores a comparar. Si dicho ángulo es pequeño, la distancia producida será también pequeña, aunque los puntos estén muy distantes entre sí. Esta es la distancia que mejor se adapta a un modelo de red de neuronas, ya que es equivalente a multiplicar la entrada por los pesos (en este caso se estaría midiendo la distancia entre entradas y pesos). Además, en el Capítulo 13 (página 329) se vio cómo utilizar la función “sombrero mejicano” como distancia.

Mediante la distancia elegida se calcula cuál de los prototipos está más cerca, y se le etiqueta como ganador:

$$\mu_g = \arg \min_{\mu_j} \{d(\varepsilon, \mu_j)\}$$

Por lo tanto, μ_g será el prototipo ganador del ejemplo ε . Cada ejemplo tendrá un prototipo ganador asociado, o lo que es lo mismo, cada prototipo representará a un subconjunto de ejemplos.

El procedimiento general sería:

- Seleccionar un ejemplo de los del conjunto de entrenamiento.
- Calcular la distancia de dicho ejemplo a cada una de las células (prototipos) del mapa de Kohonen.
- Etiquetar como ganadora a aquella célula (prototipo) cuya distancia sea menor.

El aprendizaje se realiza para cada ejemplo. Los ejemplos se introducen uno a uno, para cada uno se calcula su prototipo ganador, y se desplaza acercándolo, según la medida de distancia elegida, al ejemplo. Si se utiliza la distancia euclídea, la posición de los prototipos se realiza siguiendo la expresión:

$$\frac{d_{\mu_j}}{dt} = \alpha(t) \cdot \tau_j(t) \cdot (\varepsilon_i(t) - \mu_{ij}(t))$$

donde:

$$\tau_i = \begin{cases} 1 & \text{si } C_i \text{ ganadora} \\ 0 & \text{en caso contrario} \end{cases}$$

Las dos ecuaciones anteriores podrían sustituirse por:

$$\frac{d_{\mu_j}}{dt} = \begin{cases} \alpha(t) \cdot (\varepsilon_i(t) - \mu_{ij}(t)) & \text{si } C_i \text{ ganadora} \\ 0 & \text{en caso contrario} \end{cases}$$

En la Figura 16.6 se muestra cómo se lleva a cabo dicha aproximación.

El parámetro α es lo que se denomina la tasa de aprendizaje. Tiene el efecto de regular la potencia del desplazamiento, como se vio en otras aproximaciones de aprendizaje neuronal en el Capítulo 13. Valores grandes de α hacen que los prototipos se acerquen mucho a los ejemplos, y valores bajos que los prototipos apenas se desplacen. Es conveniente que en los primeros momentos del aprendizaje los prototipos se desplacen en gran medida, ya que, inicialmente, son colocados de manera aleatoria en el espacio, y un desplazamiento rápido a la zona cercana a los puntos que representan, permite un mejor posicionamiento final.

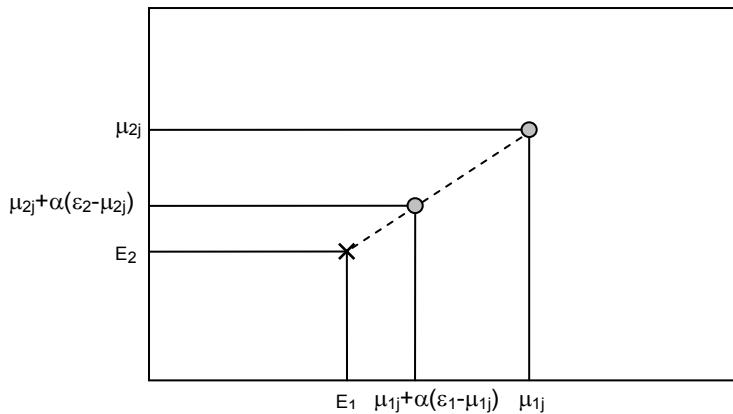


Figura 16.6. Aproximación del prototipo ganador al ejemplo.

A medida que el aprendizaje avanza, y debido a que los ejemplos se introducen repetidamente, es conveniente que los prototipos se desplacen cada vez más lentamente, ya que en esta fase sólo será necesario ajustar la colocación de los prototipos. Por este motivo, en el método de Kohonen la tasa de aprendizaje es decreciente con el tiempo. El método utilizado para decrementar la variable α forma parte del esquema de aprendizaje del método. Normalmente se utilizan dos esquemas equivalentes:

- El valor de α se decrementa en una cantidad constante pequeña β , tras cada ciclo completo de todos los patrones de aprendizaje:

$$\alpha(t+1) = \alpha(t) - \beta$$

mediante la asignación del parámetro β se pueden determinar el número total de ciclos aprendizaje que sería:

$$\alpha(t+1) = \alpha(t) - \beta$$

ya que después de dicho número de ciclos el valor de α sería cero y la red estaría estabilizada.

- El valor de α es decrementado siguiendo un esquema logarítmico. En este caso, el decremento es muy elevado en las primeras iteraciones y se va reduciendo paulatinamente hasta que alcanza valores muy pequeños, con una asíntota en el cero. Este esquema hace que se le dé más importancia a las primeras iteraciones, que son las que tienden a formar la estructura de la red, después se ajusta con ligeros desplazamientos de las células hasta alcanzar el equilibrio.

El algoritmo general de los mapas topológicos de Kohonen en forma de tabla quedaría como se muestra en la Figura 16.7.

Lo que se ha querido dejar patente en este algoritmo es que la función de distancia es un parámetro del algoritmo. Variándola, podemos obtener mejores o peores resultados, y el algoritmo puede utilizarse para agrupar, para reducir la dimensionalidad, como paso previo de otros algoritmos, etc. En la página 350 y 351 del Capítulo 13 se muestran ejemplos de uso de las redes de Kohonen.

ALGORITMO Kohonen(E :datos, $d(\cdot, \cdot)$:función, m :entero)

Situar los m prototipos en el espacio.

Iniciarizar α

REPETIR

Extraer un ejemplo e de E (se presentan todos los ejemplos cíclicamente hasta que converja).

Actualizar α .

Calcular la distancia de e con los prototipos usando la función $d(e, \pi_j)$.

Seleccionar el prototipo π_g con distancia menor.

Aproximar el prototipo π_g siguiendo las ecuaciones del aprendizaje.

MIENTRAS $\alpha > \text{umbral}$

FIN ALGORITMO

Figura 16.7. Algoritmo de aprendizaje de mapas de Kohonen.

16.2.2 K medias

El algoritmo K medias (del inglés *Kmeans*) se trata de un método de agrupamiento por vecindad en el que se parte de un número determinado de prototipos y de un conjunto de ejemplos a agrupar, sin etiquetar. Es el método más popular de los métodos de agrupamiento denominados “por partición”, en contraposición de los métodos jerárquicos, que se verán en la sección siguiente. La idea del K medias es situar a los prototipos o centros en el espacio, de forma que los datos pertenecientes al mismo prototipo tengan características similares [Moody & Darken 1989, MacQueen 1967].

Todo ejemplo nuevo, una vez que los prototipos han sido correctamente situados, es comparado con éstos y asociado a aquél que sea el más próximo, en los términos de una distancia previamente elegida. Normalmente, se usa la distancia euclídea.

Las regiones se definen minimizando la suma de las distancias cuadráticas entre cada vector de entrada y el centro de su correspondiente clase, representado por el prototipo correspondiente. El algoritmo puede seguir dos enfoques distintos: K medias por lotes (*batch*) y K medias en línea (*on-line*). El primero se aplica cuando todos los datos de entrada están disponibles desde un principio, mientras que el segundo se aplica cuando no se dispone de todos los datos desde el primer momento, sino que pueden añadirse ejemplos adicionales más tarde. Cuando se aplica la versión por lotes, se debe seleccionar arbitrariamente una partición inicial de forma que cada clase disponga de, al menos, un ejemplo. Como la totalidad de los datos están disponibles, los centros de cada partición se calculan como la media de los ejemplos pertenecientes a esa clase. A medida que el algoritmo se va ejecutando, algunos ejemplos cambian de una clase a otra debiendo recalcularse los centros en cada paso, o sea, desplazar convenientemente los prototipos.

El método tiene una fase de entrenamiento, que puede ser lenta, dependiendo del número de puntos a clasificar y de la dimensión del problema. Pero una vez terminado el entrenamiento, la clasificación de nuevos datos es muy rápida, gracias a que la comparación de distancias se realiza sólo con los prototipos.

El procedimiento es el siguiente:

- Se calcula, para cada ejemplo x_k , el prototipo más próximo A_g y se incluye en la lista de ejemplos de dicho prototipo.

$$A_g = \arg \min_{A_i} \{d(x_k, A_i)\} \quad \forall i = 1..n$$

- Después de haber introducido todos los ejemplos, cada prototipo A_k tendrá un conjunto de ejemplos a los que representa:

$$l(A_k) = \{x_{k_1}, x_{k_2}, \dots, x_{k_m}\}$$

- Se desplaza el prototipo hacia el centro de masas de su conjunto de ejemplos.

$$A_k = \frac{\sum_{i=1}^m x_{k_i}}{m}$$

- Se repite el procedimiento hasta que ya no se desplazan los prototipos.

Mediante este algoritmo el espacio de ejemplos de entrada se divide en k clases o regiones (Figura 8.8), y el prototipo de cada clase estará en el centro de la misma. Dichos centros se determinan con el objetivo de minimizar las distancias cuadráticas euclídeas entre los patrones de entrada y el centro más cercano, es decir minimizando el valor J :

$$J = \sum_{i=1}^k \sum_{n=1}^m M_{i,n} d_{EUCL}(x_n - A_i)^2$$

donde m es el conjunto de patrones, d_{EUCL} es la distancia euclídea, x_n es el ejemplo de entrada n , A_i es el prototipo de la clase i , y M_{in} es la función de pertenencia del ejemplo n a la región i de forma que vale 1 si el prototipo A_i es el más cercano al ejemplo x_n y 0 en caso contrario, es decir:

$$M_{i,n} = \begin{cases} 1 & \text{si } d_{EUCL}(x_n - A_i) < d_{EUCL}(x_n - A_s) \forall s \neq i, s = 1, 2, \dots, k \\ 0 & \text{en caso contrario} \end{cases}$$

Éste es un caso parecido al de los mapas de Kohonen, la diferencia fundamental está en que los prototipos no son desplazados después de la presentación de cada ejemplo, sino al final de cada iteración, con lo cual hay que determinar una regla para mover los prototipos. Esta regla es simplemente moverlos hasta el centro de masas de los ejemplos a los que representa. Como tras una nueva iteración los ejemplos son reasignados, debido al desplazamiento de todos los prototipos al final de la iteración anterior, es necesario volver a calcular los centros de masas de los nuevos ejemplos. A medida que los prototipos vayan situándose, se producirán menos cambios en sus conjuntos de ejemplos, y su desplazamiento será menor. Así, poco a poco los desplazamientos se van haciendo más ligeros, hasta desaparecer. La colocación final de los prototipos definirá la solución encontrada por el método. En la Figura 16.8 se ilustra un ejemplo de este proceso.

Una de las cuestiones más importantes en este método es el valor de K inicial. Por ejemplo, la Figura 16.8 muestra el ejemplo ideal; existen, aparentemente, cinco grupos y los prototipos se ajustan para capturar esos grupos. Si hubiéramos creado cuatro prototipos, el resultado habría sido bien diferente, porque al menos dos grupos “reales” habrían acabado unidos. No obstante, hay que tener en cuenta que incluso con cinco prototipos podría no haber convergencia y no capturar los cinco grupos “reales”. Además, hay que considerar que si se incluyen demasiados prototipos el resultado obtenido puede no ser mejor, al contrario, ya que habrá grupos “reales” que serán divididos de forma artificial e ineficiente.

Finalmente, al igual que en el método de los mapas de Kohonen, la función de distancia es un parámetro del método y puede ser modificada para adaptarse al dominio del problema.

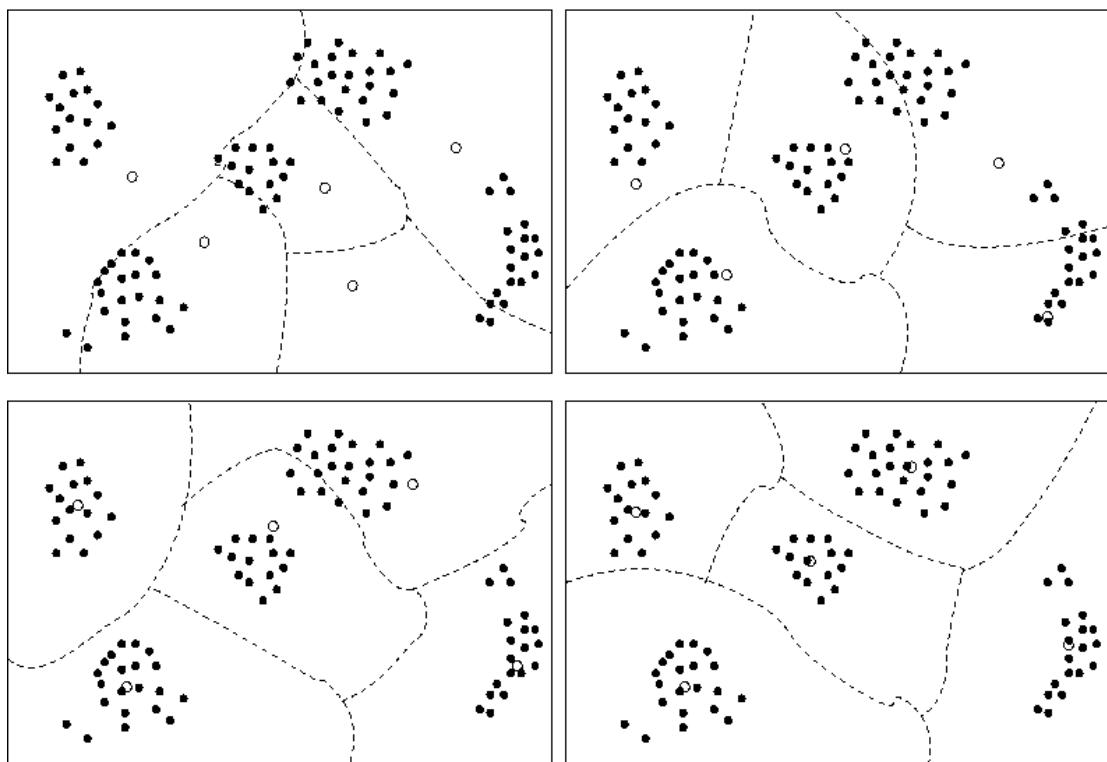


Figura 16.8. Ejemplo de evolución de los prototipos y grupos formados con el método K medias.

Ejemplo de agrupamiento con K medias

Pasemos a ver un ejemplo de uso de K medias. Para ello vamos a utilizar los datos de una muestra del censo estadounidense de 1994 (denominada *census-adult*, véase Apéndice B) que contiene información de 48.842 trabajadores, de los cuales se conocen, entre otros campos, la edad, el tipo de trabajo, la educación (estudios), el estado civil, la ocupación, la raza, el sexo, las horas de trabajo por semana, el país de procedencia y el sueldo (en dos categorías: >50.000 dólares y <=50.000 dólares). Los datos están divididos en un conjunto de entrenamiento, de 32.561 trabajadores, y un conjunto de test, de 16.281 trabajadores.

Vamos a utilizar la versión del K medias que se encuentra en el sistema SPSS Clementine (véase Apéndice A). Para realizar un agrupamiento debemos etiquetar todos los atributos como entrada y ninguno de salida, ya que queremos encontrar *grupos*, sin considerar ningún atributo especial sobre los demás. Si utilizamos el K medias con K=2 (llamados “conglomerados” o “clusters” según la versión del Clementine), obtenemos dos grupos. En la siguiente tabla mostramos los dos grupos y los atributos que los determinan: la media en el caso de que el atributo sea numérico y los valores más frecuentes de los atributos nominales (hasta llegar al 60 por ciento o un máximo de tres valores):

GRUPO 1 (18.373 ejemplos)	GRUPO 2 (14.188 ejemplos)
<p>Edad : 35</p> <p>TipoDeTrabajo : Private -> 0.736516</p> <p>Educación : Bachelors -> 0.144615</p> <p> HS-grad -> 0.326621</p> <p> Some-college -> 0.253143</p> <p>EducaciónNum : 10</p> <p>EstadoCivil : Divorced -> 0.238666</p> <p> Never-married -> 0.581451</p> <p>Ocupación : Adm-clerical -> 0.162848</p> <p> Other-service -> 0.149187</p> <p> Sales -> 0.112284</p> <p>Relación : Not-in-family -> 0.449628</p> <p> Own-child -> 0.272792</p> <p>Raza : White -> 0.816307</p> <p>Sexo : Female -> 0.545693</p> <p> Male -> 0.454308</p> <p>HorasPorSemana : 38</p> <p>PaísDeOrigen : United-States -> 0.894791</p> <p>Sueldo : <=50K -> 0.943939</p>	<p>Edad : 44</p> <p>TipoDeTrabajo : Private -> 0.645898</p> <p>Educación : Bachelors -> 0.190161</p> <p> HS-grad -> 0.31717</p> <p> Some-college -> 0.186072</p> <p>EducaciónNum : 10</p> <p>EstadoCivil : Married-civ-spouse -> 0.990556</p> <p>Ocupación : Craft-repair -> 0.180786</p> <p> Exec-managerial -> 0.169298</p> <p> Prof-specialty -> 0.146674</p> <p>Relación : Husband -> 0.929871</p> <p>Raza : White -> 0.903439</p> <p>Sexo : Male -> 0.94749</p> <p>HorasPorSemana : 44</p> <p>PaísDeOrigen : United-States -> 0.897238</p> <p>Sueldo : <=50K -> 0.519947</p> <p> >50K -> 0.480053</p>

La partición realizada no se puede considerar excesivamente nítida, aunque sí que permite una interpretación bastante razonable. El primer grupo (el mayoritario) está formado por personas, por lo general, jóvenes, solteras o divorciadas, con leve predominio de mujeres, horas por semana menor que la media (puede indicar contratos a tiempo parcial frecuentes) y con un sueldo generalmente menor de 50.000 dólares. En cambio, el segundo grupo (algo menor en tamaño que el primero) está formado por personas maduras, casadas, casi en su totalidad hombres y blancos, trabajando a tiempo completo y posiblemente con horas extras (por la media de horas por semana alta) y con unos sueldos más altos que en el primer grupo. Para esta agrupación, aparecen como atributos no relevantes (al menos para casos con una cierta frecuencia) el tipo de trabajo, la educación (nivel de estudios) y el país de origen (aunque aquí sería interesante agrupar los países por continentes o por el tipo de país).

Aunque la partición puede resultar interesante, no sabemos si el valor de k elegido es adecuado. Para indagar un poco más en el ejemplo, vamos a realizar una partición usando K medias con un valor de k=3. En este caso, sin mostrar los grupos formados, prácticamente el agrupamiento se realiza atendiendo a dos atributos: sexo y sueldo. No es muy difícil dar una interpretación a los grupos formados, pero hemos de ir con precaución, si con k=2, la distancia entre grupos era de 1,308, ahora tenemos una distancia de grupos de 1,14 (1-2), 1,17 (1-3) y 1,659 (2-3), indicando que mientras la distancia entre los grupos 2 y 3 es mayor, esto se ha hecho a costa de un grupo, en cierto modo intermedio, el 1, donde las distancias se han reducido.

Una manera de aportar un poco más de luz sobre si es mejor la partición con k=2 o con k=3 es comparar los grupos formados por el K medias con los que formaría otro método de agrupamiento. Para ello, vamos a utilizar otro algoritmo de agrupamiento diferente, el *Two-step*, de la familia jerárquica, que se verá en la sección siguiente. Sin entrar en más detalles sobre este segundo método propietario de SPSS, podemos generar dos particiones diferentes con *Two-step*, la primera con dos grupos y la segunda con tres grupos. Si utilizamos los datos de test para comparar qué ejemplos caen en cada grupo, según el

método Kmeans o el método *Two-step* y lo mostramos en una matriz de contingencia tenemos el resultado que se muestra en la Figura 16.9:

Método *Two-step* con 2 y 3 grupos, respectivamente

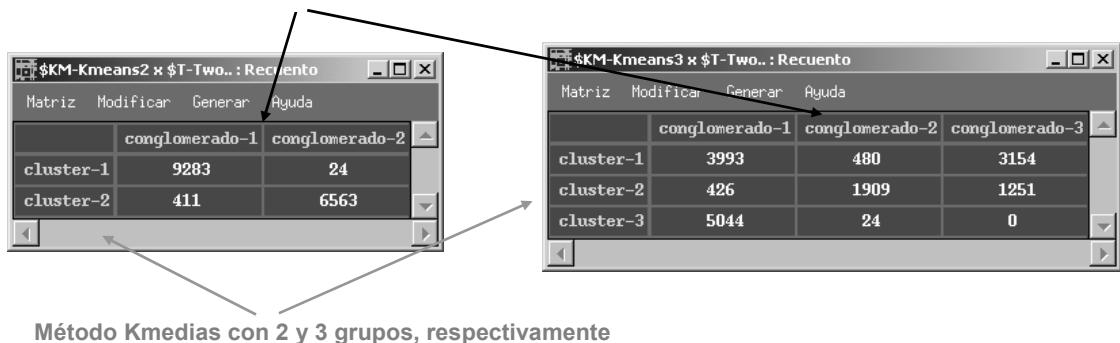


Figura 16.9. Matrices de contingencia para comparar los grupos formados con dos métodos diferentes.

En la parte izquierda de la Figura 16.9 se observa que la correspondencia de grupos en la partición de dos grupos es bastante clara (el conglomerado-1 del *Two-step* coincide claramente con el cluster-1 del K medias, y lo mismo ocurre con el conglomerado-2 y el cluster-2). Por el contrario, la equiparación de grupos en el caso de partición de tres grupos es bastante dudosa, como se ve en la parte derecha de la Figura 16.9. Estas observaciones nos sugieren quedarnos con la primera partición y mantener, por tanto, el número de grupos extraídos en 2.

16.2.3 Agrupamiento jerárquico

Como hemos visto en el ejemplo anterior, uno de los problemas del agrupamiento es discernir, a priori, cuántos grupos puede haber en los datos. De hecho, en la Figura 16.8 puede parecer *evidente* que hay cinco grupos, pero otro observador puede pensar que el grupo de la parte inferior derecha tiene tres puntos en la parte superior que podrían considerarse un nuevo grupo. Lo mismo ocurre con el grupo de la parte inferior izquierda; alguien puede opinar que hay dos puntos “descolgados” que podrían considerarse un nuevo grupo. En realidad, el número de grupos que realmente existen en los datos es algo relativo y depende de los umbrales de unidad o separación que se establezcan. Con este razonamiento nacieron los métodos jerárquicos de agrupamiento.

Los métodos jerárquicos se basan en la construcción de un árbol en el que las hojas son los elementos del conjunto de ejemplos, y el resto de los nodos son subconjuntos de ejemplos que pueden ser utilizados como particionamiento del espacio. Este gráfico se denomina dendrograma y se muestra en la Figura 16.10.

En la Figura 16.10 se muestra un ejemplo de un árbol de agrupamientos. En la raíz está el conjunto de ejemplos, en este caso 11 ejemplos etiquetados de la *a* a la *k*. Cada descendiente es una división del nodo de partida, de forma que van describiendo sucesivos subconjuntos de ejemplos. Una particularidad de este tipo de árboles es que cada nodo está situado en un nivel diferente de todos los demás. De esta forma se genera una jerarquía de nodos, que da nombre al conjunto de métodos, que permite la obtención de diferentes soluciones.

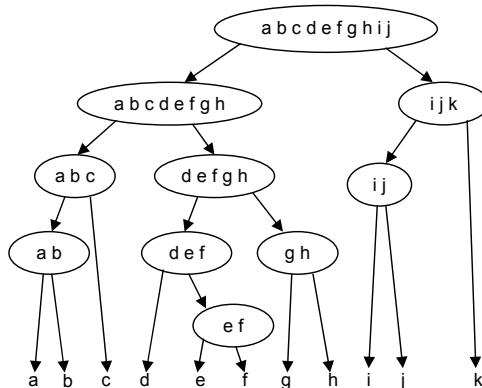


Figura 16.10. Ejemplo de un árbol de agrupamiento (dendrograma).

Así, en la figura existe una jerarquía de diez niveles:

- Nivel 1: Conjunto a,b,c,d,e,f,g,h,i,j,k
- Nivel 2: Conjunto a,b,c,d,e,f,g,h
- Nivel 3: Conjunto i,j,k
- Nivel 4: Conjunto i,j
- Nivel 5: Conjunto a,b,c
- Nivel 6: Conjunto d,e,f,g,h
- Nivel 7: Conjunto d,e,f
- Nivel 8: Conjunto a,b
- Nivel 9: Conjunto g,h
- Nivel 10: Conjunto e,f

Esta estructura jerárquica permite generar varios agrupamientos, dependiendo de lo compacta que se desee la solución o del número de grupos a generar. Para ello se elige un nivel en la jerarquía, se desprecian todos los descendientes de los nodos del mismo nivel y superior al seleccionado, y las hojas del árbol resultante definen el agrupamiento generado. Así, en el ejemplo anterior, si se elige el nivel 6 se obtiene el siguiente agrupamiento con cuatro clases:

$$A = \{ \{a,b,c\}, \{d,e,f,g,h\}, \{i,j\}, \{k\} \}$$

pero si se elige el nivel 3 el agrupamiento que se genera es:

$$A = \{ \{a,b,c,d,e,f,g,h\}, \{i,j,k\} \}$$

En la Figura 16.11, se muestran algunos ejemplos a distintos niveles:

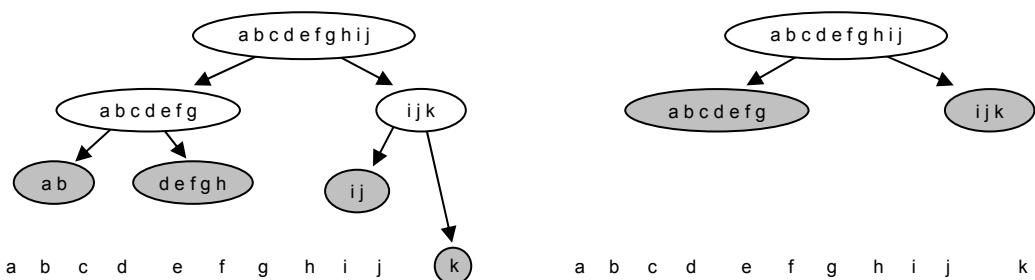


Figura 16.11. Agrupamientos de nivel 6 y 2 respectivamente.

Dependiendo de la manera de construir el árbol los métodos se dividen en:

- **Aglomerativos.** El árbol se va construyendo empezando por las hojas, hasta llegar a la raíz. En un primer momento cada ejemplo es a su vez un grupo, se van agrupando los grupos para formar conjuntos cada vez más numerosos, hasta llegar a la raíz, que contiene a todos los ejemplos.
- **Desaglomerativos o divisivos.** Se parte de la raíz, que es un solo grupo conteniendo a todos los ejemplos, y se van haciendo divisiones paulatinas hasta llegar a las hojas que representan a la situación en que cada ejemplo es un grupo.

Los métodos aglomerativos parten de dos principios fundamentales. La forma de seleccionar los grupos a mezclar, y la manera de mezclarlos. El método más común es elegir aquellos grupos cuya **distancia de enlace** (*link distance*) sea menor. Una manera de hacerlo (entre otras, como se verá a continuación) es obligar a que cada grupo tenga un **representante** (que puede crearse como un centroide) que será utilizado como elemento de referencia para el cálculo de las distancias. La mezcla de grupos consiste en hacer que todos los ejemplos de los grupos que se van a mezclar pasen a ser miembros del nuevo grupo. En cuanto al representante del nuevo grupo creado suele ser el centro de masas de los puntos pertenecientes a la clase.

El método final quedaría como sigue:

1. Hacer que cada punto sea el representante de un grupo que sólo contiene dicho punto.
2. Calcular las distancias entre todos los grupos existentes dos a dos.
3. Elegir los dos grupos cuya distancia sea menor.
4. Mezclar los grupos elegidos en el paso anterior. Si el representante de uno de los grupos es el vector $\vec{C}_a = \{c_{a_1}, c_{a_2}, \dots, c_{a_n}\}$, y el del otro grupo $\vec{C}_b = \{c_{b_1}, c_{b_2}, \dots, c_{b_n}\}$, si además el grupo a tiene j ejemplos y el grupo b tiene k ejemplos; el nuevo representante se calculará mediante la expresión:

$$\vec{C} = \left\{ \frac{j \cdot c_{a_1} + k \cdot c_{b_1}}{j+k}, \frac{j \cdot c_{a_2} + k \cdot c_{b_2}}{j+k}, \dots, \frac{j \cdot c_{a_n} + k \cdot c_{b_n}}{j+k} \right\}$$

5. Si hay más de un grupo ir a 2.

Dependiendo de cómo se calcule la distancia de enlace entre grupos se pueden distinguir tres métodos:

- **Enlace simple** (*single linkage*): para el cálculo de la distancia no se utilizan los representantes, sino que se calcula la distancia entre todos los puntos de dos grupos y se toma como distancia entre grupos la menor.
- **Enlace completo** (*complete linkage*): igual que el anterior, pero se toma como distancia entre grupos la mayor de todas.
- **Enlace en la media** (*average linkage*): se toma como distancia la existente entre los representantes (centroídes) de los grupos.

Sea el ejemplo de la Figura 16.12:

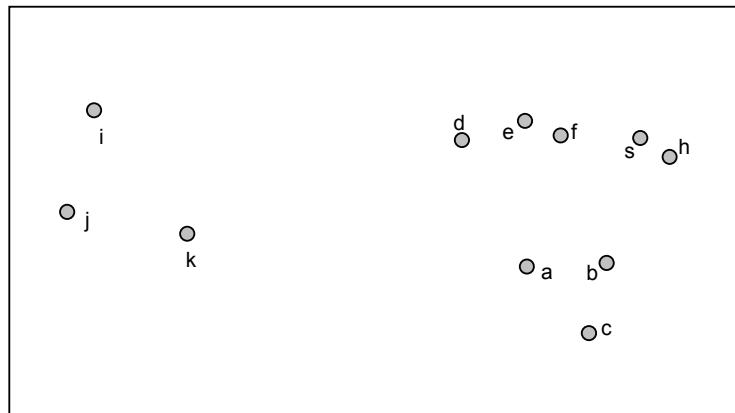


Figura 16.12. Ejemplo de distribución de puntos bidimensional.

Si se aplica el algoritmo de agrupamiento jerárquico anterior aglomerativo y con enlace de la media, habrá que calcular, en primer lugar, las distancias de los puntos de la distribución dos a dos. Sean dichas distancias las de la Tabla 16.1.

	a	b	c	d	e	f	g	h	i	j	k
a	-	2,4	4	9	7,8	7	8,9	8,7	27,6	28,3	21,4
b	-	-	4,3	11,4	9,1	7,2	7,3	7,4	30,8	32,6	26,8
c	-	-	-	13,6	12,2	11	12,6	10,5	31	31,9	25,2
d	-	-	-	-	4,8	6,4	10,9	12	21	23,7	17,6
e	-	-	-	-	-	1,7	6	7,4	26,3	28,7	22,5
f	-	-	-	-	-	-	4,5	5,6	27,3	30	23,6
g	-	-	-	-	-	-	-	1,8	31,9	34,3	28,8
h	-	-	-	-	-	-	-	-	33,4	35,4	28,8
i	-	-	-	-	-	-	-	-	-	4	6,8
j	-	-	-	-	-	-	-	-	-	-	6,9
k	-	-	-	-	-	-	-	-	-	-	-

Tabla 16.1. Tabla de distancias del ejemplo de la Figura 16.12.

La distancia más corta es entre los puntos *e* y *f*. Se crea un nuevo punto al que llamaremos *ef*, y se coloca en el punto medio de los dos. Se eliminan de la tabla los puntos *e* y *f*. Se calculan las distancias de este nuevo punto al resto:

	a	b	c	d	ef	g	h	i	j	k
a	-	2,4	4	9	7,4	8,9	8,7	27,6	28,3	21,4
b	-	-	4,3	11,4	8,5	7,3	7,4	30,8	32,6	26,8
c	-	-	-	13,6	11,6	12,6	10,5	31	31,9	25,2
d	-	-	-	-	5,6	10,9	12	21	23,7	17,6
ef	-	-	-	-	-	5,2	6,4	26,8	29,1	22,9
g	-	-	-	-	-	-	1,8	31,9	34,3	28,8
h	-	-	-	-	-	-	-	33,4	35,4	28,8
i	-	-	-	-	-	-	-	-	4	6,8
j	-	-	-	-	-	-	-	-	-	6,9
k	-	-	-	-	-	-	-	-	-	-

Tabla 16.2. Tabla de distancias del ejemplo de la Figura 16.12 tras la primera fusión de grupos.

Ahora la distancia más corta es entre las clases g y h , por lo que también formarán una categoría y así sucesivamente, hasta formar el árbol de la Figura 16.10. Este árbol puede ser representado en la nube de puntos como conjuntos y subconjuntos como refleja la Figura 16.13.

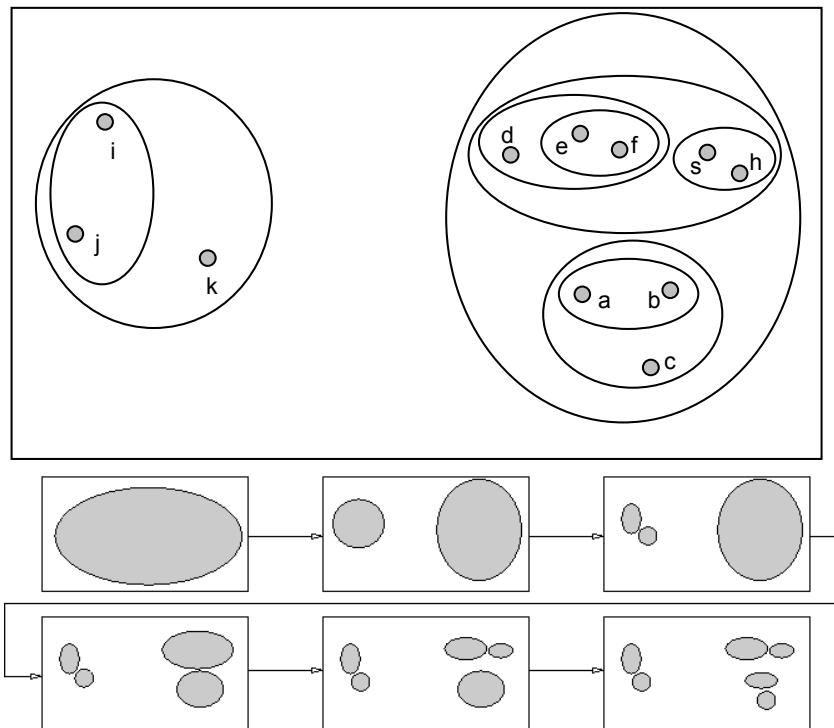


Figura 16.13. Agrupamientos posibles mediante un método jerárquico, de la nube de puntos de la Figura 16.12.

En la parte inferior se muestran los distintos agrupamientos si se examina el árbol a distintos niveles. Como se observa, la gran ventaja de estos métodos es que se puede elegir el nivel en el que la diferencia entre grupos sea más clara. Para ello se puede utilizar la distancia de enlace entre grupos (*link distance*).

Además de la versión divisiva, que es muy similar a la aglomerativa, existen variantes muy conocidas de los algoritmos jerárquicos que escalan bien para un número de ejemplos mayor. Uno de los algoritmos más populares es el COBWEB [Fisher 1987], que va incorporando los ejemplos incrementalmente al dendograma.

16.3 Técnicas para clasificación

En las secciones anteriores se han descrito métodos basados en distancias y en casos para el agrupamiento. A continuación, se mostrará cómo se puede utilizar la misma filosofía para la clasificación, es decir, para el problema en el que los ejemplos vienen acompañados de la etiqueta de una clase.

Además, se utilizará el concepto de función de densidad, que se complementa muy bien con el concepto de distancia, ya que no es sólo importante que una instancia tenga otra instancia cerca sino qué cantidad de ellas están cerca.

16.3.1 Estimación bayesiana de funciones de densidad

La tarea de clasificación basada en ejemplos parte de la existencia de un conjunto de ejemplos catalogados en clases. A partir de este conjunto de ejemplos se puede determinar el número de clases que existen en el problema. Sea q dicho número, entonces se tendrá el conjunto de clases: $A = \{A_1, \dots, A_q\}$.

Las probabilidades a priori se pueden calcular a partir del conjunto de ejemplos. Simplemente cada probabilidad será la frecuencia de ejemplos de la clase en cuestión respecto del total:

$$\pi_i = \frac{|A_i|}{\sum_j |A_j|}$$

donde $|A_i|$ es el número de ejemplos del conjunto que pertenecen a la clase i .

Como vimos en algunos capítulos anteriores, en especial el Capítulo 10 de métodos bayesianos, se puede calcular la probabilidad “a posteriori”, utilizando el teorema de Bayes:

$$p(A_i | x) = \frac{\pi_i \cdot p(x | A_i)}{\sum_{j=1}^q \pi_j \cdot p(x | A_j)}$$

donde $p(x | A_i)$ es la probabilidad de que x tome determinados valores sabiendo que pertenece a la clase A_i . Como en la expresión anterior el divisor es común a todas las clases, se puede eliminar en el cálculo del máximo:

$$\arg \max_i p(A_i | x) = \arg \max_i \frac{\pi_i \cdot p(x | A_i)}{\sum_{j=1}^q \pi_j \cdot p(x | A_j)} = \arg \max_i \pi_i \cdot p(x | A_i)$$

Hasta ahora tenemos el mismo resultado de otros métodos bayesianos, como el Naive Bayes, visto en el Capítulo 10. Los métodos bayesianos se basan en calcular o estimar dichas probabilidades mediante reglas que utilizan el conjunto de ejemplos [Chen 1969]. Según sea la manera de estimar las probabilidades condicionadas se tiene un tipo de técnicas u otras [Kashyap & Blaydon 1968], paramétricas o no paramétricas (del mismo modo que se vio en los capítulos 6 y 7).

En este capítulo se hablará de algunas técnicas no paramétricas que se basan en la idea de una región local. Partimos de que, si los atributos de los ejemplos tienen distribuciones continuas, se pueden cambiar las probabilidades por funciones de densidad de probabilidad, quedando:

$$p(A_i | x) = \frac{\pi_i \cdot f(x)}{\sum_{j=1}^q \pi_j \cdot f_j(x)}$$

Se puede calcular la función de densidad a partir de la probabilidad de que un dato x esté dentro de una determinada región R . Para ello se parte de un supuesto “a priori” cuya veracidad determinará la precisión del método. Dicho supuesto consiste en considerar a la función de densidad más o menos constante dentro de la región de referencia R .

Si se tiene un conjunto representativo y abundante de ejemplos, la probabilidad de que un nuevo dato x caiga dentro de una región R ($p_R(x)$) se puede estimar mediante la expresión:

$$p_R(x) = \frac{k_n}{n}$$

donde k_n es el número de ejemplos que se encuentran dentro de la región R , de entre los del conjunto de ejemplos, y n es el tamaño de dicho conjunto de ejemplos. Para el cálculo de k_n hay que definir, una vez más, una función de distancia d , de forma que un ejemplo estará dentro de la región R si su distancia al dato x es menor que cierto valor r :

$$x' \in K_n \text{ si } d(x', x) < r$$

Por tanto, la forma y el alcance de la región, no sólo depende del radio sino también de la función de distancia utilizada. Por otra parte, también a partir de la función de densidad se puede calcular la probabilidad $p_R(x)$:

$$p_R(x) = \int_R f(x) dx$$

Si se igualan las dos expresiones anteriores queda:

$$\int_R f(x) dx = \frac{k_n}{n}$$

de donde se puede calcular la función de densidad:

$$f(x) = \frac{k_n}{n v_n} \quad (1)$$

Veamos a continuación un método clásico que es una instancia particular de usar un clasificador bayesiano con esta función de densidad.

16.3.2 K vecinos

La regla del vecino más próximo simplemente asigna la clase del ejemplo más próximo, utilizando una función de distancia. Esta regla, conocida como 1-NN (*one nearest neighbor*), tiene bastantes problemas, ya que ignora la densidad o la región donde se encuentra el ejemplo. Por ejemplo, en la parte izquierda de la Figura 16.14 se muestra un ejemplo de aplicación del vecino más próximo.

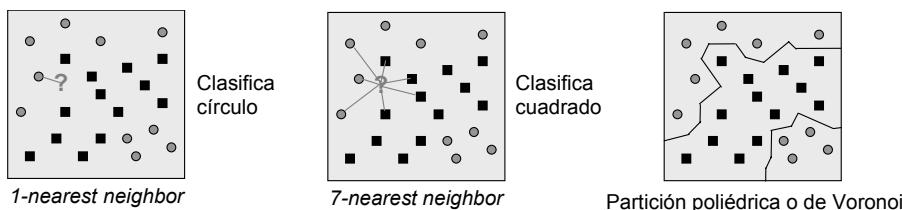


Figura 16.14. Diferencia en el valor de k de los vecinos más próximos y partición realizada.

Una variante de este método son los k vecinos más próximos (kNN, *k-nearest neighbors*) [Cover & Hart 1967] en el que se asigna la clase mayoritaria entre los k vecinos más próximos, como se muestra en el panel del medio de la Figura 16.14. Como se puede observar, el valor de k es muy importante y es difícil establecer el adecuado. De hecho,

como se aprecia en el panel derecho de la Figura 16.14, la expresividad del método es muy alta, y el problema principal de este método es determinar un buen valor de k .

Una versión un poco más elaborada de los vecinos más próximos se basa en determinar una región de cercanía, más que un valor constante de ejemplos a comparar. Una aproximación para definir este método es utilizar la aproximación bayesiana del apartado anterior. Para ello será necesario estimar las distribuciones de probabilidad: $f(x|A_h)$ y π_h . Pues bien, el método k vecinos consiste en asignar dichas probabilidades en función de los vecinos del nuevo dato a clasificar. Para ello se toma el nuevo dato a clasificar x y se crea a su alrededor una hiperesfera que dé un radio r , utilizando una función de distancia. Dicha esfera tendrá un determinado volumen $k(x)$ y contendrá $k_1(x), \dots, k_q(x)$ observaciones de las clases A_1, \dots, A_q respectivamente. Sean π las probabilidades “a priori” de cada una de las clases ($\pi_i = p(A_i)$), estas probabilidades se podrían estimar por la frecuencia de ejemplos que pertenecen a la clase respecto de la totalidad de ejemplos:

$$\pi_h = \frac{n_h}{n}$$

Si a esto se le aplica la función de la ecuación 1 vista en la página anterior para el cálculo de la probabilidad de x condicionada la clase A_h , se obtiene:

$$f(x | A_h) = \frac{k_h(x)}{n_h \times v(x)} \quad (2)$$

Por otra parte, por el teorema de Bayes se sabe que:

$$P(A_h | x) = \frac{\pi_h P(x | A_h)}{\sum_j \pi_j P(x | A_j)}$$

Sustituyendo en la expresión anterior utilizando la ecuación 2 y la fórmula de la estimación de las probabilidades “a priori” se obtiene:

$$P(A_h | x) = \frac{\frac{n_h}{n} \cdot \frac{k_h(x)}{n_h v(x)}}{\sum_j \frac{n_j}{n} \frac{k_j(x)}{n_j}} = \frac{n_h \cdot k_h(x)}{n \cdot v(x)}$$

Operando y simplificando la ecuación anterior se obtiene la expresión definitiva del cálculo de la probabilidad condicionada:

$$P(A_h | x) = \frac{k_h(x)}{\sum_j k_j(x)} = \frac{k_h(x)}{k}$$

Es decir, la probabilidad de que un nuevo ejemplo x pertenezca a una clase A_h es igual a la frecuencia del número de ejemplos que pertenecen a dicha clase y caen dentro de un volumen v alrededor del ejemplo, respecto del número total de ejemplos.

El procedimiento sería:

- Dado un nuevo x se genera el conjunto x' con los ejemplos que están a una distancia menor de r de x .

- Se calculan todos los $k_h(x')$. Es decir, el número de ejemplos en x' que pertenecen a la clase A_h .
- Sea $k_g(x')$ el de mayor valor de todos los $k_h(x')$:

$$k_g = \operatorname{Max}_{h=1, \dots, q} \{k_h\}$$

- Se le asigna a x la clase A_g .

Como se ha mencionado con anterioridad uno de los mayores problemas de los métodos retardados, entre los que se encuentra el k vecinos, es que la predicción puede ser lenta, y depende del tamaño del conjunto de ejemplos. En el caso del k vecinos este problema no afecta demasiado ya que el único cálculo que hay que realizar es una función de distancia entre el nuevo ejemplo y cada uno de los datos de ejemplo, pero estas funciones de distancia suelen ser muy rápidas. Por el contrario, una de las ventajas fundamentales es que para el cálculo de las distancias se puede utilizar una función a conveniencia. La función de distancia sólo afecta a la selección de los ejemplos, no como el K medias o Kohonen donde también es empleada para la ubicación de los prototipos, y por lo tanto con los vecinos más próximos es fácil utilizar medidas más sofisticadas (como la distancia de Mahalanobis, que se usa mucho en este contexto). Otra aproximación es el uso, por ejemplo, de distancias borrosas (*fuzzy*) de pertenencia al conjunto con una determinada intensidad, no numérica, o adaptar la función distancia según lo requiera el problema a tratar. Para problemas de recuperación de la información, los ejemplos podrían ser textos completos, y la medida de distancia podría basarse en vectores de frecuencias de aparición de palabras clave, construyéndose métricas más sofisticadas. O también utilizar diferencias entre el resultado de un preproceso de una imagen, después de filtros o transformadas de Fourier o *wavelets*, como medida de distancia entre dos imágenes.

Finalmente, un problema de la aproximación anterior es, de nuevo, determinar el radio adecuado. Una variante de los vecinos más próximos es el k vecinos con distancia ponderada, donde se evalúan *todas* las instancias, pero ponderadas por una función que dependa inversamente a la distancia. Esta función se conoce como núcleo y suele ser el inverso de la distancia al cuadrado.

$$P(A_h | x) = \sum_{i=1}^n w(x_i, x) \cdot \delta(A_h, \operatorname{clase}(x_i)) \quad \text{donde } w_i(a, b) = \frac{1}{d(a, b)^2}$$

y $\delta(a, b) = 1$ si $a=b$ y 0 en caso contrario.

Esta variante, en principio, ya no tiene como parámetro un valor de K o un radio y, por tanto, si su utilización es posible será preferible sobre las anteriores. No obstante, por cuestiones de eficiencia de la implementación, sólo se considera un radio (en este caso suele ser mayor), y entonces se habla de los k-vecinos ponderados.

Ejemplo reconocimiento del habla

Se va a describir un ejemplo de aplicación de los vecinos más próximos. El problema con el que vamos a trabajar se conoce como Isolet (véase Apéndice B) y consiste en reconocer qué carácter del alfabeto latino ha sido pronunciado por un hablante cualquiera. Aunque el reconocimiento de caracteres hablados pueda parecer una meta modesta a primera vista, debido al reducido tamaño de la población, es una tarea en realidad más difícil. Muchos pares de letras, como M-N y B-D, se diferencian en una característica articulatoria simple.

La captura de los datos se ha realizado mediante un micrófono cancelador de ruido, concretamente el modelo Sennheiser HMD 224, un filtrado paso bajo a 7,6 kHz y muestreo a 16 kHz. De cada sonido pronunciado se extraen un gran número de características referentes a la frecuencia del sonido, duración, segmentación de la onda, etc., hasta obtener un total de 617 atributos relevantes. Como vemos, se trata de un problema con alta dimensionalidad.

Para la recopilación de los datos, 150 sujetos pronunciaron el nombre de cada letra del alfabeto dos veces. Por lo tanto, hay 52 ejemplos pertenecientes a cada hablante, un total de 7.800 ejemplos, de los cuales 203 han sido descartados por problemas en la grabación, con lo que el conjunto de datos está formado por 7.597 ejemplos. De todos estos ejemplos, se utilizará el 80 por ciento para entrenamiento, y el 20 por ciento restante para la verificación sobre datos independientes. Una vez hecha esta división, quedan dos subconjuntos de datos, uno de entrenamiento con 6.077 instancias y otro de verificación con 1.520.

Cada ejemplo, formado por 617 atributos, es escalado de tal forma que esté dentro de un rango entre -1 y 1. Los ejemplos están igualmente distribuidos en cada una de las clases, existen 26 clases, etiquetadas de la A a la Z, con 292 ejemplos de cada una (en algunos casos uno más o uno menos debido a los desechados).

Se han realizado 15 experimentos con tres algoritmos diferentes (Discriminante lineal, Reglas de decisión C4.5, y k-vecinos), según su implementación en el sistema WEKA (véase Apéndice A). Se va a mostrar, para cada uno de los métodos, las tasas de acierto y una comparativa entre ellos.

Para el caso del discriminante lineal (véase el Capítulo 7 o el principio de este mismo capítulo), la clasificación no ha sido buena, de las 6.077 instancias solamente clasifica correctamente 1.799, lo que implica un porcentaje de acierto de 29,6 por ciento, y esto sobre el conjunto de aprendizaje. A continuación se evalúa el modelo obtenido en la fase de aprendizaje sobre el conjunto de datos de validación, obteniéndose el siguiente resultado: de las 1.520 instancias solamente clasifica correctamente 205, lo que implica un porcentaje de acierto de 13,49 por ciento (lo que implica un porcentaje de fallo de 86,51 por ciento).

A continuación se muestra el resultado con reglas de decisión obtenidas por el algoritmo de aprendizaje de árboles de decisión C4.5 (véase el Capítulo 11), fijándose el número mínimo de instancias contenidas en cada hoja terminal a 2, y sin poda.

El árbol que se genera tiene un tamaño de 767 niveles y 384 hojas, dicho árbol se aplica al conjunto de evaluación de 1.520 ejemplos, obteniéndose que son clasificadas correctamente 1.227 instancias (porcentaje de acierto de 80,72 por ciento).

Finalmente, se utiliza el método de k-vecinos, con un valor de k concreto, es decir se fija el número de vecinos a considerar aunque algunos de ellos resulten estar lejos. Se han realizado experimentos con varios valores de k, como se mostrará en la tabla de resultados. Se realiza la misma experimentación que en los casos anteriores, obteniéndose bien clasificados un total de 1.362 de los 1.520 ejemplos (porcentaje de acierto de 89,60 por ciento).

En estos ejemplos de experimentos se aprecia cómo los clasificadores lineales ofrecen unos resultados muy pobres cuando se les enfrenta a problemas complejos como en este caso. Los resultados mejoran con un sistema basado en reglas, ya que la precisión es mucho mayor, en contraste con lo que ocurría con la del discriminante lineal. Efectivamente la

mayoría de los caracteres están bien clasificados, habiendo pequeñas desviaciones como, por ejemplo, el caso de la N que es clasificada 11 veces como A, o el de la E, que es clasificada 11 veces como B, y algunos otros pequeños errores. Sin embargo con el k vecinos se consiguen aún mejores resultados, tal vez el error más significativo es el del carácter S que es clasificado 12 veces como F. Se han realizado más experimentos, cuyo resumen se muestra en la tabla siguiente:

Método	Exp.	Precisión	Error		Tamaño	
			Eam	Ecm	Niveles	Hojas
Disc. Lineal	1	13,49%	0,0665	0,258	1	249
	2	19,28%	0,0621	0,2492	1	21
C4.5	1	79,54%	0,0161	0,1189	767	384
	2	81,71%	0,0179	0,11	277	139
	3	79,67%	0,016	0,1184	723	362
	4	79,87%	0,0159	0,1179	707	354
	5	80,72%	0,0182	0,1082	281	141
	6	80,46%	0,0208	0,1114	163	82
	7	80,59%	0,0185	0,1095	273	137
	8	80,46%	0,0187	0,1103	269	135
K Vecinos	1	89,60%	0,0083	0,0892		
	2	88,88%	0,0142	0,0805		
	3	90,07%	0,0139	0,0793		
	4	90,92%	0,0166	0,0793		
	5	91,51%	0,0164	0,0786		

La tabla muestra los resultados de los métodos anteriores con diferentes parámetros. Así, de k vecinos se han realizado cinco experimentos donde se ha variado el número de vecinos, y el método para calcular la clase resultante, utilizándose a veces la distancia ponderada. De esta forma los experimentos anteriores se corresponden con: un vecino, cinco vecinos, cinco vecinos ponderados, 15 vecinos y 15 vecinos ponderados. Se aprecia cómo los mejores resultados se obtienen para el método de k vecinos, aumentándose la eficacia al aumentar el vecindario, y siendo más eficaz el método ponderado.

16.3.3 LVQ

En este apartado se va a estudiar un método de clasificación basado en las redes de neuronas artificiales no supervisadas de Kohonen, estudiadas previamente. Dicho método se conoce con el nombre de Redes de Cuantización Vectorial (en inglés *Learning Vector Quantization-LVQ*) [Kohonen 1986].

Se puede decir que es una versión supervisada del método de Kohonen. En dicho método se introduce cada uno de los ejemplos de entrenamiento en la red. Para cada ejemplo se calcula la célula ganadora de la capa de competición. En esta capa están los prototipos de las clases que producirá como salida la red. Así pues, la célula ganadora se corresponderá al prototipo asignado al ejemplo de entrada, que es la clase a la que se le hará pertenecer. A continuación se realiza el aprendizaje para dicha célula ganadora y las de su vecindario. Esto no puede hacerse en un aprendizaje supervisado, pues ya se conoce a priori a qué clase pertenece el ejemplo de entrenamiento. Así pues, se ha modificado el método para tener en cuenta esta información. El procedimiento es similar al de Kohonen,

ya explicado en detalle anteriormente, de modo que aquí se describirá únicamente qué es lo que varía entre ambos métodos.

En primer lugar, a partir del conjunto de entrenamiento se conoce el número de categorías o clases existentes, por tanto en la capa de competición se introducen ese mismo número de células o mayor si se piensa que las distribución de puntos de cada clase no es uniforme. Estas células representan los prototipos de la clasificación, y se distribuyen inicialmente de forma aleatoria por el espacio de estados. La solución al problema de clasificación será la colocación final de dichos prototipos, o células de la capa de competición, junto con la regla del vecino más cercano. La colocación de los prototipos vendrá especificada por los valores de los pesos de las conexiones entre la capa de entrada y la capa de competición de la red. Mientras que la regla del vecino más cercano funciona de la siguiente manera:

1. Se introduce el nuevo ejemplo a clasificar.
2. Se calcula la distancia de este nuevo ejemplo a todos los prototipos existentes.
3. Se etiqueta al nuevo ejemplo como de la clase del prototipo cuya distancia calculada en el paso 2 sea más pequeña.

En un modelo de red de neuronas artificiales esta regla equivale simplemente a introducir el ejemplo en la red, propagarlo hasta obtener una salida y asignarle la clase a la que pertenece la célula ganadora.

Una vez distribuidos los prototipos de forma aleatoria, se van desplazando a medida que se van introduciendo los ejemplos de entrenamiento. Para ello se utiliza la misma regla de aprendizaje que en el método de Kohonen:

$$\frac{d_{\mu_{ij}}}{dt} = \alpha(t) \cdot \tau_j(t) \cdot (\varepsilon_i(t) - \mu_{ij}(t))$$

Pero en este caso se varía la función τ de la siguiente forma:

$$\tau_i = \begin{cases} 1 & \text{si } C_i \text{ ganadora y pertenece a la misma clase que } \varepsilon_i \\ -1 & \text{si } C_i \text{ ganadora y no pertenece a la misma clase que } \varepsilon_i \\ 0 & \text{en caso contrario} \end{cases}$$

Esta modificación de la función τ cambia sustancialmente el procedimiento. En primer lugar, al ser aprendizaje supervisado se ha eliminado el concepto de vecindario. Ahora sólo se modificará la célula ganadora, y el resto no sufrirán ningún cambio. La modificación de la célula ganadora no es siempre en la misma dirección. En el método de Kohonen, la célula ganadora se aproximaba siempre al ejemplo introducido, de esta forma las células acababan en el centro de agrupaciones de ejemplos. En el método LVQ, si la célula ganadora pertenece a la misma clase que el ejemplo, la salida de la red es correcta, y para reforzarla se aproxima la célula al ejemplo, de esta forma también se colocará cerca de los ejemplos a los que representa, y que son de su misma clase. Sin embargo, si la célula ganadora no pertenece a la misma clase que el ejemplo, se estará produciendo una salida incorrecta, y habrá que penalizar dicha salida. Para ello lo que se hace es *alejar* (de ahí el signo negativo de la función τ_i en el segundo supuesto) la célula del ejemplo para que en presentaciones posteriores de este mismo ejemplo haya otras células que lo representen y no esta. De esta forma se consigue que los prototipos se vayan paulatinamente acercando a

los ejemplos a cuya clase representan, y alejando de aquellos ejemplos a los que no representan.

En este caso, el valor de la tasa de aprendizaje α también se decrementa con el tiempo, al igual que ocurría en el método de Kohonen. Se recomienda que este parámetro se inicie a valores pequeños, 0,1 y se decremente de forma lineal muy ligeramente.

ALGORITMO Kohonen(E :datos, $d(\cdot, \cdot)$:función, m :entero)

Situar los m prototipos en el espacio e inicializar aleatoriamente los pesos.

Inicializar α

REPETIR

Extraer un ejemplo e de E (se presentan todos los ejemplos cíclicamente hasta que converja).

Actualizar α

Calcular la distancia de e con los prototipos usando la función $d(e, \pi_j)$.

Seleccionar el prototipo π_g con distancia menor.

Aproximar o alejar el prototipo π_g siguiendo las ecuaciones del aprendizaje.

MIENTRAS los prototipos se desplacen o $\alpha > 0$

FIN ALGORITMO

Figura 16.15. Algoritmo de aprendizaje de redes LVQ.

El método LVQ se encuentra dentro de los métodos basados en distancia, ya que utiliza una medida de distancia para calcular el prototipo ganador así como para la función de aprendizaje, tanto para aproximar como para alejar a los prototipos de los ejemplos de entrenamiento. Al igual que en los otros métodos basados en distancia, ésta es un parámetro del método. No tiene tanta versatilidad como en el caso de k vecinos, pero la distancia se puede modificar y adaptar a cada dominio concreto. Otro parámetro del método es el número de prototipos. En general este valor suele ser mayor que el número de clases, ya que una misma clase puede estar distribuida en zonas dispersas (distantes) del espacio y necesitar, por tanto, más de un “grupo” para capturarla.

16.4 Métodos de vecindad con técnicas evolutivas

Vistos los métodos de vecindad clásicos, para agrupamiento, como por ejemplo el K medias o los métodos jerárquicos, o para clasificación, como por ejemplo el k vecinos, se van a describir dos nuevos métodos, algo más sofisticados y más potentes, para la tarea de clasificación, que se orientan en utilizar métodos evolutivos para ir acercando los prototipos, utilizando para ello funciones basadas en vecindad o distancia como función de adaptación.

16.4.1 Clasificación por vecindad mediante algoritmos genéticos

Como se ha mencionado a lo largo del capítulo, en una aproximación por vecindad para el problema de clasificación, una solución consistirá en un conjunto de vectores representando las posiciones de los prototipos empleados por la regla del vecino más cercano:

$$P = \{p_0, p_1, \dots, p_m\}$$

donde $p_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$.

Una vez obtenidas las posiciones de los prototipos, un nuevo dato será asociado con la clase a la que pertenece su prototipo más cercano. Así pues, cualquier método de búsqueda

heurística de las posiciones de los prototipos puede ser utilizado como método de clasificación. Uno de estos métodos son los algoritmos genéticos.

Los algoritmos genéticos, como vimos en el Capítulo 15, basan gran parte de su eficacia en una buena representación del problema. En este caso la representación es en forma de cadena de caracteres pertenecientes a un alfabeto restringido ($A=\{a_1, a_2, \dots, a_n\}$) llamado alfabeto de alelos. Habitualmente el alfabeto suele ser binario, $A=\{0,1\}$, pero si el problema lo requiere puede ser cualquier tipo de alfabeto. A cada elemento de representación se le llama individuo, y se relaciona con una posible solución al problema [Goldberg 1989, Holland 1975]. En el caso de la clasificación, los individuos serían cadenas de números reales representando las posiciones de los prototipos en el espacio de estados.

Sea el problema de la Figura 16.16, donde existe un conjunto de ejemplos pertenecientes a dos clases. Cada ejemplo consta de dos atributos, representados gráficamente, los ejemplos forman círculos concéntricos alternos por clases, cada uno conteniendo únicamente puntos de una clase.

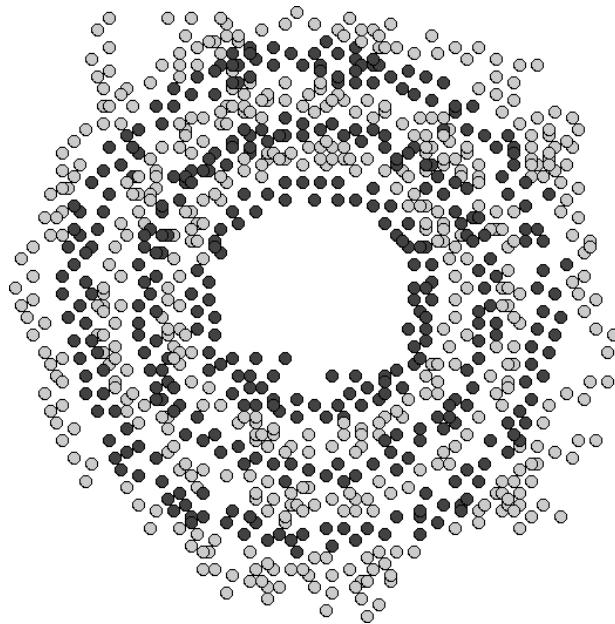


Figura 16.16. Ejemplo de problema de clasificación con dos clases.

Este ejemplo es un caso claro en el que se necesita un número de prototipos elevado si se utiliza la distancia euclídea, ya que la distribución de ejemplos no permite su solución con sólo dos prototipos. En este caso cada prototipo vendrá representado por un vector bidimensional. Si se normalizan las dos dimensiones entre cero y uno, y, además, se discretiza el espacio en bloques de 10^{-2} , se puede utilizar una representación binaria con siete bits de precisión. De esta forma cada prototipo estará representado por una cadena de 14 bits. Si se introducen, pongamos, 20 prototipos para solucionar el problema, y cada individuo se corresponde con una solución, se necesitarán 280 bits en total para representar a cada individuo.

Lógicamente, como todo método evolutivo, el procedimiento parte de la generación de un conjunto de soluciones, llamado población de individuos. La población inicial se genera de manera aleatoria, de forma que las soluciones producidas en primera instancia son muy

malas. A partir de esta población inicial se van generando poblaciones sucesivas de soluciones, según un determinado procedimiento que se explicará más adelante. El objetivo del método es que, pasados un número de ciclos (en terminología de algoritmos genéticos se denomina generaciones), la población generada contenga muy buenas soluciones al problema.

La generación de nuevas poblaciones se realiza aplicando una serie de operadores a la población antigua. A estos operadores se les llama operadores genéticos. Los operadores genéticos más comunes, como se vio en el Capítulo 15, son: reproducción, cruce y mutación. La reproducción consiste en seleccionar de entre todos los individuos de una población aquellos que mejor resuelvan el problema. Dicha selección se hace de forma aleatoria, no uniforme, donde la probabilidad de que un individuo sea seleccionado será proporcional a lo bien que resuelva el problema. Para esto se necesita una función de evaluación o adaptación (*fitness*) que determine en qué medida una solución es buena o mala. Esta función toma como argumento un individuo y produce un valor numérico a maximizar.

En el caso del ejemplo anterior de las zonas concéntricas (Figura 16.16), una función de evaluación que funcione bien podría ser la siguiente:

1. Decodificar el individuo para obtener los emplazamientos de los prototipos.
2. Asociar a cada ejemplo del conjunto de ejemplos con su prototipo más cercano.
3. Etiquetar a cada prototipo con la categoría a la que pertenecen la mayoría de los ejemplos a los que representa.
4. Producir como salida la suma, para todos los prototipos, del número de ejemplos que representa y que son de su misma categoría.

La selección de individuos se realiza mediante la anterior función de evaluación de forma que cada vez se selecciona a un individuo, siendo las selecciones sucesivas independientes unas de otras. Esto quiere decir que un mismo individuo puede ser seleccionado varias veces. Despues de aplicar el operador de reproducción se genera una población intermedia del mismo tamaño que la inicial, y que puede contener, generalmente, elementos repetidos.

Esta selección se utiliza para aplicar sobre ella los operadores de cruce, mediante los cuales se combinan las codificaciones de dos individuos para generar uno nuevo; y mutación, mediante el cual se cambia el valor de una de las posiciones de un individuo al azar. Estos operadores se aplican sucesivamente hasta que se obtiene una nueva generación de soluciones. El procedimiento se realiza una y otra vez hasta que se cumpla algún criterio de convergencia previamente especificado.

De esta forma los prototipos se van desplazando en la distribución. Cada desplazamiento hace que la nueva clasificación, producida por los prototipos utilizando la regla del vecino más próximo, clasifique más puntos correctamente. Ésta, como vemos, es una manera relativamente sencilla de ir colocando unos prototipos en un sitio adecuado para realizar una buena clasificación. Veamos, a continuación, algunos métodos más sofisticados.

16.4.2 Algoritmos evolutivos de estimación de distribuciones

Los algoritmos de estimación de distribuciones utilizan la misma representación de los algoritmos genéticos, pero su evolución es completamente diferente. Igual que el método anterior, se basan en una población de individuos, que representan un conjunto de

soluciones, que va modificándose de forma que los individuos de la población son cada vez mejores soluciones al problema, pero, en cambio, la forma de modificar y generar nuevas poblaciones es diferente.

A partir de una población de tamaño m se genera una población intermedia de tamaño n , donde $m > n$, seleccionando a los n mejores individuos de la población original, en términos de la función de evaluación elegida que puede ser, por ejemplo, la misma que hemos visto antes. Cada uno de los individuos es un vector $x = \{x_1, x_2, \dots, x_l\}$ que representa los atributos de los prototipos. Se desea expresar, en términos de funciones de probabilidad de cada una de las componentes x_i , las características de los individuos seleccionados. Estas distribuciones servirán de base para generar la nueva población, de forma que, al no generarse aleatoriamente, se espera que esta nueva población contenga mejores individuos [Larrañaga & Lozano 2002].

En este caso se asumirá que todos los atributos del individuo son independientes en lo que al cálculo de la función de evaluación se refiere. Esta asunción dista mucho de ser un reflejo de la realidad en problemas complejos, sin embargo existen variantes de estos algoritmos que permiten asumir cierto tipo de dependencias entre los atributos. Así pues, si los atributos son independientes se cumple que la probabilidad del individuo es igual al producto de las probabilidades de los atributos de sus prototipos.

$$p_t(x) = \prod_{i=1}^l p_t(x_i)$$

Bajo esta hipótesis habrá que determinar, o en su caso estimar las probabilidades anteriores a partir de la población intermedia generada. Según el procedimiento para estimar las probabilidades pueden plantearse diferentes métodos no muy distintos entre sí.

Algoritmo de distribución univariada marginal

Las probabilidades de cada atributo se calculan a partir de sus frecuencias marginales en el conjunto de la población intermedia:

$$p_{t+1}(x_{ik}) = \frac{\sum_{j=1}^n \delta_j(x_i, a_k)}{n}$$

Cada x_i podrá tomar valores dentro de un conjunto discreto de posibilidades, el alfabeto de alelos $A = \{a_1, a_2, \dots, a_o\}$ (antes hemos visto $A = \{0, 1\}$), por lo tanto $p_t(x_{ik})$ será la probabilidad de que el valor de x_i sea a_k , y:

$$\delta_j(x_i, a_k) = \begin{cases} 1 & \text{si en el individuo } j \text{ su } x_i = a_k \\ 0 & \text{en caso contrario} \end{cases}$$

Una vez calculadas las posibilidades, los valores de los atributos de los individuos de la nueva generación se generarán mediante ensayos de Bernoulli con dichas probabilidades.

Sea la siguiente representación. El alfabeto está constituido por tres símbolos $A = \{0, 1, *\}$, los individuos son de tamaño 6, y la función de evaluación se calcula según la regla:

- Los símbolos $*$ suman su posición menos 4.
- Los símbolos 1 suman 1 en posiciones pares y nada en posiciones impares.
- Los símbolos 0 suman 1 en posiciones impares y nada en posiciones pares.

Por ejemplo, si se desea evaluar el individuo $I=\{10^*1^*0\}$ habrá que comprobar qué símbolo contiene en cada una de sus posiciones y aplicar la regla anterior:

Pos.	1	0	*
1	-	1	-
2	-	-	-2
3	0	-	-
4	-	-	0
5	-	1	-
6	0	-	-

La evaluación final es la suma de todos los elementos de la tabla, cuyo resultado es cero.

Con este planteamiento se va a realizar una iteración de una población, con el algoritmo de distribución univariada marginal. Se genera una población inicial con 20 individuos, en los que cada símbolo en cada posición tiene la misma probabilidad de aparecer:

Ind.	x_1	x_2	x_3	x_4	x_5	x_6	$F(x)$
1	*	0	0	*	0	*	1
2	1	0	1	1	*	1	3
3	1	0	*	1	1	1	1
4	0	*	1	0	*	*	2
5	1	*	*	*	0	0	-2
6	*	0	0	*	1	*	0
7	*	1	*	0	0	1	-1
8	1	0	1	1	0	1	3
9	0	1	*	0	*	*	4
10	0	1	0	*	1	0	3
11	0	0	1	*	0	*	4
12	1	1	*	1	1	*	3
13	*	*	1	0	0	1	-3
14	1	0	0	*	1	*	3
15	*	*	*	0	*	0	-5
16	0	1	1	*	0	0	3
17	*	*	0	1	1	0	-3
18	0	1	*	0	1	*	4
19	1	0	1	*	*	0	1
20	0	1	0	1	*	*	7

En la última columna aparece el resultado de la evaluación de cada uno de los individuos. A partir de esta población, se genera la población intermedia de tamaño 10, conteniendo los 10 mejores individuos de la población inicial:

Ind.	x_1	x_2	x_3	x_4	x_5	x_6	$F(x)$
2	1	0	1	1	*	1	3
8	1	0	1	1	0	1	3
9	0	1	*	0	*	*	4
10	0	1	0	*	1	0	3
11	0	0	1	*	0	*	4
12	1	1	*	1	1	*	3
14	1	0	0	*	1	*	3
16	0	1	1	*	0	0	3
18	0	1	*	0	1	*	4
20	0	1	0	1	*	*	7

A partir de la población intermedia generada se calculan las probabilidades. Por ejemplo la probabilidad de que el tercer atributo contenga el símbolo asterisco, $p(x_3^*)$, se calculará

como el número de asteriscos de la columna 3 de la tabla anterior (hay tres asteriscos), entre el tamaño de la población intermedia n , en este caso 10. La tabla completa de probabilidades así calculadas será:

	x_1	x_2	x_3	x_4	x_5	x_6
1	0,4	0,6	0,4	0,4	0,4	0,2
0	0,6	0,4	0,3	0,2	0,3	0,2
*	0	0	0,3	0,4	0,3	0,6

Esta es la tabla que se utilizará para la generación de los individuos de la población en el siguiente intervalo de tiempo. La población resultante podría ser:

Ind.	x_1	x_2	x_3	x_4	x_5	x_6	$F(x)$
1	0	1	0	*	0	*	6
2	0	1	1	1	*	1	5
3	0	1	*	1	1	*	3
4	1	1	1	0	*	*	4
5	0	0	*	*	0	0	1
6	1	1	0	1	1	*	5
7	0	0	*	0	*	1	2
8	0	1	1	1	0	*	6
9	1	0	*	*	*	*	2
10	1	0	0	*	1	0	1
11	1	1	1	*	0	*	4
12	0	0	*	1	1	*	3
13	0	1	1	0	0	1	4
14	0	1	0	*	1	*	5
15	0	1	1	1	*	0	4
16	1	0	1	*	0	*	3
17	0	1	0	1	1	0	4
18	1	0	*	0	1	*	1
19	0	1	1	*	*	1	4
20	1	0	0	1	1	*	4

A esta población se le repite el proceso anterior, se seleccionan los 10 mejores individuos:

Ind.	x_1	x_2	x_3	x_4	x_5	x_6	$F(x)$
1	0	1	0	*	0	*	6
2	0	1	1	1	*	1	5
4	1	1	1	0	*	*	4
6	1	1	0	1	1	*	5
8	0	1	1	1	0	*	6
13	0	1	1	0	0	1	4
14	0	1	0	*	1	*	5
15	0	1	1	1	*	0	4
17	0	1	0	1	1	0	4
20	1	0	0	1	1	*	4

Y se calculan de nuevo las probabilidades:

	x_1	x_2	x_3	x_4	x_5	x_6
1	0,3	0,9	0,5	0,6	0,4	0,2
0	0,7	0,1	0,5	0,2	0,3	0,2
*	0	0	0	0,2	0,3	0,6

Se puede apreciar cómo estas probabilidades se aproximan a la óptima. En los tres primeros atributos, que son para los que el asterisco descuenta valor en la función de

evaluación, la probabilidad del asterisco es cero, y es alta (0,6) en el último atributo donde suma dos. Además en los atributos pares la probabilidad de cero es muy alta, como lo es la del uno en los atributos impares. El método será capaz de encontrar la mejor solución tras sólo un número pequeño de iteraciones.

Aprendizaje incremental basado en poblaciones

Este método necesita de una representación binaria, es decir $A=\{0,1\}$. En cada generación la población de individuos viene representada por un vector de probabilidades:

$$p_t(x) = \{p_t(x_1), p_t(x_2), \dots, p_t(x_l)\}$$

donde $p_t(x_i)$ es la probabilidad de que el atributo i en la generación t contenga el carácter 1. El algoritmo es el siguiente:

1. Se genera la población utilizando el vector de probabilidades $p_t(x)$, como en el método anterior.
2. Se seleccionan los n mejores individuos de la población, $xm=\{xm_1, xm_2, \dots, xm_n\}$, donde cada xm_i es un vector de l componentes: $xm_i=\{xm_{i1}, xm_{i2}, \dots, xm_{il}\}$.
3. Este conjunto de mejores individuos se utiliza para actualizar las probabilidades de los atributos, siguiendo la ecuación:

$$p_{t+1}(x_i) = (1-\alpha)p_t(x_i) + \alpha \frac{1}{n} \sum_{k=1}^n xm_{ki}$$

donde $\alpha \in (0,1]$ es un parámetro del método. Cuando $\alpha=1$ los dos métodos anteriores coinciden.

Para más información se puede consultar [Larrañaga & Lozano 2002].

Algoritmo genético compacto

En este caso la representación es también binaria, pero no existe una población de individuos-solución. Se genera el vector de probabilidades, haciendo equiprobables los dos caracteres de todos los atributos: $p_0(x) = \{0,5, 0,5, \dots, 0,5\}$. El vector de probabilidades se utiliza para generar dos individuos. Los dos individuos se evalúan y se dice que entran en competición. Se elige el mejor de los dos, y se actualiza el vector de probabilidades, incrementando aquellos atributos en los que ambos tengan diferentes valores, y el valor del ganador sea 1; y decrementando aquellas en las que, siendo también diferente, tenga un 0. Más concretamente el procedimiento sería:

1. Inicializar el vector de probabilidades como se ha indicado.
2. Generar, mediante dicho vector, dos individuos: x^1 y x^2 .
3. Evaluar ambos individuos. Sea x^1 el mejor.
4. Actualizar el vector de probabilidades de la siguiente manera:

$$p_{t+1}(x_i) = \begin{cases} p_t(x_i) + \frac{1}{K} & \text{si } x_i^1 = 1 \text{ y } x_i^2 = 0 \\ p_t(x_i) - \frac{1}{K} & \text{si } x_i^1 = 0 \text{ y } x_i^2 = 1 \end{cases}$$

5. Si el vector de probabilidades no ha convergido, tiene valores que aún están lejanos al 1 o al 0, ir a 2.
6. El vector $p_t(x)$ representa la solución final.

Todos los métodos anteriores suponen una mejora sustancial en los resultados de clasificación sobre otros métodos basados en vecindad. Para más información se puede consultar [Larrañaga & Lozano 2002].

16.5 Otros métodos y aplicabilidad

En este capítulo se han visto una serie de métodos basados en vecindad. Muchos de estos métodos han ido mejorándose desde su aparición, y existen así variantes que, frecuentemente, se conocen por su encarnación en algoritmos.

Así, de los métodos de agrupamiento que se suelen denominar basados en partición (como el K medias), tenemos los k-mediodes [Kaufman & Rousseeuw 1990], con variantes PAM, CLARA [Kaufman & Rousseeuw 1990] y CLARANS [Ng & Han 1994]. Una variante eficiente para grandes volúmenes de datos que utiliza distintas medidas de distancia es BIRCH [Zhang et al. 1996]. Otro método de este tipo, en este caso basado en densidad, es el algoritmo DBSCAN (*Density Based Spatial Clustering of Applications with Noise*) [Ester et al. 1996].

Del mismo modo, existen otros métodos de clasificación también basados en vecindad, o en la idea de que instancias similares han de tener clases similares, como por ejemplo K* [Cleary & Trigg 1995] o PEBLS [Cost & Salzberg 1993]. Un aspecto no tratado aquí es el hecho de que muchos de los métodos de clasificación vistos aquí se pueden adaptar para regresión.

Finalmente, algunos métodos no paramétricos vistos en el Capítulo 8, en especial aquellos que utilizan una función de localidad, como la regresión lineal local, pueden considerarse métodos retardados y basados en distancias, como los que hemos visto en este capítulo. De hecho, el Capítulo 8 de [Mitchell 1997], lo incluye dentro de los métodos de aprendizaje basados en instancias.

Antes de pasar a comentar las herramientas *software* disponibles para utilizar los métodos comentados en este capítulo, es importante destacar la relación existente entre los métodos basados en instancias (*instance-based learning*), los métodos basados en vecindad y un área de la inteligencia artificial que generaliza, en cierto modo, estas aproximaciones, denominada razonamiento basado en casos.

16.5.1 Razonamiento basado en casos

Los métodos basados en instancias del tipo k vecinos descritos anteriormente, comparten tres propiedades esenciales. Son métodos retardados, clasifican los nuevos ejemplos mediante el análisis de ejemplos similares, descartando completamente los ejemplos que no se les parecen, y representan los ejemplos como vectores de números reales, que son puntos en un espacio n-dimensional. El razonamiento basado en casos (*case-based reasoning*) [Sycara et al. 1992] posee las dos primeras propiedades, pero no la tercera. Los ejemplos se representan utilizando descripciones simbólicas mucho más sofisticadas, y por lo tanto deben manejar reglas de vecindario o funciones de distancia más complejas.

El razonamiento basado en casos es un modelo computacional de razonamiento por analogía, basado en casos históricos existentes. Una de las premisas del razonamiento basado en casos es que gran parte de las soluciones a problemas encontradas por los

especialistas no son originales, son variaciones sobre un problema tipo. A veces es mejor resolver un problema partiendo de la solución existente a un problema similar, que tratar de resolverlo desde el principio. Esto es lo que se llama razonamiento por analogía de casos previos. La forma de afrontar el problema es buscar un caso similar en el pasado, junto con la solución que funcionó para ese caso. Adaptar la solución para ajustar las diferencias existentes entre los dos casos y almacenar la solución sugerida, para que pueda ser utilizada posteriormente en nuevos casos [Aamodt & Plaza 1994; Kolodner 1993].

Generalmente un sistema de razonamiento basado en casos consiste en:

- Una base de datos de ejemplos anteriores y sus soluciones.
- Un conjunto de índices para poder recuperar casos pasados y almacenar los nuevos.
- Un conjunto de reglas para medir las similitudes entre casos.
- Y una regla de modificación de soluciones existentes, en función de las diferencias entre los casos.

La modificación de las soluciones se realiza mediante la obtención de los valores de los atributos del ejemplo a procesar, que identifican el tipo de problema, y que diferencian un problema de otro. Estos atributos se utilizan como índices para la distinción de casos análogos y el almacenamiento de los nuevos. Los índices y reglas para medir la similitud de los casos se centran en las características importantes del problema, por ejemplo aquellas características que pueden explicar por qué las soluciones de diferentes casos son distintas.

Una vez que el sistema “comprende” el problema, se buscan casos análogos, con el menor número de diferencias posibles, medidas en los términos anteriores. Los casos encontrados generalmente tendrán pequeñas diferencias que harán que la solución que estos casos similares proponen sea inaplicable al nuevo caso que se quiere resolver. Será necesario, pues, adaptar y combinar las soluciones propuestas por los casos similares, de forma que la solución obtenida después de la adaptación pueda ser aplicable al caso en cuestión. Las reglas de adaptación de las soluciones deberán incluir conocimiento del dominio del problema acerca del impacto de los valores de los atributos en la solución.

Los sistemas de razonamiento basado en casos tienen un gran número de ventajas: en los problemas en los que no existe un gran conocimiento del dominio, o donde la relación entre los atributos de los casos y la solución no está suficientemente clara como para que pueda ser representada en forma de reglas, o cuando la relación de casos que son “excepción a la regla” es alta; los sistemas de razonamiento basado en casos son buenos candidatos, ya que pueden hacer modelos de las excepciones y de los casos nuevos.

Los sistemas de razonamiento basado en casos son también útiles a la hora de explicar o justificar una solución. Cuando la teoría del dominio es débil, es bastante difícil justificar o explicar una postura de forma razonada. Sin embargo el poder dar una analogía o encontrar un antecedente puede ser más eficaz que un argumento basado en un modelo. Los usuarios de sistemas basados en casos pueden encontrar más instructivo poder ver ejemplos de casos pasados que ver reglas sofisticadas de razonamiento.

Estos sistemas tienen también inconvenientes. Si los casos encontrados, o existentes en la base de datos no tienen la suficiente similitud con el caso a procesar, la solución a adaptar puede ser inapropiada. Esto puede darse por limitaciones en la base de datos, que no tenga un número suficiente de casos significativos, o por la ineficacia de las reglas de

similitud utilizadas. Estos sistemas tienen también problemas de reconocimiento de nuevos problemas tipo. Esto ocurre cuando un nuevo caso se distingue de casos anteriores por atributos que no están representados en los índices. En este caso los sistemas de razonamiento basado en casos son incapaces de reconocer la diferencia.

Para obtener más información de el razonamiento basado en casos se puede consultar (<http://www.ai-cbr.org> o <http://www.cbr-web.org>).

Ejemplo de recuperación de la información

A continuación se incluye un ejemplo de razonamiento basado en casos para recuperación de documentos. Los documentos van a ser, por tanto, los ejemplos, instancias o casos para trabajar. El problema de la recuperación de la información trata básicamente de encontrar información relevante a petición del usuario. En este caso la información a recuperar, como se ha mencionado, son documentos que vienen definidos por sus resúmenes. La tarea en concreto viene definida por:

- La representación del contenido de los documentos.
- La representación de la información relativa a las solicitudes de los usuarios. Esto es la representación de la formulación de las preguntas.
- La comparación de dichas representaciones, de forma que permita decidir qué documentos deben ser recuperados.

La representación de los documentos se realiza mediante la asociación de un conjunto de atributos con el documento, cuyos valores se asume describen los contenidos de los documentos. La elección de los documentos depende de la similitud entre los atributos de dichos documentos y los atributos de la formulación de la pregunta por parte del usuario. La medida de similitud se realiza mediante la equiparación de los valores de los atributos, buscando un grado, suficientemente alto, de coincidencia entre los conjuntos de atributos que relacionan la pregunta y los documentos (se puede utilizar una función de distancia basada en conjuntos, como se presentó al principio del capítulo).

Una solución, siguiendo un esquema de razonamiento basado en casos podría ser la siguiente [Croft 1993]: en primer lugar, se construye una base de datos de casos; cada caso representa una pregunta, una transformación de la misma, y la respuesta producida, como solución del caso. Esta respuesta podría ser la lista de documentos generada, en la que además cada documento está asociado a un vector de atributos por los que ha sido seleccionado. Además se construye una pregunta-caso que estará formada por un conjunto de atributos representativos de la pregunta del usuario, hay también un contexto-caso en el que se representará el contexto de la pregunta, con atributos como por ejemplo: temas relacionados, objetivo de la información, utilización de la información, términos clave relacionados, etc. Éstos pueden ser obtenidos mediante el estudio estadístico de las preguntas de la base de datos de preguntas.

Con toda esta información se construye un caso, que será el objeto de la búsqueda de una solución, que consistirá en el conjunto de documentos y los atributos relevantes de la búsqueda. El sistema buscará un caso similar en la base de datos de casos. Si existe una equiparación lo suficientemente buena, se producirá la lista de documentos del caso encontrado. Si no existe una equiparación satisfactoria, lo cual ocurrirá en la mayoría de los casos, se realizarán una serie de modificaciones en la pregunta, en función a las diferencias

encontradas entre el caso a clasificar, y los más cercanos a él en la base de datos. Una vez que las modificaciones a la pregunta permitan equipararla con algunos casos de la base de datos (el número mínimo de casos que tienen que equipararse con la pregunta, para considerarla correcta, es un parámetro del sistema, pero suele ser entre uno y cuatro [Salton & McGill 1983]), se comparan las soluciones de los casos y se construye una nueva solución combinando las semejanzas de las soluciones equiparadas con las modificaciones realizadas en la pregunta. Esto producirá una nueva solución al caso presentado, basada en soluciones próximas en la base de datos. Además, este nuevo caso, junto con la solución propuesta, se incluye en la base de datos.

En algunos sistemas interactivos es posible utilizar al usuario como parte del proceso de aprendizaje. En este caso existen dos interacciones interesantes, pero no obligatorias, con el usuario. En primer lugar, después de la modificación de la pregunta, se puede solicitar al usuario si la nueva pregunta es adecuada a sus pretensiones, si está de acuerdo con continuar con el proceso, y, si no lo está, permitirle realizar las modificaciones que sean pertinentes en la pregunta, y continuar con el proceso. Otro punto de interacción es solicitar al usuario si la lista de documentos producida por el sistema satisface sus expectativas, y sólo si es así introducir el nuevo caso en la base de datos.

Esto es solamente un ejemplo de cómo el razonamiento basado en casos se puede aplicar a problemas cuya formulación es difícil bajo la estricta representación como tablas de pares atributo-valor. En el Capítulo 21 se verán más aproximaciones para recuperación de documentos.

16.5.2 Sistemas

Muchos sistemas genéricos de minería de datos incluyen varios de los métodos comentados en este capítulo, en especial los basados en modelo (los no retardados): K medias, mapas de Kohonen, LVQ, etc. En cambio, los métodos retardados, como los vecinos más próximos o el agrupamiento jerárquico, no son tan habituales.

Quizás el sistema que dispone de más métodos basados en instancias es el WEKA (véase Apéndice A). WEKA dispone de los comentados anteriormente, así como el IB1 y IBk (para vecinos más próximos), el kstar (que es el K*) o el Cobweb.

El paquete estadístico gratuito R (<http://www.r-project.org>), dispone de redes SOM (en modo *batch* y *online*), vecinos más próximos, LVQ, algoritmos de agrupamiento jerárquicos y no jerárquicos como PAM, CLARA o DIANA [Kaufman & Rousseeuw 1990].

Possiblemente el mayor problema de las implementaciones de estos métodos en los paquetes de minería de datos es que no explotan su mejor característica, la posibilidad de variar la función de distancia. Muchas implementaciones incorporan una medida de distancia (generalmente la euclídea) y no permiten variarla fácilmente. En este sentido, las librerías, como MLC++, Xelopes (véase Apéndice A) o el mismo WEKA, pueden permitir, modificando el código, cambiar la función de distancia.

PARTE IV

EVALUACIÓN, DIFUSIÓN Y

USO DE MODELOS

En esta parte se aborda la última fase del proceso de extracción de conocimiento, la evaluación, combinación, interpretación, difusión y uso de los modelos extraídos. Las técnicas presentadas en esta parte permiten sacar el verdadero provecho y consolidar los modelos obtenidos en la fase anterior.

CAPÍTULOS

- 17. Técnicas de evaluación**
- 18. Combinación de modelos**
- 19. Interpretación, difusión y uso de modelos**

Capítulo 17

TÉCNICAS DE EVALUACIÓN

En los capítulos anteriores hemos presentado varias de las técnicas más utilizadas en la minería de datos. Estas técnicas permiten la generación de modelos a partir de una evidencia formada por un conjunto de datos. Llegados a este punto, cabe preguntarse cómo podemos saber si un determinado modelo es lo suficientemente válido para nuestros propósitos. Esto, en algunos casos concretos, puede determinarse mediante un análisis de hipótesis nula, clásico en estadística (véase por ejemplo [Peña 2001; Peña 2002a]), como se ha hecho frecuentemente en los capítulos 7 y 8. Otras veces, sin embargo, o no es posible o lo que queremos es seleccionar entre varios modelos o estimar su fiabilidad. Para ello, existen varios métodos que sirven para evaluar la calidad de un modelo a partir de la evidencia. En este capítulo vamos a presentar las técnicas más utilizadas para la evaluación de modelos.

17.1 Introducción

Los métodos de aprendizaje permiten construir modelos o hipótesis a partir de un conjunto de datos, o evidencia. En la mayoría de los casos es necesario evaluar la calidad de las hipótesis de la manera más exacta posible. Por ejemplo, si en el ámbito de aplicación de un modelo un error en la predicción conlleva importantes consecuencias (por ejemplo detección de células cancerígenas), es importante conocer con exactitud el nivel de precisión de los modelos aprendidos.

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

conjunto de entrenamiento, por lo que favorecen los modelos que sobreajustan el conjunto de datos de entrenamiento y no generalizan para otros datos.

Consecuentemente, una mejor opción es evaluar los modelos sobre un conjunto de datos diferente al conjunto de entrenamiento. En las siguientes secciones abordamos diferentes técnicas de evaluación basadas en esta partición de los datos de una evidencia en dos partes, una para el aprendizaje (entrenamiento), y otra para la evaluación (test).

Otra aproximación, más realista por lo general, es la evaluación basada en costes. En este tipo de evaluación se evalúa el coste de los errores cometidos por un modelo. En este contexto, el mejor modelo es el modelo que comete errores con menor coste asociado, no el modelo que cometa menor número de errores.

Por otra parte, los modelos de regresión no pueden evaluarse comparando si la clase predicha es igual, o no, a la clase real, ya que la clase es de tipo numérico. En estos casos, se utiliza la distancia entre ambos valores, ya sea real o cuadrática. El mejor modelo es, entonces, el modelo que minimice la distancia media entre ambos valores de los puntos utilizados para la evaluación.

Finalmente, la evaluación de los modelos descriptivos, como por ejemplo, los modelos de agrupamiento, es bastante más complicada. Esto se debe fundamentalmente a la ausencia de una clase donde medir el grado de acierto de un modelo. Es por ello que las medidas de evaluación de modelos descriptivos se basan en conceptos tales como la complejidad del modelo y de los datos a partir del modelo, o bien, en agrupamiento, el nivel de compactación de los diferentes grupos.

17.2 Evaluación de clasificadores

El objetivo del aprendizaje automático supervisado es el aprendizaje de una función objetivo f , considerando un espacio de posibles hipótesis H [Mitchell 1997]. Para ello, los algoritmos de aprendizaje emplean normalmente una evidencia o muestra S formada por ejemplos de la función objetivo f de acuerdo con una distribución D . Esta distribución define la probabilidad de encontrar cada instancia de la evidencia en el espacio de todas las posibles evidencias (véase Figura 17.1).

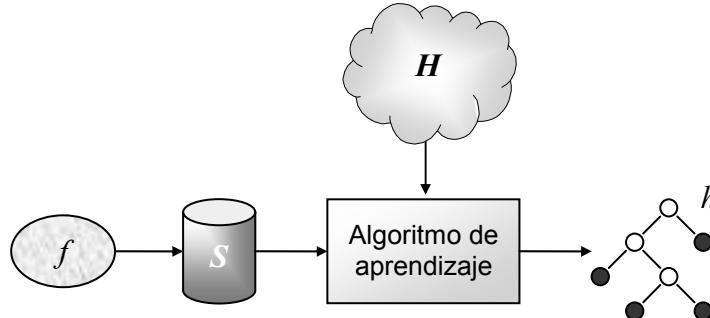


Figura 17.1. Aprendizaje de una hipótesis.

Dado que es posible inducir un conjunto bastante grande y variado de hipótesis desde una única evidencia mediante la utilización de diversas técnicas de aprendizaje, es necesario establecer medidas de calidad para evaluar las hipótesis. Estas medidas deben reflejar la

similaridad semántica de la hipótesis con respecto a la función objetivo f . En el caso del problema de la clasificación, la función f tiene como dominio una categoría (la clase).

17.2.1 Evaluación de hipótesis basada en precisión

Frecuentemente, las medidas están basadas en la precisión de la hipótesis, o, de manera equivalente, en el porcentaje de error de la hipótesis con respecto a la función f . Este porcentaje de error puede calcularse utilizando los datos que se poseen de f (la evidencia). En este caso, al porcentaje de error se le llama *error de muestra*. Otra opción es utilizar la distribución D para calcular el *error verdadero*.

Formalmente, el error de muestra ($error_s(h)$) de una hipótesis h con respecto a la función objetivo f y una muestra S es:

$$error_s(h) = \frac{1}{n} \sum_{x \in S} \delta(f(x) \neq h(x))$$

donde n es el número de componentes de S , $\delta(\text{verdadero})=1$, y $\delta(\text{falso})=0$.

El error verdadero ($error_t(h)$) de una hipótesis h con respecto a la función objetivo f y una distribución D , es la probabilidad de que h prediga un resultado diferente a f en un ejemplo tomado de acuerdo con la distribución D :

$$error_t(h) = Pr_{x \in D} [(f(x) \neq h(x))]$$

donde $Pr_{x \in D}$ denota que x se toma siguiendo la distribución de probabilidad D .

El error verdadero es lo que desearíamos conocer sobre una determinada hipótesis. Sin embargo, el error de muestra es todo lo que se puede conocer acerca de la precisión de una hipótesis, dado que, generalmente, se desconoce la función objetivo f . La única muestra del comportamiento de f se obtiene en la evidencia S .

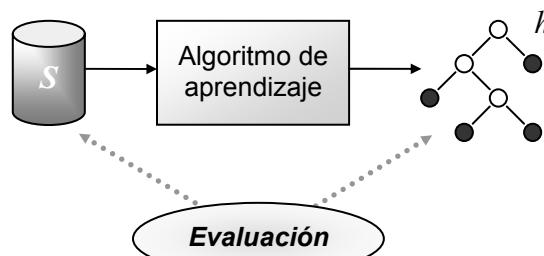


Figura 17.2. Evaluación de una hipótesis.

Una primera aproximación para evaluar hipótesis (que se muestra en la Figura 17.2) consiste en utilizar la evidencia completa para el aprendizaje y entonces emplear la misma evidencia para calcular el error de muestra de las hipótesis.



Figura 17.3. Subajuste y sobreajuste.

Este simple mecanismo es bastante problemático, ya que favorece claramente a las hipótesis que sobreajustan (*overfitting*) la evidencia, es decir, las hipótesis que se centran

demasiado en la evidencia. Por otra parte, si se intenta corregir sin ninguna referencia exterior (por ejemplo, podando demasiado un árbol de decisión) existe el problema del subajuste (*underfitting*), en este caso las hipótesis generalizan demasiado dejando gran cantidad de datos como excepciones. En la Figura 17.3 representamos gráficamente ambas situaciones. En la Sección 6.4.1 ya se comentaron los problemas de sobreajuste y subajuste. ¿Existe alguna manera de aminorar el problema o al menos detectarlo de una manera objetiva? Veamos que sí es posible.

Un mecanismo más correcto para calcular el error de muestra (*error(h)*) de una hipótesis con respecto a una evidencia, consiste en separar el conjunto de datos que forma la evidencia en dos subconjuntos disjuntos como se muestra en la Figura 17.4. El primer subconjunto, denominado de **entrenamiento** (*training*), se utiliza para el aprendizaje de las hipótesis. Al segundo subconjunto, denominado conjunto de **prueba** (*test*), y sólo se emplea, en este caso, para calcular el error de muestra de las hipótesis construidas con los datos de entrenamiento. Habitualmente, la partición se realiza de manera aleatoria, dejando el 50 por ciento de los datos para el entrenamiento y el 50% restante, para el subconjunto de prueba. Los porcentajes suelen variar (80 por ciento - 20 por ciento, 75 por ciento - 25 por ciento, etc.).

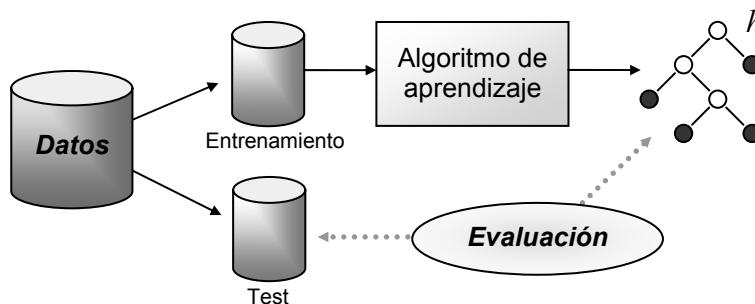


Figura 17.4. Evaluación de una hipótesis utilizando dos conjuntos de datos (entrenamiento/test).

El hecho de utilizar dos conjuntos de datos independientes, uno para aprender la hipótesis y el otro para evaluarla, permite resolver el problema de premiar el sobreajuste en la evaluación de hipótesis. De hecho, podemos definir el sobreajuste como la situación en la que *el modelo da mucho mejores resultados para el conjunto de entrenamiento que para el conjunto de test*.

Sin embargo, existen todavía importantes problemas, ya que el resultado es demasiado dependiente del modo en el cual se ha realizado la partición en dos subconjuntos de la evidencia completa. Dado que, normalmente, esta partición se realiza de manera aleatoria, puede ocurrir que dos experimentos realizados sobre la misma evidencia y con el mismo método de aprendizaje, obtengan resultados muy dispares. Otro problema es que muchas veces tenemos pocos datos y reservar parte de ellos para el test puede hacer que todavía podamos utilizar menos para el entrenamiento, obteniendo peores modelos.

Evaluación mediante validación cruzada

Un mecanismo que permite reducir la dependencia del resultado del experimento en el modo en el cual se realiza la partición, es utilizar validación cruzada (*cross-validation*). Este método, ver Figura 17.5, consiste en dividir el conjunto de la evidencia en k subconjuntos disjuntos de similar tamaño. Entonces, se aprende una hipótesis utilizando el conjunto

formado por la unión de $k-1$ subconjuntos y el subconjunto restante se emplea para calcular un error de muestra parcial. Este procedimiento se repite k veces, utilizando siempre un subconjunto diferente para estimar el error de muestra parcial. El error de muestra final se calcula como la media aritmética de los k errores de muestra parciales. De esta manera, el resultado final recoge la media de los experimentos con k subconjuntos de prueba independientes.

Otra ventaja de la validación cruzada es que la varianza de los k errores de muestra parciales, permite estimar la variabilidad del método de aprendizaje con respecto a la evidencia. Comúnmente, se suelen utilizar 10 particiones (10-fold cross validation).

Como hemos indicado, una de las ventajas de esta técnica es que los k subconjuntos de prueba son independientes. No obstante, esto no sucede en los conjuntos de entrenamiento. Por ejemplo, en una validación cruzada con $k=10$, cada par de subconjuntos de entrenamiento comparten el 80 por ciento de los datos. Algunos autores [Dietterich 1998] defienden que este solapamiento entre los subconjuntos de entrenamiento podría afectar la calidad de la estimación, y proponen una modificación en la técnica que permita utilizar subconjuntos de entrenamiento independientes. Concretamente, en [Dietterich 1998] se introduce la validación cruzada 5×2 (5×2 cross validation). Esta técnica consiste en aplicar cinco repeticiones de una validación cruzada con $k=2$. En cada una de las cinco iteraciones el conjunto de datos se divide en dos subconjuntos disjuntos (entrenamiento y prueba) de idéntico tamaño. El error de muestra final se calcula como la media de los cinco errores de muestra parciales.

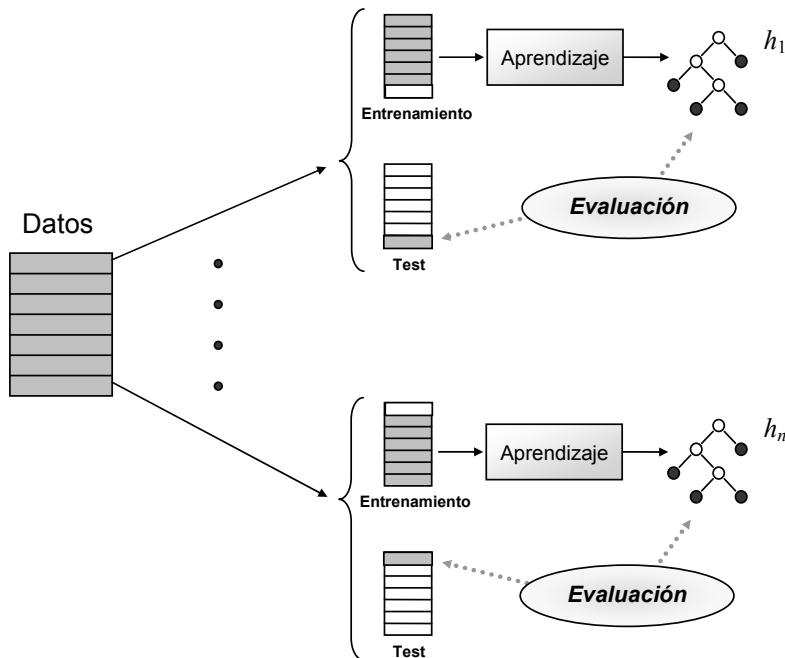


Figura 17.5. Evaluación mediante validación cruzada.

El trabajo [Dietterich 1998] compara varios métodos de evaluación mediante varios experimentos con datos reales y simulados. La recomendación final de este trabajo es utilizar la validación cruzada 5×2 cuando las técnicas de aprendizaje son lo suficientemente

eficientes para ejecutarse diez veces, o utilizar la clásica partición de los datos en entrenamiento/prueba en el otro caso.

Evaluación por *bootstrap*

La evaluación por *bootstrap* se utiliza en casos en los que todavía se tienen menos ejemplos y está especialmente indicada en estos casos. La idea es en cierto modo similar a la validación cruzada, aunque la forma de proceder es diferente.

Supongamos que tenemos N ejemplos. A partir de este conjunto realizamos un muestreo aleatorio con reposición de N ejemplos. Esta muestra, que será el conjunto de entrenamiento, al ser con reemplazamiento, puede contener ejemplos repetidos (que se mantienen). Lógicamente, esto significa que no contendrá algunos ejemplos del conjunto original. Precisamente, los ejemplos no elegidos por la muestra se reservan para el conjunto de test. Esto nos da un conjunto de entrenamiento de N ejemplos y un conjunto de test de aproximadamente $0,368 \times N$ ejemplos (ahora veremos por qué esta cantidad). Con estos conjuntos se entrena y evalúa un modelo.

El proceso anterior se repite un número prefijado k de veces (digamos diez veces) y después se actúa como en el caso de la validación cruzada, promediando los errores/precisiones. Quizá lo interesante de este proceso es que las k repeticiones del proceso son independientes y esto es más robusto estadísticamente.

Respecto a los tamaños, este valor del 0,368 se obtiene simplemente calculando la probabilidad de que un ejemplo no salga en una extracción (que es $1-1/n$) y multiplicando este número las veces que se realiza la extracción, es decir n . Más formalmente, tenemos:

$$\lim_{n \rightarrow \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0,368$$

De hecho, esta cantidad se utiliza a veces para obtener una variante del error estimado, que se calcula de la siguiente manera:

$$Error_{ESTIMADO} = 0,368 \times Error_{ENTRENAMIENTO} + 0,632 \times Error_{TEST}$$

Intervalos de confianza de la evaluación

Por último, nos podemos preguntar si el valor de precisión estimado (mediante entrenamiento-test o validación cruzada) es fiable. Dada una muestra S de n ejemplos tomada a partir de una función objetivo f con una distribución D , es posible establecer unos intervalos de confianza para el error verdadero ($error_v(h)$) de una hipótesis h a partir del error de muestra ($error_s(h)$). Aunque la distribución que deberíamos utilizar es la binomial, para valores de n mayores de 30 se puede utilizar la distribución normal. Por tanto, el intervalo del error, a un nivel de confianza $c\%$, puede calcularse como:

$$error_s(h) \pm z_c \sqrt{\frac{error_s(h)(1 - error_s(h))}{n}}$$

Donde la constante z se establece el partir del nivel de confianza según la siguiente tabla de la normal, o en cualquier paquete estadístico u hoja de cálculo:

Nivel de confianza $c\%$	50%	80%	90%	95%	99%
Constante z_c	0,67	1,28	1,64	1,96	2,58

Por ejemplo, consideremos que una hipótesis da 12 errores sobre 40 ejemplos, tenemos un $error_s(h) = 0,30$. Tenemos, por tanto, que con confianza 95 por ciento ($z = 1,96$), el intervalo del error será: $0,30 \pm 0,14$, lo que quiere decir que, para el 95 por ciento de otras muestras de 40 ejemplos que probáramos, el error estaría dentro de ese intervalo.

17.2.2 Evaluación de hipótesis basada en coste

En los puntos anteriores hemos presentado varias formas de evaluar hipótesis basadas en el porcentaje de error que cometían, sin distinguir entre los errores cometidos. En muchos casos reales ocurre, sin embargo, que diferentes errores tienen costes muy dispares. Considerese, por ejemplo, un sistema de aprendizaje cuya labor es determinar si un correo entrante se puede considerar *spam*, y en ese caso eliminarlo. Evidentemente, es mucho más desfavorable que el sistema considere como basura un correo válido, que la situación inversa, que un correo de publicidad pase el filtro y llegue al destinatario. Este tipo de situaciones es bastante común en aplicaciones de sistemas de aprendizaje, como por ejemplo, diagnóstico de enfermedades, detección de alarmas, campañas de marketing... De hecho, es bastante usual el caso en el que los errores de clasificar ejemplos de clases minoritarias en la clase mayoritaria (por ejemplo, predecir que un sistema es seguro, cuando en realidad no lo es) tiene asociado un coste mayor, que la situación inversa (predecir que un sistema es inseguro, cuando en realidad lo es). Obviamente, los costes de cada error dependen del problema, pero, en cualquier caso, es bastante excepcional que todos los costes sean uniformes para un determinado problema. Por lo tanto, la precisión no es, generalmente, la mejor medida para evaluar la calidad de un determinado modelo, o un determinado algoritmo de aprendizaje.

El aprendizaje sensible al coste puede considerarse como una generalización más realista del aprendizaje predictivo. En este contexto, la calidad de un determinado modelo se mide en términos de minimización de costes, en vez de en minimización de errores.

Considerese el siguiente ejemplo (ya introducido en la Sección 2.5.3): la sala de urgencias de un hospital desea mejorar sus criterios de admisión utilizando técnicas de minería de datos. Cuando un nuevo paciente llega a la sala, tras unos primeros cuidados y exámenes médicos, se le asigna uno de los tres posibles destinos dependiendo de las condiciones del paciente: ingreso en la Unidad de Cuidados Intensivos (UCI), ingreso en observación, o no admisión (salida con alta). La probabilidad de cada uno de los casos es 0,5 por ciento, 13 por ciento, 86,5 por ciento, respectivamente. Si se recopilan datos precedentes de pacientes y con ellos se aprende un modelo de predicción mediante alguna de las técnicas de aprendizaje automático, un modelo que siempre enviase a casa un paciente obtendría una precisión del 86,5 por ciento de los casos. Este modelo, desde el punto de vista de la precisión, podría considerarse como válido, sin embargo, este modelo sería inútil y muy peligroso, dado que se enviaría a casa pacientes en estado grave.

En este caso, sería más conveniente utilizar las técnicas de aprendizaje que obtienen modelos que minimizan los costes de aplicación. Considerese que se conocen, aproximadamente, los costes de cada clasificación errónea. La manera más habitual de expresar estos costes en problemas de clasificación es mediante la denominada **matriz de costes**. En esta matriz se expresan los costes de todas las posibles combinaciones que se pueden dar entre

la clase predicha y la real. Por lo tanto, para un problema de c clasificadores, la matriz debe ser de tamaño $c \times c$.

La siguiente matriz de costes es un ejemplo para el problema de la sala de urgencias descrito anteriormente:

Estimado	Real		
	Salida	Observación	UCI
Salida	0 €	5.000 €	500.000 €
Observación	300 €	0 €	50.000 €
UCI	800 €	500 €	0 €

Esta matriz se interpreta de la siguiente manera: si el sistema predice que un paciente debe ir a la UCI, cuando en realidad no necesita tratamiento tiene un coste de 800 euros. Nótese que todos los costes en la diagonal (es decir los aciertos) tienen valor 0 (también se podrían expresar costes negativos o beneficios en estas casillas).

Es bastante sencillo estimar el coste de un clasificador para un determinado conjunto de ejemplos si se dispone de la matriz de costes del problema. Para ello se utiliza la **matriz de confusión** del clasificador para ese conjunto de datos. La matriz de confusión es una manera detallada de informar cómo distribuye los errores un determinado clasificador. Por ejemplo, una posible matriz de confusión para una evidencia formada por 100 casos (conjunto de test) en el problema de la sala de urgencias sería:

Estimado	Real		
	Salida	Observación	UCI
Salida	71	3	1
Observación	8	7	1
UCI	4	2	3

Cada celda de la matriz de confusión enumera, para cada clase real, el número de casos en los cuales el clasificador ha predicho una clase. Por ejemplo, este clasificador ha enviado dos pacientes a la UCI, de los que deberían haberse enviado a observación. A partir de una matriz de confusión se puede estimar la precisión de un clasificador directamente, dividiendo el número de aciertos (casos en la diagonal) entre el número total de casos. Para este clasificador la precisión sería del 81 por ciento.

Como hemos comentado anteriormente, estimar el coste de un clasificador es bastante sencillo para una determinada evidencia si se disponen de las matrices de coste y de confusión:

$$\text{Coste} = \sum_{1 \leq i \leq n, 1 \leq j \leq n} C_{i,j} \cdot M_{i,j}$$

Donde $C_{i,j}$ expresa la posición i,j de la matriz de coste, y $M_{i,j}$ la posición i,j de la matriz de confusión. Por ejemplo, el coste del clasificador asociado a la anterior matriz de confusión utilizando la matriz de coste para el problema de la sala de urgencias sería de 571.600 euros. En cambio, el coste del clasificador que siempre da el alta sería de 2.560.000 euros con una precisión del 83 por ciento. En este ejemplo vemos que un clasificador con más error puede tener mucho menos coste.

Otra situación bastante frecuente en problemas de clasificación sobre bases de datos es que haya muchísima más proporción de algunas clases sobre otras. Esto puede ocasionar

que haya muy pocos casos de una clase. Esta situación puede causar que la clase escasa se tome como ruido y sea ignorada por la teoría. Por ejemplo, si en un problema binario (sí / no) sólo hay un 1 por ciento de ejemplos de la clase no, la teoría “todo es de la clase sí” tendría un 99 por ciento de precisión. Además suele ocurrir que clasificar un ejemplo de la clase minoritaria como la clase mayoritaria suele tener asociado costes más altos que la situación inversa. Por ejemplo, en un modelo de detección de operaciones fraudulentas con tarjetas de crédito, la inmensa mayoría de las operaciones son válidas, sin embargo es bastante más problemático no detectar un fraude, que detectar un fraude falso.

Una posible solución a esta situación es utilizar la técnica denominada estratificación. Esta técnica consiste en utilizar sobremuestreo y submuestreo sobre la distribución de las clases. El sobremuestreo (*oversampling*) consiste en duplicar ejemplos (tuplas) de las clases con menor proporción, manteniendo las tuplas de las clases con mayor proporción. Esto, evidentemente, cambia la proporción de las clases, pero permite aprovechar a fondo los ejemplos de las clases más raras. Debemos utilizar sobremuestreo cuando una clase es muy extraña (poco frecuente), o cuando todas las clases (especialmente las escasas) deben ser validadas. Por otra parte, el submuestreo (*undersampling*) consigue efectos similares, pero en este caso filtrando los ejemplos (tuplas) de las clases con mayor proporción, y manteniendo las tuplas de las clases con menor proporción

Análisis ROC

Por desgracia, no siempre se dispone de una matriz de coste que permita estimar adaptar el aprendizaje a ese determinado contexto de coste. Muchas veces, la matriz de coste sólo se conoce durante el tiempo de aplicación y no durante el aprendizaje, generalmente porque los costes varían frecuentemente o son dependientes del contexto.

En esta situación, lo que se suele hacer es aprender un conjunto de clasificadores y seleccionar el que mejor se comporte para unas circunstancias o contextos de coste determinados a posteriori. Para ello, la técnica denominada análisis ROC (*Receiver Operating Characteristic*) provee herramientas que permiten seleccionar el subconjunto de clasificadores que tienen un comportamiento óptimo en general. Asimismo, el análisis ROC permite evaluar clasificadores de manera más independiente y completa a la clásica precisión.

El análisis ROC se utiliza normalmente para problemas de dos clases (se suelen denominar clase positiva y clase negativa), y para este tipo de problemas utiliza la siguiente notación para la matriz de confusión:

Estimado	Real	
	True Positives (TP)	False Positives (FP)
	False Negatives (FN)	True Negatives (TN)

Concretamente, se utilizan estos valores normalizados por columna, con la siguiente notación:

- True Positive Rate: $TPR = TP/(TP+FN)$
- False Negative Rate: $FNR = FN/(TP+FN)$
- False Positive Rate: $FPR = FP/(FP+TN)$
- True Negative Rate: $TNR = TN/(FP+TN)$

Con estos valores se completa la matriz de confusión normalizada:

Estimado	Real	
	TPR	FPR
	FNR	TNR

Lógicamente, $TPR = 1 - FNR$, y $FPR = 1 - TNR$. Ambas equivalencias permiten que se puedan seleccionar sólo dos de las ratios y construir un espacio bidimensional con ellas (llamado espacio ROC), limitado por los puntos $(0, 0)$ y $(1, 1)$. El análisis se basa en representar cada clasificador como un punto en el espacio ROC. En concreto, en el análisis ROC clásico se utiliza el valor TPR para el eje de las y , mientras que el eje de las x corresponde al FPR.

Por ejemplo, dado un clasificador con la siguiente matriz de confusión:

Estimado	Real	
	30	30
	20	70

A partir de la matriz de confusión, calculamos directamente los valores TPR y FPR, en este caso $TPR=0,6$ y $FPR=0,3$. La siguiente figura muestra este clasificador en el espacio ROC:

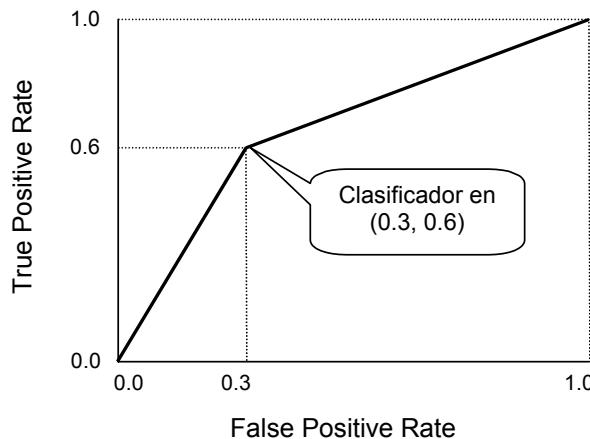


Figura 17.6. Clasificador en el espacio ROC.

Los puntos $(0, 0)$ y $(1, 1)$ corresponden, respectivamente, el clasificador que predice todo como clase positiva, y el clasificador que asigna todo como clase negativa. Estos clasificadores se conocen como los clasificadores triviales.

Representar un clasificador de esta forma en el espacio ROC tiene interesantes propiedades [Provost & Fawcett 1997]. Sobre todo, la siguiente:

- Dado un conjunto de clasificadores, se pueden descartar los clasificadores que no caigan dentro de la superficie convexa formada por todos los clasificadores, junto con los puntos $(0, 1)$, $(1, 0)$ y $(0, 0)$.

Por lo tanto, desde el punto de vista del aprendizaje sensible al coste, el mejor sistema de aprendizaje será aquel que produzca el conjunto de clasificadores con mayor área bajo la superficie convexa o AUC (*Area Under the ROC Curve*).

Un diagrama ROC típico puede verse en la Figura 17.7. En esta figura se muestran los puntos ROC correspondientes a cinco clasificadores (A, B, C y D). Los clasificadores B y E

se sitúan en el interior de la superficie convexa por lo que sabemos que siempre habrá algún otro clasificador que tenga mejor comportamiento que estos dos para cualquier contexto de coste.

Una vez seleccionado el subconjunto óptimo de clasificadores, para seleccionar el clasificador con comportamiento óptimo para un determinado contexto, necesitamos conocer el valor de los costes de clasificación errónea y la distribución de las clases (es decir, el contexto específico donde se utilizará el clasificador). Con esos valores se calcula una pendiente m asociada al contexto, denominada también inclinación o tendencia (*skew*). La pendiente se obtiene como $m = p(N) \cdot C(P|N) / p(P) \cdot C(N|P)$, donde $C(P|N)$ es el coste asociado a clasificar un objeto como P cuando realmente es N, $C(N|P)$ es el coste asociado a clasificar un objeto como N cuando realmente es P, $p(P)$ es la probabilidad de que un nuevo dato sea de clase P, $p(N)$ es la probabilidad de que un nuevo dato sea de clase N ($p(N) + p(P) = 1$). Una vez calculada la pendiente, para seleccionar el clasificador, dibujamos una línea imaginaria de pendiente m que pase por el punto $(0, 1)$, y deslizamos esa línea (sin perder la pendiente) hacia la curva ROC. El primer punto con el cual intersecte la línea nos indicará el clasificador con mejores prestaciones para el contexto asociado a la pendiente de la línea imaginaria.

Un problema que se le ha achacado a la representación tradicional del análisis ROC, es la imposibilidad de conocer el coste de un clasificador una vez seleccionado como óptimo para un determinado contexto de costes. Esta limitación ha sido parcialmente por [Drummond & Holte 2000], donde se presenta una representación que posee las ventajas del análisis ROC, pero en este caso, el coste esperado de un clasificador se calcula directamente a partir de la posición en el espacio.

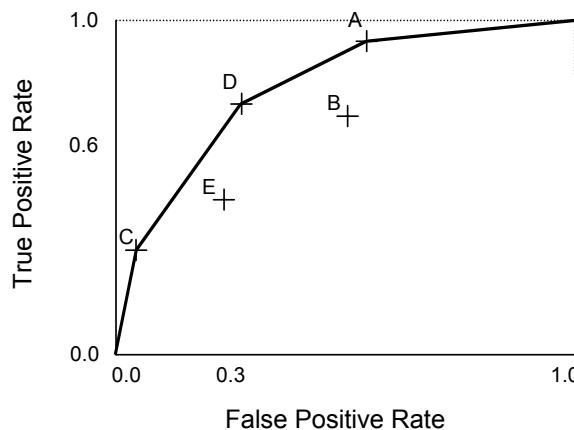


Figura 17.7. Ejemplo de diagrama ROC.

Sin embargo, la limitación principal de esta técnica es que sólo es fácilmente aplicable a problemas de dos clases. [Srinivasan 1999] ha estudiado cómo extender el análisis ROC para problemas de más de dos clases. Para un problema con c clases, el espacio ROC tiene $c \cdot (c-1)$ dimensiones. Este hecho hace prácticamente inviable el cómputo de superficies convexas para $c > 2$, por lo que se han estudiado varias aproximaciones para calcular el AUC de clasificadores en problemas multiclase [Hand & Till 2001; Ferri et al. 2003a].

La estratificación (véase la sección de evaluación basada en coste) se utiliza también para derivar un conjunto de clasificadores desde un solo clasificador, y así aumentar su AUC asociada.

Evaluación de estimadores de probabilidad

La medida AUC se ha utilizado también como medida de evaluación de estimadores de probabilidad. Como vimos en el Capítulo 6, dado un conjunto de ejemplos E y un conjunto C de c clases, un estimador de probabilidad es un conjunto de c funciones $p_{i \in C} : E \rightarrow \mathfrak{N}$ tal que:

$$\forall p_{i \in C}, e \in E : 0 \leq p_i(e) \leq 1 \text{ y } \forall e \in E : \sum p_{i \in C}(e) = 1$$

Es decir, dado un ejemplo, el modelo retorna una probabilidad de que pertenezca a una clase determinada. Este tipo de clasificadores también son denominados clasificadores suaves (*soft classifiers*). Transformar un clasificador suave en un clasificador clásico es inmediato, ya que se puede seleccionar la clase con mayor probabilidad. El paso inverso es, sin embargo, bastante más problemático. Los estimadores de probabilidad son extremadamente útiles en diversos escenarios: combinación de clasificadores (véase el Capítulo 18), aprendizaje sensible al coste, o aplicaciones críticas. Por último, dado un conjunto de ejemplos E sin clasificar, utilizar un clasificador suave para clasificarlos permite ordenarlos según la certeza (probabilidad de que la clase asignada sea cierta) que se tenga. De esta manera, si queremos minimizar los errores, se podrían considerar como válidos tan sólo aquellos que superen algún límite de certeza.

El siguiente ejemplo muestra cómo aplicar algunos de los conceptos explicados en esta sección en una aplicación: una empresa ha enviado publicidad sobre un producto a 100 clientes seleccionados aleatoriamente de un total de 100.000 para realizar un “sondeo”. A partir de los datos históricos de esos 100 clientes y la respuesta que han dado a la campaña, la empresa decide emplear técnicas de minería de datos para construir un modelo que determine, con la máxima precisión, si un cliente es potencial comprador o no. Este modelo permitirá a la empresa ahorrarse el coste del envío de publicidad a los clientes que se determinen como no interesantes, al hacer la campaña masiva sobre todos los datos.

En principio la empresa utiliza un algoritmo de aprendizaje de reglas que aprende un modelo formado por seis reglas con la siguiente distribución de los datos de entrenamiento:

Regla	Potencial	No potencial
1	2	28
2	12	3
3	21	2
4	1	7
5	0	10
6	12	2

Las reglas 2, 3, 6 asignan a la clase como potencial (es decir, han comprado el producto, o han solicitado más información) con 3, 2 y 2 errores respectivamente, y el resto (1, 4 y 5) como no potencial (no ha habido respuesta), con 2, 1 y 0 errores respectivamente. Por lo tanto, de acuerdo a esta distribución, el modelo tiene una precisión del 90 por ciento, ya que hay diez errores. En una primera fase, se determina utilizar el modelo para determinar

qué clientes son potenciales de un grupo de 30. Se clasifica este conjunto, dando el siguiente resultado:

Regla	Casos	Clase
1	3	No Pot.
2	7	Pot.
3	5	No Pot.
4	11	No Pot.
5	3	Pot.
6	1	Pot.

Es decir, en principio, el modelo determina 13 clientes como potenciales, y 17 como no potenciales. Sin embargo, la dirección de la empresa, motivada por la necesidad de una drástica reducción de costes de correo, decide enviar sólo publicidad a los clientes que el modelo determine que tengan más de un 80 por ciento de probabilidad de adquirir el producto. Para conocer qué clientes son los seleccionados, necesitamos un estimador de probabilidades, o clasificador suave. Convertir un modelo basado en reglas excluyentes en clasificador suave es relativamente directo. Una primera aproximación es utilizar directamente los ejemplos de entrenamiento para asignar las probabilidades a cada una de las reglas. Sin embargo, diversos estudios [Provost & Domingos 2003; Ferri et al. 2003b] han demostrado que esta aproximación puede mejorarse sensiblemente si las probabilidades se calculan utilizando correcciones. En concreto, la clásica corrección de Laplace (véase la Sección 5.4.1 o Sección 10.3.1), permite mejorar de manera significativa la calidad del clasificador suave⁵⁴. De esta manera, el modelo utilizado previamente correspondería al siguiente clasificador suave:

Regla	Sin corrección de Laplace		Con corrección de Laplace	
	Potencial	No potencial	Potencial	No potencial
1	0,067	0,933	0,094	0,906
2	0,8	0,2	0,765	0,235
3	0,913	0,087	0,88	0,12
4	0,125	0,875	0,2	0,8
5	0	1	0,083	0,917
6	0,857	0,143	0,813	0,187

Sólo las reglas 3 y 6 otorgan probabilidades de ser un cliente potencial por encima del 80 por ciento, y por tanto, se enviaría la propaganda a los seis clientes que caen en esas reglas, del total de 30.

Otros tipos de coste

Finalmente, nótese que además de los costes de clasificación errónea, existen otros costes asociados al aprendizaje como por ejemplo costes de evaluar atributos (costes de test), o costes de computación. Turney estableció una interesante taxonomía de tipos de costes en aprendizaje automático en [Turney 2000]. Entre todos los costes descritos en este artículo, cabe resaltar a los costes de test, o los costes asociados a conocer el valor de un determina-

⁵⁴ Existen otras correcciones más elaboradas, que consiguen incrementar aún más la calidad de un estimador de probabilidades (Ver [Provost & Domingos 2003; Ferri et al. 2003b]).

do atributo para realizar una predicción. Este tipo de costes tiene considerable importancia en algunas áreas de aplicación tales como la medicina. Por ejemplo, en problemas de predicción de alguna dolencia, algunos de los atributos del problema pueden ser costosos (radiografías) o incluso dolorosos o peligrosos para los pacientes (punción en la médula). Existen varios trabajos que han introducido modificaciones en las técnicas de aprendizaje para que además de obtener modelos precisos, minimicen al máximo el coste de test de los modelos. Por ejemplo, en árboles de decisión, [Núñez 1988] introduce una modificación a los criterios de partición clásicos de manera que consideran el coste del atributo seleccionado para realizar la partición.

El siguiente ejemplo muestra la importancia de tener en cuenta este tipo de costes en el aprendizaje de modelos. Considere un problema en el cual se debe seleccionar entre tres variantes (C1, C2 y C3) de una misma enfermedad a partir de los siguientes atributos:

ATRIBUTO	TIPO	VALORES
BP_Min	Numérico	
BP_Max	Numérico	
Div_end	Nominal	pos/neg
Cysts_Scopy	Nominal	pos/neg
Meningitis_Lumbar	Nominal	pos/neg
Cysts_Echo	Nominal	pos/neg
Glucose_BA	Numérico	
Leucocytoses_Urin	Nominal	pos/neg

Medir los datos de cada atributo tiene asociado el siguiente coste (calculado teniendo en cuenta costes económicos y riesgos para el paciente):

ATRIBUTO	COSTE
BP_Min	1€
BP_Max	1€
Div_end	30€
Cysts_Scopy	50€
Meningitis_Lumbar	200€
Cysts_Echo	15€
Glucose_BA	15€
Leucocytoses_Urin	10€

Supongamos que tenemos tres árboles de decisión (expresados como reglas) diferentes para este problema:

Árbol1

Cysts_Scopy=neg, Glucose_BA>=120	C1	0.2
Cysts_Scopy=neg, Glucose_BA<120, Leucocytoses_Urin=neg	C1	0.3
Cysts_Scopy=neg, Glucose_BA<120, Leucocytoses_Urin=pos	C2	0.1
Cysts_Scopy=pos	C3	0.4

Árbol2

BP_Min>=100, Cysts_Echo=pos	C3	0.1
BP_Min>=100, Cysts_Echo=neg, Meningitis_Lumbar=neg	C1	0.03
BP_Min>=100, Cysts_Echo=neg, Meningitis_Lumbar=pos	C2	0.05
BP_Max>=150, Cysts_Echo=pos	C3	0.15
BP_Max>=150, Cysts_Echo=neg, Meningitis_Lumbar=neg	C1	0.02
BP_Max>=150, Cysts_Echo=neg, Meningitis_Lumbar=pos	C2	0.05

BP_Min<100, BP_Max<150, Div_end=pos	C3	0.15
BP_Min<100, BP_Max<150, Div_end=neg	C1	0.45
Árbol3		
Leucocytoses_Urin=neg, Cysts_Echo=pos	C3	0.4
Leucocytoses_Urin=neg, Cysts_Echo=neg	C1	0.5
Leucocytoses_Urin=pos	C3	0.1

Los tres árboles que hemos visto utilizan diferentes atributos para realizar la predicción. En la parte derecha de cada regla tenemos su frecuencia (calculada utilizando el conjunto de entrenamiento), es decir el porcentaje de ejemplos que han caído en esa regla. Empleando esta información podemos estimar el coste medio de test asociado a cada árbol de la siguiente forma:

$$\text{Coste} = \sum_{1 \leq i \leq n} \left(\text{frecuencia}(\text{regla}_i) \cdot \sum_{a \in \text{Atributos}(\text{regla}_i)} \text{Coste}(a) \right)$$

Es decir, el coste medio de un árbol equivale a la suma del coste de cada regla de ese árbol, multiplicada por la frecuencia de aparición de cada regla. Utilizando esta expresión, obtenemos que el coste medio del árbol 1 es de 62 euros, el del árbol 2 corresponde a 57,76 euros, y el del árbol 3 es de 23,5 euros. Por lo tanto, si los tres árboles tuviesen una precisión similar, elegiríamos el tercero, ya que es el que realiza los tests sobre los atributos de la manera menos costosa.

Otras medidas de evaluación

Otras medidas de evaluación muy populares en el área de recuperación de información (IR) [Salton & McGill 1983; Baeza-Yates & Ribeiro-Neto 1999] son la precisión y el alcance (*precision and recall*). Estas medidas están basadas en la matriz de confusión normalizada para problemas binarios. La precisión mide la probabilidad de que si un sistema clasifica un documento en una cierta categoría, el documento realmente pertenezca a la categoría. Por otro lado, el *alcance* mide la probabilidad de que si un documento pertenece a cierta categoría, el sistema lo asigne a la categoría. Recordando el aspecto que tenía la matriz de confusión normalizada para problemas binarios:

		Real	
		TPR	FPR
Estimado	TPR	FNR	TNR
	FPR		

TPR + *TNR* representan los aciertos del sistema y *FNR*+*FPR* son los errores, y la suma de las cuatro celdas equivale al número total de ejemplos. Los valores de la tabla de contingencia permiten estimar las medidas de precisión y alcance según las siguientes expresiones:

$$\text{precision} = \frac{\text{TPR}}{\text{TPR} + \text{FNR}} \quad \text{alcance} = \frac{\text{TPR}}{\text{TPR} + \text{FPR}}$$

Describir el comportamiento de un modelo con dos medidas es poco útil para comparar sistemas, por lo que es común utilizar la medida F_β , que se define como:

$$F_\beta = \frac{(1 + \beta^2) \text{precision} \cdot \text{alcance}}{\beta^2 \cdot \text{precision} + \text{alcance}}$$

El parámetro β controla la importancia relativa entre las dos medidas. Si $\beta=1$, F_β es la media armónica de la precisión y el *alcance*, siendo el valor más común para esta medida. En la Sección 14.4.2 vimos un ejemplo de uso de estas medidas para la categorización de textos con máquinas de vectores soporte. En la medida, es común usar el valor 1, que da igual importancia a las dos medidas.

17.3 Evaluación de modelos de regresión

Cuando se desea evaluar la calidad de una hipótesis de regresión, obviamente, no se puede evaluar la precisión como el número de aciertos. En estos casos, se suele estimar la calidad de un modelo calculando la *diferencia* entre las predicciones del modelo y las de la función objetivo.

Considérese una función objetivo f , y un modelo de regresión h para esa función. Dado un dataset D formado por n ejemplos, una de las medidas de evaluación más comúnmente usada se basa sumar los errores cuadráticos:

$$MSE = \frac{1}{n} \sum_{x \in D} (h(x) - f(x))^2$$

A esta medida se le denomina MSE (*Mean Squared Error*) error cuadrático medio. Dado que la suma de los cuadrados puede evitar conocer la magnitud real de los errores, se suele también utilizar la raíz cuadrada de la suma final:

$$RMSE = \sqrt{\frac{1}{n} \sum_{x \in D} (h(x) - f(x))^2}$$

Un problema de estas aproximaciones es que elevar al cuadrado la diferencia tiende a dar demasiado peso a los errores más extremos, afectando al resultado final. Una manera de limitar este problema es utilizar el error absoluto medio (MAE):

$$MAE = \frac{1}{n} \sum_{x \in D} |h(x) - f(x)|$$

Esta medida realiza la media de los errores ignorando el signo de los mismos. Otra alternativa que se puede considerar es utilizar errores relativos, de manera que pese lo mismo un error de magnitud 10 en una predicción de 100, que un error de magnitud 1 en una predicción de 10. El error cuadrático relativo (*relative squared error*) se define como:

$$RSE = \frac{1}{n} \sum_{x \in D} \frac{(y(x) - f(x))^2}{(y(x) - \bar{f})^2}, \text{ donde } \bar{f} = \frac{1}{n} \sum_{x \in D} f(x)$$

A esta medida se le pueden aplicar las mismas variaciones que hemos visto anteriormente, es decir, aplicar la raíz cuadrada al valor final, o utilizar el error absoluto dando lugar a nuevas medidas de evaluación.

Dada la gran cantidad de medidas, seleccionar la adecuada para una determinada situación puede ser complicado. El error cuadrático medio da más peso a las divergencias mayores que a las menores. Utilizar la raíz cuadrada permite reducir el valor final a la misma magnitud de los errores. Finalmente, los errores relativos compensan los errores con la magnitud del valor a predecir.

En cuanto a la metodología para la realización de los experimentos, los métodos introducidos en la Sección 17.2 (entrenamiento / test, validación cruzada o *bootstrap*) son también válidos para la regresión.

17.4 Comparación de técnicas de aprendizaje

En muchas ocasiones, se desea comparar dos técnicas de aprendizaje, para conocer si alguna de las dos se comporta significativamente mejor con respecto a una evidencia dada. Para ello, lo que se hace es realizar un test estadístico con cierto nivel de significancia. Una de las pruebas estadísticas más populares para este propósito es el llamado *t-test* (*student's t-test*). Esta prueba se utiliza para certificar si las medias de dos grupos son estadísticamente diferentes, y se basa en calcular el valor t que está definido de la siguiente manera:

$$t = \frac{\bar{d}}{s_d}$$

donde \bar{d} representa la diferencia entre la media de los dos grupos y s_d la variabilidad entre dos grupos.

Considérese que se desea comparar dos métodos de aprendizaje x e y . Para ello, se realiza una validación cruzada con k particiones, obteniéndose en cada caso k valores de error de muestra $\{x_1, x_2, \dots, x_{10}\}$ y $\{y_1, y_2, \dots, y_{10}\}$. Dados esos resultados, los valores \bar{d} y s_d se pueden calcular:

$$\bar{d} = \frac{d_1 + d_2 + \dots + d_k}{k} \quad s_d = \sqrt{\frac{d_1^2 + d_2^2 + \dots + d_k^2}{k}}$$

donde $d_i = x_i - y_i$. Una vez calculado el valor t , éste corresponde a una variable *t-student* con $k-1$ grados de libertad. Para determinar si la diferencia es significativa, se debe fijar un nivel de confianza y comparar con el valor límite de la variable *t-student* en la tabla correspondiente en un manual o paquete estadístico (o en una hoja de cálculo) para esos grados de libertad e intervalo de confianza.

En la anterior prueba, hemos supuesto que los valores error de muestra $\{x_1, x_2, \dots, x_{10}\}$ y $\{y_1, y_2, \dots, y_{10}\}$ se han calculado bajo las mismas condiciones, es decir utilizando las mismas muestras. A esta situación se le denomina test pareado. También es posible aplicar una variante del *t-test* utilizando muestras diferentes, en este caso se denomina test no pareado.

17.5 Evaluación basada en complejidad de la hipótesis. El principio MDL

En este punto presentamos un punto de vista diverso de abordar la evaluación de hipótesis: la evaluación basada en la complejidad o el tamaño de la hipótesis.

El problema de seleccionar una teoría de entre un conjunto de teorías plausibles ha sido centro de discusión filosófica a lo largo de la historia. Una de las opciones más defendidas ha sido preferir las teorías más simples. En concreto, al principio de elegir, en igualdad de condiciones, la hipótesis más simple, se le conoce como navaja de Occam (*Occam's Razor*), debido a que fue promulgada por el filósofo medieval William de Occam.

Pero, ¿qué significa la hipótesis más simple?, ¿cómo podemos medir la complejidad de una determinada hipótesis? Considérense dos diferentes hipótesis para una misma evidencia. La primera es una hipótesis bastante compleja y extensa que cubre todos los ejemplos de la evidencia. La segunda es muy simple, pero no cubre todos los ejemplos de la evidencia; hay algunos ejemplos, excepciones, que no son cubiertos. Llegados a este punto, ¿cómo podemos determinar la hipótesis más simple?

Esta incógnita se puede resolver mediante el principio de la longitud de descripción mínima, o principio MDL (*Minimum Description Length*) [Rissanen 1978; Li & Vitányi 1997]. Esta teoría consiste en calcular el tamaño en bits (unidad de información) de la descripción de una hipótesis más la descripción de los ejemplos que no son cubiertos (excepciones).

Formalmente, el principio MDL recomienda seleccionar la hipótesis h que minimice la expresión:

$$K(h) + K(D | h)$$

Donde $K(h)$ es la complejidad en bits de describir la hipótesis h , y $K(D|h)$ es la complejidad en bits de describir la evidencia D a partir de la hipótesis h (lo que incluye las excepciones).

La moraleja es que una hipótesis simple con muchas excepciones no es adecuada, al igual que tampoco es adecuada una hipótesis compleja sin excepciones. La idea del principio MDL es buscar el compromiso.

Por otra parte, en muchos casos, nos interesa obtener la hipótesis más probable h a partir de un conjunto de datos D , es decir la hipótesis con mayor $P(h|D)$, también llamada hipótesis máxima a posteriori (MAP).

Una manera de seleccionar la hipótesis MAP es utilizar el teorema de Bayes (véase la Sección 10.2). Formalmente, una hipótesis h_{MAP} , es una hipótesis MAP de un conjunto de hipótesis H , si:

$$h_{MAP} \equiv \arg \max_{h \in H} p(h | D) = \arg \max_{h \in H} \frac{p(D | h)p(h)}{p(D)} = \arg \max_{h \in H} p(D | h)p(h)$$

Podemos ignorar $p(D)$, ya que es una constante independiente de h . Tomando logaritmos en base 2:

$$\begin{aligned} h_{MAP} &\equiv \arg \max_{h \in H} \log_2 p(D | h) + \log_2 p(h) = \\ &= \arg \min_{h \in H} -\log_2 p(D | h) - \log_2 p(h) \end{aligned}$$

Esta última expresión puede considerarse como una justificación de la selección de las hipótesis más simples, si asumimos algunos conocimientos de la teoría de la información. En particular si asumimos el “prior universal”, entonces $p(h) = 2^{-K(h)}$, es decir las hipótesis más cortas son más probables, lo que es una formalización de la navaja de Occam. Por lo tanto, la última expresión corresponde al principio MDL:

$$MDL(h, D) = K(h) + K(D | h)$$

El principio MDL es, por lo tanto, un método que nos permite seleccionar una hipótesis de acuerdo con su complejidad y la información que se requiere para describir la evidencia a partir de esa hipótesis (excepciones). De esta manera, el principio MDL da un marco teórico para tratar el problema del sobreajuste (véase la Sección 6.4.1).

Veamos un ejemplo. Recuperemos los ejemplos (D) de los clientes de la compañía de seguros que vimos en la Tabla 2.1 (página 29), que repetimos a continuación:

edad	hijos	practica_deporte	salario	buen_cliente
joven	sí	no	alto	sí
joven	no	no	medio	no
joven	sí	sí	medio	no
joven	sí	no	bajo	sí
mayor	sí	no	bajo	sí
mayor	no	sí	medio	sí
joven	no	sí	medio	sí
joven	sí	sí	alto	sí
mayor	sí	no	medio	sí
mayor	no	no	bajo	no

Tabla 17.1. Datos de una compañía de seguros.

Y consideremos las siguientes hipótesis h_1 , h_2 y h_3 :

Hipótesis 1:	Hipótesis 2:	Hipótesis 3:
<p>SI hijos="sí" ENTONCES buen-cliente="sí"</p> <p>EN CASO CONTRARIO: buen-cliente="no"</p>	<p>SI hijos="no" Y practica-deporte="no" ENTONCES buen-cliente="no"</p> <p>EN CASO CONTRARIO: buen-cliente="sí"</p>	<p>SI hijos="no" Y practica-deporte="no" ENTONCES buen-cliente="no"</p> <p>SI edad="joven" Y hijos="sí" Y practica-deporte="sí" ENTONCES buen-cliente="no"</p> <p>EN CASO CONTRARIO: buen-cliente="sí"</p>

Podríamos estimar la complejidad del modelo como el número de igualdades. De este modo tendríamos $K(h_1)=3$, $K(h_2)=4$ y $K(h_3)=8$. Podemos considerar también que cada error de predicción es un bit. Por tanto, tenemos $K(D|h_1)=3$, $K(D|h_2)=1$, $K(D|h_3)=0$.

Atendiendo a la navaja de Occam, nos quedaríamos con la hipótesis 1, pues es la que tiene menor tamaño y es la más simple. Atendiendo a la minimización del error de entrenamiento nos quedaríamos con la hipótesis 3, pues no tiene errores. Pero, justamente, si usamos el principio MDL, tenemos que $MDL(h_1,D)=6$, $MDL(h_2,D)=5$, $MDL(h_3,D)=8$. Por tanto, nos quedaríamos con la hipótesis 2, ya que busca un compromiso para cubrir bastantes ejemplos y capturar así las regularidades presentes en los datos (no subajusta), pero no llega a encontrar reglas demasiado específicas que cubran pocos ejemplos y que simplemente sirvan para evitar excepciones (no sobreajusta).

Lógicamente, de la forma de medir $K(h)$ y $K(D|h)$ va a depender la elección, y existen maneras más sofisticadas, ajustadas y precisas de hacerlo, pero podemos ver con este ejemplo que incluso con una manera bastante sencilla de aproximar estas dos medidas se pueden hacer buenas elecciones.

De hecho, el principio MDL se ha utilizado en el diseño de varios algoritmos de aprendizaje, por ejemplo, se ha utilizado para podar árboles de decisión [Quinlan 1990], en el criterio de partición de árboles de decisión [Ferri et al. 2001], o bien en sistemas de ILP como PROGOL (véase el Capítulo 12).

17.6 Evaluación de modelos de agrupamiento

En las secciones anteriores hemos presentado diversas técnicas de evaluación de hipótesis para problemas de clasificación y regresión. En este punto, nos centramos en los modelos de agrupamiento. Este tipo de modelos son más difíciles de evaluar, ya que no existe una clase o un valor numérico donde medir las veces que el modelo aprendido predice correctamente.

Una primera aproximación consiste en utilizar la verosimilitud (*likelihood*), ya aparecida en otros capítulos y que significa quedarse con la hipótesis con la cual los datos sean más verosímiles (es decir, que maximice $P(D|h)$, ignorando, por tanto, el prior de hipótesis). Para ello, sea $p(x)$ la probabilidad estimada de observar un punto en la posición x utilizando el modelo a evaluar. En realidad, p es una función de densidad de probabilidad. Si el modelo es bueno, se deben esperar probabilidades altas para los puntos que son observados. Consecuentemente, podemos tomar $p(x)$ como una función de evaluación para el punto x .

Esta idea se puede generalizar para n puntos combinando la probabilidad mediante la multiplicación de cada valor único, si asumimos que los puntos son seleccionados independientemente:

$$L = \prod_{i=1}^n p(x(i))$$

Utilizando los logaritmos de las probabilidades:

$$\log L = \sum_{i=1}^n \log p(x(i))$$

Y si utilizamos logaritmos negativos, el objetivo consistirá en minimizar la siguiente función:

$$S_L = -\log L = -\sum_{i=1}^n \log p(x(i))$$

El término $-\log p(x)$ puede verse como el error para el punto x . La suma de todos los puntos es por lo tanto una medida de error. En realidad, S_L puede también interpretarse como un tipo de entropía que mide cómo el modelo describe los datos de entrenamiento.

En el caso de trabajar con modelos que no otorgan probabilidad a los puntos, se puede utilizar una medida que estudia cuán compactos son los grupos del modelo. Una medida bastante utilizada consiste en utilizar la suma del error cuadrático de cada grupo.

$$S_{EC}(m) = \sum_{i \in \text{cluster}_k} \|x(i) - c_k\|^2$$

Donde m es el modelo formado por n grupos, con centros en $\{c_1, c_2, \dots, c_n\}$.

Otra opción que se ha considerado es utilizar la distancia entre los grupos como medida de calidad del modelo de agrupamiento. Cuanto mayor sea la distancia entre los grupos, significa que se ha efectuado una mejor separación, y por tanto el modelo se

considera mejor. Falta determinar qué se considera como distancia entre grupos; se han estudiado varias opciones, podemos destacar las siguientes⁵⁵:

- Distancia media: la distancia entre dos grupos se calcula como la media de las distancias entre todos los componentes de ambos grupos.
- Distancia simple: se considera en este caso la distancia entre los vecinos más próximos de los dos grupos.
- Distancia completa: similar a la anterior, pero utilizando los vecinos más lejanos, es decir la distancia entre los componentes que se encuentren a más distancia.

Otra alternativa para conocer si un determinado modelo de agrupamiento es adecuado para un conjunto de datos consiste en aprender varios modelos desde ese mismo conjunto utilizando diversas técnicas de aprendizaje. Si los comparamos entre sí y coinciden, podremos pensar que esa agrupación es acertada. Para evaluar la similitud entre dos modelos de agrupamiento podemos utilizar una estrategia, similar a la aproximación entrenamiento / test de clasificación y regresión, basada en la partición de los datos en dos partes. La primera de ellas se utiliza para construir modelos de agrupamiento, y la segunda para comprobar si los modelos construidos son similares. Para comprobar la similitud entre dos modelos a y b , con n_a y n_b grupos respectivamente, generamos una matriz con n_a filas y n_b columnas, y se inicializa a 0. Por cada dato perteneciente al conjunto de test utilizamos los modelos a y b para que asignen un grupo cada uno. Los grupos se utilizan para incrementar una celda de la matriz de comparación. Por ejemplo, si a retorna 1, y b retorna 2, incrementaremos la posición (1, 2). De esta manera, tras evaluar todo el conjunto de test, si los modelos son similares, la matriz estará concentrada en unas pocas celdas. Si son diferentes, la matriz estará bastante dispersa.

Por último, es posible utilizar el principio MDL para evaluar modelos de agrupamiento en particular, o modelos descriptivos en general. Suponga un conjunto E de n datos, que es dividido por un algoritmo de agrupamiento en k grupos. Si el agrupamiento es bueno, podría utilizarse para codificar E de una manera más eficiente. El mejor agrupamiento permitiría la mejor manera de codificación de los datos.

Como conclusión, destacamos que los modelos descriptivos en general, son complicados de evaluar debido a la ausencia de una clase determinada donde medir el grado de acierto del modelo. La mejor evaluación de este tipo de modelos es saber si el modelo resultado de la fase de aprendizaje tiene un comportamiento útil cuando se utilice en su área de aplicación.

17.7 Evaluación de reglas de asociación

Las reglas de asociación (véase el Capítulo 9) expresan patrones de comportamiento entre los datos. En concreto, las combinaciones de valores de los atributos (items) que suceden más frecuentemente. Dado que en este tipo de reglas no hay un atributo definido como clase sobre el cual evaluar la calidad de una regla, no son válidos, a priori, los procedimientos vistos para la clasificación.

⁵⁵ Éstas son muy similares a las distancias de enlace vistas en la Sección 16.2.3.

Para evaluar una regla de asociación, se suele trabajar con dos medidas para conocer la calidad de la regla: cobertura y confianza. La cobertura (también denominada soporte) de una regla se define como el número de instancias en las que la regla se puede aplicar. Por otra parte, la confianza (también llamada precisión) mide el porcentaje de veces que la regla se cumple cuando se puede aplicar. Por ejemplo, considérese un conjunto de 1.000 datos y una regla que 200 ejemplos cumplen las condiciones de la parte izquierda. De estos 200 ejemplos, se cumple la parte derecha 150 veces. Por tanto, tendremos una cobertura de 150 (15 por ciento) y una confianza del 75 por ciento (150 entre 200).

Existen otras medidas de calidad para reglas de asociación no basadas directamente en el número de ejemplos cubiertos, aunque miden otro tipo de características de las reglas tales como el interés o novedad que aporta una regla respecto al conocimiento previo. Por ejemplo en la Sección 9.3.1 hemos introducido el interés entre dependencias. De la misma manera podemos calcular el interés de una regla y utilizarlo como medida de calidad.

17.8 Otros criterios de evaluación

La mayoría de las medidas de evaluación que hemos visto en el capítulo se basan en criterios que utilizan el número de ejemplos que el modelo contempla. Existen otras aproximaciones que evalúan modelos basándose en criterios tales como:

- **Interés:** el interés de un modelo intenta medir la capacidad de ese modelo de suscitar la atención del usuario del modelo.
- **Novedad:** criterio relacionado con la capacidad de un modelo de sorprender al usuario con respecto al conocimiento previo que tenía sobre determinado problema. Criterio muy afín al interés.
- **Comprendibilidad o Inteligibilidad:** como hemos comentado en varias ocasiones a lo largo del libro, la comprensibilidad de un modelo es en algunas ocasiones un factor muy importante.
- **Simplicidad:** este criterio se basa en establecer el tamaño o complejidad del modelo. Obviamente, este criterio está muy relacionado con el criterio de comprensibilidad.
- **Aplicabilidad:** en este caso, la calidad de un modelo se basa en su capacidad de ser utilizado con éxito en el contexto real donde va ser aplicado.

Algunos de estos criterios están muy relacionados, por ejemplo, el interés y la novedad, el interés y la aplicabilidad, así como la comprensibilidad y la simplicidad. A continuación vamos a detallar ligeramente los criterios más significativos.

17.8.1 Comprendibilidad

La comprensibilidad del modelo es una cuestión mucho más subjetiva, ya que un modelo puede ser poco comprensible para un usuario y muy comprensible para otro. A pesar de ello, podemos encontrar algunas medidas útiles. Así, cuando el modelo se expresa como un conjunto de reglas, cuanto más cortas sean (es decir, cuanto menor sea el número de condiciones) más comprensible suele ser el modelo. Esta medida puede calcularse contando las condiciones de las reglas y el número de reglas del modelo. Sin embargo, la longitud no es el único factor que influye en la comprensibilidad del modelo. Deberían también tenerse en cuenta las preferencias de los usuarios y otras cuestiones semánticas.

Así, el nivel de abstracción usado para expresar los valores de los parámetros en el modelo puede influir en su comprensibilidad. En general, cuanto más alto es el nivel de abstracción mayor será la comprensibilidad.

Por ejemplo, una condición para el atributo *Salario* de la forma *Salario>60.000 euros* puede resultar más difícil de comprender que la condición *Salario=alto*. La discretización de atributos (comentada brevemente en la página 24 y que se vio en detalle en el Capítulo 4) puede servir para obtener generalizaciones más comprensibles. No obstante, tenemos que tener en cuenta que esta transformación de atributos continuos en discretos puede hacer que perdamos detalles importantes sobre los datos y que la precisión del modelo se vea afectada negativamente por ello.

Del mismo modo, y como se ha comentado en varias ocasiones, ciertos modelos no son comprensibles por el tipo de técnica utilizada, por ejemplo, las redes neuronales, aunque el resultado puede ser muy preciso. Resumiendo, en muchos casos hay que alcanzar un consenso entre comprensibilidad y precisión.

17.8.2 Interés

Existen patrones evaluados muy satisfactoriamente con las técnicas vistas en este capítulo que pueden no tener ningún interés. Por ejemplo, una regla de asociación como “si el paciente ingresa en maternidad entonces el sexo del paciente es mujer”, es muy válida con las medidas vistas (soporte y precisión), pero no sirve de nada, porque no aporta nada nuevo e interesante.

Las medidas del interés de un modelo pueden ser de dos tipos: subjetivas y objetivas [Freitas 2002]. Las primeras se basan en tener en cuenta conocimiento o expectativas previas del usuario. En estos casos el modelo se considera interesante (con la connotación de novedoso) si contradice el conocimiento o expectativas previas. Las medidas objetivas, sin embargo, tratan de estimar cuán interesante es el modelo para el usuario basándose principalmente en los datos minados (en su entropía, en su distribución, etc.). La principal diferencia entre ambas es que las medidas subjetivas son dependientes de la aplicación, ya que se basan en las impresiones que el usuario tiene a priori sobre dicha aplicación, mientras que las medidas objetivas son independientes del dominio de la aplicación. Muchas son las aproximaciones que se han definido para ambos tipos de medidas (véase [Freitas 1998] para un resumen).

17.8.3 Aplicabilidad

Un modelo puede tener una precisión muy alta con respecto a los datos de entrenamiento, pero poseer una aplicabilidad pobre. Por ejemplo, un modelo que utiliza frecuentemente un atributo que en muchos casos reales se desconoce para realizar las predicciones, y esto provoca que la predicción sea errónea.

Para finalizar el capítulo podemos concluir que la correcta elección del método de evaluación es un paso determinante del éxito final de todo el proceso de minería de datos. Una evaluación realizada de manera incorrecta puede provocar que seleccionemos un modelo que presente bajas prestaciones en el entorno real de aplicación. Hemos de considerar las características propias de cada problema, adaptando la evaluación (y si es posible el aprendizaje) a esas características. Por ejemplo, en problemas de predicción

médica a partir de resultados de pruebas realizadas sobre pacientes, es frecuente encontrar costes importantes asociados a los atributos, por lo que estos costes se deben tener en cuenta a la hora de evaluar (y aprender) los modelos.

Capítulo 18

COMBINACIÓN DE MODELOS

En el anterior capítulo hemos introducido el principio de la navaja de Occam. Este principio nos indica que dado un conjunto de hipótesis válidas debemos elegir la más simple. El filósofo griego Epicurus, sin embargo, defendía una tesis prácticamente inversa: “si más de una hipótesis es consistente con los datos, debemos mantenerlas todas”. Es decir, debemos considerar todas las hipótesis plausibles, ya que todas extraen patrones que podrían ser válidos. La idea es bien conocida en los sistemas de decisión. Es preferible consultar a varios asesores que a uno sólo. Esta teoría es la base de los métodos llamados métodos multiclasicadores, métodos de construcción de meta-modelos, métodos de fusión o ensamble (*ensemble methods*). La idea intuitiva es utilizar un conjunto de modelos diferentes para combinarlos creando un nuevo modelo. Otra técnica similar que estudiamos en este capítulo es la fusión o combinación de algoritmos de aprendizaje creando nuevas técnicas de aprendizaje híbridas que unifican las ventajas de los métodos de aprendizaje fusionados en un sólo método de aprendizaje. Estas técnicas se conocen como métodos híbridos.

18.1 Introducción

Con el objetivo de mejorar la precisión de las predicciones, ha surgido un interés creciente en los últimos años en la definición de métodos que combinan hipótesis. Estos métodos construyen un conjunto de hipótesis (*ensemble*), y entonces combinan las predicciones del conjunto de alguna manera (normalmente por votación) para clasificar ejemplos o para hacer regresión sobre ejemplos. La precisión obtenida por esta combinación de hipótesis supera, generalmente, la precisión de cada componente individual del conjunto. A estas técnicas de combinación de hipótesis se les conoce como métodos multiclasicadores, métodos de construcción de meta-modelos, modelos combinados o métodos de ensamblaje

de modelos (*ensemble methods*)⁵⁶ [Dietterich 2000a]. La combinación de modelos se ha desarrollado principalmente para modelos predictivos (clasificación y regresión), aunque ciertas ideas podrían extenderse a modelos descriptivos. En la Figura 18.1 podemos apreciar el esquema general de combinación de modelos.

La calidad del modelo combinado depende en alto grado de la precisión y diversidad de los componentes del conjunto [Hansen & Salamon 1990]. Considérese el siguiente ejemplo [Dietterich 2000a], dado un conjunto formado por tres clasificadores $\{h_1, h_2, h_3\}$, sea x un nuevo dato a ser clasificado. Si los tres clasificadores son similares, entonces cuando $h_1(x)$ sea erróneo, probablemente $h_2(x)$ y $h_3(x)$ también lo serán. Sin embargo, si los clasificadores son lo bastante diversos, los errores que cometan estarán poco correlacionados, y por tanto, cuando $h_1(x)$ sea erróneo, $h_2(x)$ y $h_3(x)$ podrían ser correctos, y entonces, si la combinación se realizase por votación mayoritaria, el conjunto combinado clasificaría correctamente el dato x .

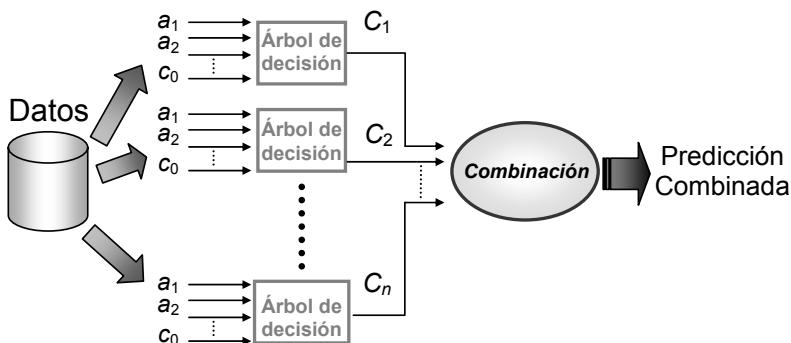


Figura 18.1 Combinación de modelos usando árboles de decisión como modelos base.

Combinar hipótesis puede verse también como un incremento en el poder expresivo del lenguaje de representación. Un algoritmo de aprendizaje puede emplearse para aprender diferentes hipótesis $\{h_1, h_2, \dots, h_m\}$ en un determinado espacio de hipótesis H . Construyendo un multiclásificador formado por este conjunto, el modelo combina cada uno de los votos de cada clasificador o regresor, de manera que la hipótesis combinada puede estar fuera del espacio H .

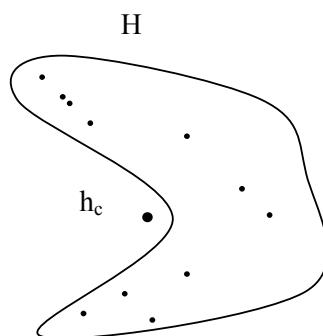


Figura 18.2. Representación de un multiclásificador.

⁵⁶ El término “multiclásificador” puede provocar confusión, ya que también sirve para efectuar regresión. En este capítulo, mantenemos este término, pero se debe tener en cuenta este hecho.

La Figura 18.2 muestra esta situación. La curva externa denota el espacio de hipótesis H . Los puntos representan los componentes del conjunto de modelos y h_c es el multiclassificador.

Otra importante ventaja de los métodos multiclassificadores es que la combinación de varios modelos semánticamente diferentes subsana el problema del sobreajuste de los datos de entrenamiento.

Existen dos puntos clave que afectan la calidad final de un modelo multiclassificador; principalmente la construcción de modelos lo suficientemente diferentes y precisos, y posteriormente la forma de combinación de los modelos. En los siguientes puntos abordamos con detenimiento estos temas.

Otra técnica de combinación, muy difundida en los últimos años, consiste en la definición de nuevas técnicas de aprendizaje mediante la composición de partes de otras técnicas de aprendizaje básicas. De esta manera, los nuevos métodos permiten aglutinar características de los métodos de aprendizaje básicos, por ejemplo la capacidad de representación. Estos métodos de combinación de técnicas de aprendizaje se denominan métodos híbridos. Ejemplos de métodos híbridos son las RBF (*Radial Basis Functions*), *stacking* o *cascading*.

18.2 Métodos de construcción de multiclassificadores

Muchos métodos han sido definidos para la construcción de un conjunto de modelos desde una única evidencia, utilizando para ello un único método de aprendizaje. Existen, además, muchos modos de generar y combinar conjuntos de modelos. Dietterich, en [Dietterich 2000a], estableció una taxonomía de métodos de construcción de conjuntos de clasificadores o regresores:

- **Manipulación de los datos de entrenamiento:** estos métodos construyen un conjunto de modelos mediante la ejecución repetidas veces del mismo algoritmo de aprendizaje. En cada iteración se parte de un conjunto diferente de datos de entrenamiento, y de esta manera se genera un modelo diferente. La clave en este tipo de métodos multiclassificadores es la definición de un mecanismo adecuado para la selección de subconjuntos a partir del conjunto de datos de entrenamiento. Ejemplos de métodos multiclassificadores de esta familia son: *Bagging* [Breiman 1996; Quinlan 1996b], *Boosting* [Freund & Schapire 1996; Quinlan 1996b], y *Cross-validated Committees* [Parmanto et al. 1996].
- **Manipulación de los atributos de entrada:** una técnica general para aprender los componentes de un conjunto de modelos consiste en alterar el conjunto de atributos de entrada del algoritmo de aprendizaje. La técnica *forest* [Ho 1998] pertenece a esta familia.
- **Manipulación de los datos de salida:** otro método diferente para problemas de clasificación consiste en modificar las clases de los ejemplos del conjunto de entrenamiento. En [Dietterich & Bakiri 1995] se define una técnica consistente en generar subconjuntos mediante la partición al azar de las k clases del conjunto original.
- **Métodos aleatorios:** por último, otra familia de métodos multiclassificadores la forman las técnicas que introducen componentes aleatorios en los procesos de

aprendizaje para obtener diferentes multiclassificadores a partir de la misma evidencia. Por ejemplo, en [Parmanto et al. 1996] establecen aleatoriamente los pesos iniciales del algoritmo de propagación hacia atrás de redes neuronales. También se ha modificado el algoritmo de aprendizaje de árboles de decisión inyectando componentes aleatorios. El método *randomisation* [Dietterich 2000b] introduce aleatoriedad en la selección interna de la partición. Una estrategia similar se emplea en los multi-árboles de decisión [Estruch et al. 2003b]. En este caso el componente aleatorio se introduce en la selección de la partición donde continuar la generación de la estructura multi-árbol. Un multi-árbol se diferencia de un bosque (conjunto de árboles de decisión) en que los árboles de decisión comparten algunas ramas.

A continuación vamos a presentar con más detenimiento algunos de los métodos multiclassificadores que hemos comentado.

18.2.1 Bagging

Bagging [Breiman 1996; Quinlan 1996b] es un método simple y efectivo de generar conjuntos de clasificadores a partir de una evidencia. El término *bagging* deriva del mecanismo denominado *bootstrap aggregation*, mecanismo que genera subconjuntos de entrenamiento seleccionando aleatoriamente y con reemplazamiento (puede haber ejemplos repetidos) una muestra de m ejemplos de entrenamiento del conjunto original de entrenamiento formado por m ejemplos. Los subconjuntos nuevos son llamados *bootstrap replicates*. El método *bootstrap* lo vimos en el capítulo anterior utilizado para evaluación.

Se podría esperar que si se utiliza siempre el mismo algoritmo, los modelos aprendidos en cada iteración fuesen siempre similares, ya que existe un gran porcentaje de ejemplos comunes en todos los subconjuntos de entrenamiento. Sin embargo esto no siempre es cierto. Por ejemplo, los árboles de decisión son muy sensibles al conjunto de entrenamiento, es decir, pequeños cambios en el conjunto de entrenamiento provocan que el algoritmo de aprendizaje de árboles de decisión construya un árbol bastante diferente. Este comportamiento es bastante habitual si el conjunto de entrenamiento es limitado. A esta propiedad de un algoritmo de aprendizaje se le denomina inestabilidad o debilidad, y al algoritmo se le llama “aprendiz débil” (*weak learner*). Como hemos comentado, un ejemplo de algoritmo de aprendizaje inestable es el algoritmo de aprendizaje de árboles de decisión.

ALGORITMO Bagging(k :iteraciones, E :conjunto de ejemplos, A :Algoritmo de aprendizaje)
$i \leftarrow 1$ $C \leftarrow \emptyset$ REPITE Extrae una muestra E' de n ejemplos con reemplazamiento desde E $\quad // E$ contiene n ejemplos $m \leftarrow A(E')$ // Aprende un modelo con A utilizando el conjunto E' $\quad \{M\} \leftarrow \{M\} + m$ HASTA $i = K$ FIN ALGORITMO

Para clasificar un ejemplo e , se predice la clase de ese ejemplo para cada clasificador de C , y se selecciona la clase con mayor número de votos.

Figura 18.3. Algoritmo de bagging para clasificación.

Dado que hay un conjunto de clasificadores, la predicción de nuevos ejemplos se efectúa por votación mayoritaria. Es decir, se elige la clase con mayor número de votos entre todos los clasificadores. La Figura 18.3 describe el algoritmo de *bagging* para clasificación.

Bagging también puede utilizarse para combinar modelos de regresión. En este caso, lo que se hace es calcular la media de los valores predichos por cada uno de los modelos del conjunto. Esta media suele reducir el error cuadrático con respecto a un único modelo de regresión.

18.2.2 Boosting

El mecanismo que emplea *boosting* [Freund & Schapire 1996; Quinlan 1996b] para construir los componentes del conjunto se basa de la asignación de un peso a cada ejemplo del conjunto de entrenamiento. En cada iteración, *boosting* aprende un modelo que minimiza la suma de los pesos de los ejemplos clasificados erróneamente. Además, los errores de cada iteración se utilizan para actualizar los pesos de los ejemplos del conjunto de entrenamiento, de manera que se incremente el peso de los ejemplos errados y se reduce el peso de los ejemplos acertados. De esta forma se consigue que el modelo que se aprenda en la siguiente iteración otorgue más relevancia a los ejemplos que los modelos anteriores habían clasificado erróneamente.

La estrategia de *boosting* es bastante astuta, ya que construye los nuevos modelos tratando de corregir los errores cometidos previamente. Además, da más relevancia para la predicción de los ejemplos nuevos a los modelos que tienen un mejor comportamiento, en vez de situarlos a todos en el mismo nivel como hace *bagging*.

En realidad, existen muchas variantes del algoritmo básico de *boosting*, siendo probablemente *AdaBoost* [Freund & Schapire 1996] una de las versiones originales y todavía más populares. *AdaBoost* ha sido definido para problemas de clasificación (aunque se puede adaptar a regresión) a diferencia de *bagging*. La razón es que es necesario que el algoritmo de aprendizaje sea capaz de tratar con los pesos de los ejemplos. Esto se debe a que el error del clasificador se debe calcular como la suma de los pesos de los ejemplos clasificados erróneamente entre la suma de los pesos de los ejemplos totales. De esta manera, se intenta que el clasificador minimice el peso de los ejemplos erróneos, en vez del número de ejemplos erróneos.

La Figura 18.4 contiene un esquema del algoritmo *AdaBoost*. El algoritmo comienza asignando el mismo peso a todos los ejemplos de la evidencia. A continuación se inicia el bucle principal del algoritmo. El primer paso consiste en aprender un modelo utilizando para ello el método de aprendizaje desde la evidencia ponderada. Se estima un error del modelo, y si el error no está en el intervalo entre 0 y 0,5 se detiene el algoritmo. A continuación se actualiza el peso de los ejemplos correctamente. Para ello, cada ejemplo se multiplica por el valor $e/(e-1)$. Dado que e oscila entre 0 y 0,5, esta multiplicación disminuye el peso del ejemplo. Además, cuanto menor es el error del modelo, mayor es la disminución del peso del ejemplo. Finalmente, se almacena el modelo, y se realiza una normalización del peso de todos los ejemplos. Esta normalización consiste en multiplicar el peso de cada ejemplo por la suma de todos los pesos de los ejemplos en la iteración anterior y dividir por la suma de todos los pesos de los ejemplos de la presente iteración. De esta manera, se consigue que la suma de todos los pesos de los ejemplos se mantenga siempre

constante. Igualmente, dado que se ha reducido el peso relativo de los ejemplos clasificados correctamente, la normalización provoca que se aumente el peso relativo de los ejemplos que han sido mal clasificados.

ALGORITMO Boosting(k :iteraciones, E :conjunto de ejemplos, A :Algoritmo de aprendizaje)

```

 $i \leftarrow 1$ 
 $C \leftarrow \emptyset$ 
PARA CADA ejemplo  $e$  de  $E$ 
   $e_p \leftarrow p$  // Asigna a todos los ejemplos de  $E$  el peso  $p$ 
FIN PARA
REPITE
   $m \leftarrow A(E)$  // Aprende un modelo con  $A$  utilizando el conjunto con pesos  $E$ 
   $\varepsilon \leftarrow$  error de  $m$  sobre  $E$ 
  SI ( $\varepsilon = 0$ ) O ( $\varepsilon \geq 0,5$ ) ENTONCES termina bucle
  FIN SI
  PARA CADA ejemplo  $e$  de  $E$ 
    SI la clase de  $e$  se ha predicho correctamente
    ENTONCES  $e_p \leftarrow e_p * \varepsilon / (1-\varepsilon)$ 
    FIN SI
  FIN PARA
  Normaliza( $E$ ) // Normaliza los pesos de los ejemplos
   $\{M\} \leftarrow \{M\} \cup m$ 
HASTA  $i = K$ 
FIN ALGORITMO

```

Para clasificar un ejemplo e :

```

PARA CADA clase  $c$ 
   $c_p \leftarrow 0$  // asigna peso 0 a todas las clases
FIN PARA
PARA CADA modelo  $m$  de  $M$ 
   $t \leftarrow m(e)$  //  $t$  es la clase predicha por  $m$ 
   $c_t \leftarrow c_t - \log(\varepsilon / (1-\varepsilon))$  // actualiza peso de la clase  $t$ 
FIN PARA
DEVUELVE la clase con mayor  $c_p$  // peso

```

Figura 18.4. Algoritmo de boosting.

A diferencia del algoritmo *Bagging*, este algoritmo no siempre realiza las k iteraciones requeridas por el usuario, dado que considera un criterio de parada de acuerdo con el error e . Si el error es demasiado elevado (superior o igual a 0,5), o no se puede disminuir (igual a 0), el modelo se descarta y se detiene el bucle principal.

Como hemos comentado previamente, la clasificación de nuevas instancias no se realiza con una simple votación mayoritaria. En este caso, y tal como se puede observar en la Figura 18.4, se tiene en cuenta el error estimado de cada modelo para calcular la clase final de la nueva instancia. Para ello, se asigna a cada modelo un peso que debe ser mayor cuanto menor sea el error estimado. En concreto se utiliza $-\log(e/(e-1))$. Este valor oscila entre infinito y 0. Para predecir nuevos ejemplos, se suman los pesos de los modelos que eligen la misma clase, y se toma la clase que acumule el mayor valor.

18.2.3 Comparación de métodos

Existen varios trabajos que comparan experimentalmente varios métodos multiclasicadores. En [Maclin & Optiz 1997] se obtienen resultados similares comparando *bagging* y *boosting* con árboles de decisión y redes neuronales. Concluyen que *boosting* obtiene mejor precisión en general, pero en los casos de problemas con ruido, *bagging* es más robusto.

[Bauer & Kohavi 1999] presentan otro trabajo que compara empíricamente *bagging* y *boosting*. Los autores emplean en este caso dos algoritmos de aprendizaje: árboles de decisión y clasificadores bayesianos. Señalan como mejor método multiclasicador a *boosting*, en su implementación *Adaboost*. Este comportamiento, según comentan, se da de manera general, pero no en todos los casos. Según sus experimentos, *Adaboost* reduce un 26 por ciento el error medio de C4.5, y un 24 por ciento el error medio del clasificador bayesiano naive.

En [Dietterich 2000b] se han comparado experimentalmente *bagging*, *boosting* y *randomisation*. Este último método, introducido en el mismo trabajo, es un método multiclasicador basado en introducir componentes aleatorios en las decisiones internas del algoritmo de construcción de árboles de decisión. La idea consiste en seleccionar de manera aleatoria la partición donde continuar la construcción del árbol entre las veinte mejores particiones. De esta manera en cada iteración se genera un árbol diferente.

Los resultados de esta comparación destacan a *boosting* como el mejor algoritmo en problemas sin ruido, mientras *bagging* y *randomisation* obtienen niveles inferiores. En problemas con ruido, *bagging* obtiene los mejores resultados, seguido de *randomisation* y finalmente *boosting*.

18.2.4 Aprendizaje colaborativo (*co-learning*)

Una aplicación de la combinación de diferentes modelos para mejorar el comportamiento individual de estos modelos es el *co-learning* (también llamado *co-training*), o aprendizaje colaborativo. En el aprendizaje colaborativo, cuando un modelo tiene dudas sobre una determinada predicción, pide consejo a otros modelos para confirmar su predicción. Por ejemplo, un niño utiliza la vista para determinar si una persona es un hombre o una mujer, pero en el caso de que tenga dudas, escucha la voz de la persona para asegurar su pronóstico. En otra situación puede dudar atendiendo a la voz y entonces vuelve su mirada para mejorar el razonamiento. La vista y el oído funcionan, por tanto, como dos clasificadores distintos, que se apoyan uno al otro. En general, si tenemos varios clasificadores, cada uno con un valor de fiabilidad, cuando uno tiene dudas pregunta a los demás a ver si alguien puede clasificar el ejemplo con mayor fiabilidad y así aprender conjuntamente. Esta idea es especialmente útil en entornos multiagente, aunque en la práctica de la minería de datos, se ha demostrado que es una idea que mejora los resultados.

En [Blum & Mitchell 1998] se utiliza el aprendizaje colaborativo para mejorar las predicciones de los modelos mediante la utilización de ejemplos sin etiquetar (sin clase asignada). En muchos problemas existen gran cantidad de datos que no pueden ser utilizados por el aprendizaje por carecer de clase. El motivo de que no tengan clase asignada es que suele ser bastante costoso determinar la clase correcta de cada objeto, ya que normalmente esta tarea la realiza una persona. Un ejemplo de este tipo de problemas

es la categorización de páginas web. Podemos tener un conjunto de páginas web, cada una con una clase que indique su categoría (deportes, informática, personal, etc.), y entonces aprender un clasificador. Sin embargo, el número de páginas web en Internet es inmenso, por lo que el conjunto de que dispongamos no será más que una minúscula muestra. El método propuesto en [Blum & Mitchell 1998] utiliza vistas distintas de los mismos datos para aprender diferentes modelos. En el caso de la categorización de páginas web, dado un conjunto de páginas web podríamos aprender un modelo desde un conjunto de datos donde cada objeto contiene información sobre contenidos de la página web. Por otra parte, podríamos aprender otro modelo desde un conjunto de datos donde cada objeto contiene información de las páginas que apuntan a una determinada página web. Los modelos pueden utilizarse para etiquetar nuevos ejemplos sin clase, y posteriormente añadir al conjunto de entrenamiento los nuevos ejemplos que obtengan la etiqueta sin discrepancias entre los modelos.

Este algoritmo ha sido utilizado con resultados positivos para extraer conocimiento desde páginas web [Craven et al. 2000], y para clasificar correo electrónico [Kiritchenko & Matwin 2001].

18.3 Métodos de fusión

En la sección anterior hemos discutido varias estrategias de generar diferentes modelos con un algoritmo de aprendizaje a partir de una evidencia. Una vez construidos los modelos, la predicción de nuevos casos se realiza mediante la combinación o fusión de las predicciones de cada uno de los modelos. Por defecto, las versiones que hemos presentado de *bagging* y *boosting* implementan un método de fusión específico. Sin embargo, se han estudiado gran variedad de métodos para computar la predicción de nuevos ejemplos a partir de la predicción de los modelos del multiclásificador. Estos métodos se conocen como métodos de fusión.

Dado un problema con k clases, definimos el vector v_{ik} de k componentes como vector de probabilidades. Para un ejemplo y por cada modelo del multiclásificador calculamos un vector de probabilidades. Este vector recibe este nombre porque contiene las probabilidades de que el ejemplo sea de una determinada clase. Por lo tanto, la suma de los componentes del vector debe ser igual a 1. En realidad, también puede verse como un estimador de probabilidades (véase la Sección 6.2.1).

Si trabajamos con un multiclásificador de m miembros, se han definido varios métodos que permiten unificar los m vectores de probabilidades en un único vector α [Kuncheva 2002]. Este vector se utiliza para extraer la clase predicha, que será la clase que obtenga la mayor probabilidad. Veamos algunas estrategias de fusión:

- Suma:

$$\alpha = \sum_{j=1}^m v_j$$

- Media aritmética:

$$\alpha = \sum_{j=1}^m \frac{v_j}{m}$$

- Producto:

$$\alpha = \prod_{j=1}^m v_j$$

- Media geométrica: $\alpha = \sqrt[m]{\prod_{j=1}^m v_j}$
- Máximo: $\alpha = \max_{1 \leq j \leq m} (v_j)$
- Mínimo: $\alpha = \min_{1 \leq j \leq m} (v_j)$
- Mediana: $\alpha = \text{mediana}_{1 \leq j \leq m} (v_j)$

Varios trabajos han estudiado las propiedades de estos métodos de fusión. Por ejemplo [Kuncheva 2002] concluye que para problemas de dos clases, máximo y mínimo son las mejores estrategias, seguidas de la media aritmética.

Veamos el siguiente ejemplo de aplicación de estas estrategias de fusión. Supongamos que tenemos cuatro modelos para un problema de tres clases (a , b , y c):

- $v_1 = \{0,2, 0,6, 0,2\}$
- $v_2 = \{0,8, 0, 0,2\}$
- $v_3 = \{0,1, 0,4, 0,5\}$
- $v_4 = \{0,1, 0,7, 0,2\}$

Los métodos de fusión comentados darían los siguientes vectores fusión, con la clase que se seleccionaría en cada caso:

- Suma: $\{1,2, 1,7, 1,1\} \rightarrow b$
- Media aritmética: $\{0,3, 0,425, 0,275\} \rightarrow b$
- Producto: $\{0,0016, 0, 0,004\} \rightarrow c$
- Media geométrica: $\{0,2, 0, 0,2514\} \rightarrow c$
- Máximo: $\{0,8, 0,7, 0,5\} \rightarrow a$
- Mínimo: $\{0,1, 0, 0,2\} \rightarrow c$
- Mediana: $\{0,15, 0,5, 0,2\} \rightarrow b$

Un tipo de fusión similar es el que vimos en el Capítulo 6, cuando se realizaba binarización de problemas de clasificación de más de dos clases, en el que posteriormente se habían de combinar las predicciones en un clasificador conjunto. Métodos como el ECOC (*Error Correcting Output Code*) [Allwein et al. 2000] están relacionados con este tipo de fusión.

Respecto a la fusión de modelos de regresión, el contexto es bastante diferente ya que en este caso cada modelo devuelve un valor numérico, no un vector. En estos casos se utiliza la media (aritmética y geométrica) o bien la mediana.

18.3.1 Combinación bayesiana

Tal y como venimos comentando en este capítulo, la tasa de error de los modelos aprendidos puede reducirse generalmente si se aprende un conjunto de modelos y se combinan para realizar las predicciones. La teoría de aprendizaje bayesiano (véase el Capítulo 10) aporta una explicación teórica del comportamiento, y una manera óptima de combinar los modelos. Desde el punto de vista del aprendizaje bayesiano, utilizar tan sólo un modelo para realizar predicciones ignora la incertidumbre de trabajar con un conjunto de ejemplos finito que representa sólo una parte del problema. Por lo tanto, deberíamos utilizar todos los posibles modelos del espacio del problema para realizar las predicciones.

Además cada modelo debería tener un peso asociado a la probabilidad que tenga de acierto, de manera que los modelos más fiables cuenten más que los más inciertos. Esta probabilidad (*posterior*) se calcula a partir de la probabilidad del modelo antes de utilizar los datos (*prior*), y de la probabilidad de los datos dado el modelo o verosimilitud (*likelihood*). Esta manera de combinar modelos se denomina combinación bayesiana de modelos (*Bayesian Averaging Models*, BMA) [Hoeting et al. 1999].

Consideremos una evidencia D y tres hipótesis h_1 , h_2 y h_3 , cuyas probabilidades a posteriori se han calculado: $P(h_1|D)=0.4$, $P(h_2|D)=0.3$, $P(h_3|D)=0.3$. Para una próxima observación h_1 la clasifica como positiva (*true*), mientras que h_2 y h_3 la clasifican como negativa (*false*). La mejor clasificación de una nueva instancia se obtiene combinando las predicciones de las distintas hipótesis consideradas:

$$P(v_j | D) = \sum_{h_i \in H} P(v_j | h_i)P(h_i | D)$$

Para el ejemplo anterior:

$$P(h_1 | D) = 0.4, \quad P(h_2 | D) = 0.3, \quad P(h_3 | D) = 0.3$$

$$P(false | h_1) = 0, \quad P(false | h_2) = 1, \quad P(false | h_3) = 1$$

$$P(true | h_1) = 1, \quad P(true | h_2) = 0, \quad P(true | h_3) = 0$$

Por lo tanto:

$$P(false | D) = \sum_{h_i \in H} P(false | h_i)P(h_i | D) = 0.6$$

$$P(true | D) = \sum_{h_i \in H} P(true | h_i)P(h_i | D) = 0.4$$

Teóricamente, la combinación bayesiana de modelos representa la manera óptima de combinar modelos si se dispone de la información correcta de las probabilidades del modelo a priori ($P(h)$), y de los datos dado el modelo ($P(D|h)$). Sin embargo, todos estos datos no son siempre fácilmente computables en problemas reales de minería de datos, por lo que la combinación bayesiana no ha sido ampliamente utilizada. Normalmente, y dado que es imposible conocer los valores exactos de las probabilidades, se trabaja con aproximaciones. Estas aproximaciones causan que se pierdan parcialmente las propiedades de la combinación bayesiana. Por ejemplo, en [Domingos 2000] se compara la combinación bayesiana con otros métodos como *bagging*. Los resultados son bastante desfavorables para la combinación bayesiana, en parte debido a que se le aprecia una tendencia a sobreajustar los datos de entrenamiento. Como conclusión, y tal y como se comenta en este trabajo, la aplicación de la combinación bayesiana de manera conveniente se considera como un problema abierto en el aprendizaje automático.

18.4 Métodos híbridos

Los métodos híbridos son métodos de aprendizaje formados a partir de componentes de otros métodos de aprendizaje. Los métodos híbridos buscan combinar varios métodos de aprendizaje. Por ejemplo, en la Sección 13.3.6, hemos introducido las RBF (*Radial Basis Functions*). Estas funciones utilizan conjuntamente un método de agrupamiento (*K medias*) y posteriormente una red neuronal de una capa (Perceptrón).

En esta sección describimos varias de las técnicas de aprendizaje calificadas como técnicas híbridas: *stacking* y *cascading*.

18.4.1 Stacking

Stacking [Wolpert 1992] es un método multiclásificador bastante simple basado en la combinación de modelos generados por diferentes algoritmos de aprendizaje. Dado que cada modelo se aprende con un mecanismo de aprendizaje distinto se consigue que los modelos del conjunto sean diferentes.

Una primera y simple versión del *stacking* consiste en aprender un modelo con uno de los algoritmos que deseemos utilizar. Si, por ejemplo, queremos utilizar redes neuronales, árboles de decisión y aprendizaje bayesiano, generaríamos un modelo con cada técnica desde la misma evidencia. La clasificación se podría realizar mediante una votación mayoritaria. Sin embargo, este esquema, aunque es sencillo, tiene un problema: la votación mayoritaria funciona relativamente bien cuando todos los modelos combinados tienen una precisión aceptable, y este caso no siempre se puede asegurar en todos los clasificadores.

Una posible solución al problema de la votación mayoritaria sería intentar conocer cuándo un determinado modelo se comporta mejor para tenerlo en cuenta a la hora de clasificar nuevos ejemplos. Esta idea se implementa en una versión más refinada del *stacking* consistente en la introducción de una fase de *meta-aprendizaje* en el propio modelo multiclásificador. El propósito de esta fase es aprender la mejor manera de combinar los modelos base del multiclásificador. En concreto, la fase de meta-aprendizaje recibe como entrada las predicciones de los modelos base del multiclásificador junto con la clase del problema a aprender.

En la Figura 18.5 podemos apreciar el esquema general de *stacking*. En este esquema se ha utilizado un árbol de decisión para la fase del meta-aprendizaje, aunque se podría utilizar cualquier otro tipo de método de aprendizaje. Sin embargo, dado que la mayor parte de la labor de aprendizaje se realiza por los modelos de base, el meta-modelo sólo realiza tareas de selección de clase, por lo que se pueden utilizar algoritmos bastante simples como el aprendizaje de árboles de decisión.

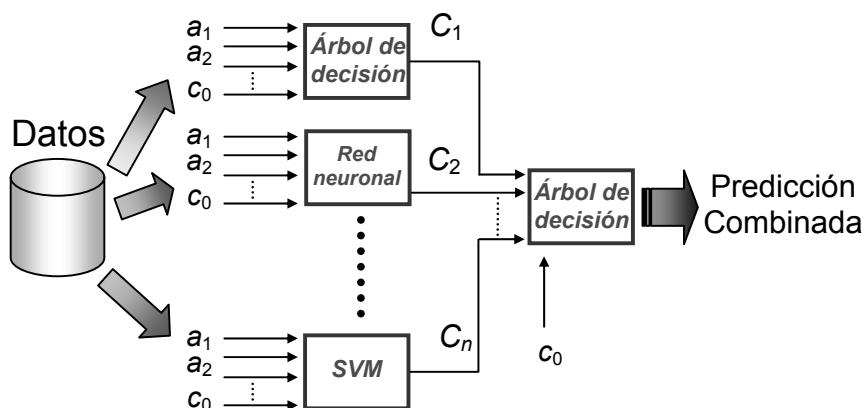


Figura 18.5. Esquema de general de stacking.

Una cuestión a plantearse es cómo entrenar el meta-modelo. Podríamos utilizar los mismos datos con los que se han construido los modelos de base, sin embargo esto entraña el

el mismo problema que discutíamos en la Sección 17.2.1, el meta-modelo estaría demasiado ajustado a los datos de entrenamiento. Podemos subsanar este problema utilizando la misma estrategia que se emplea en la evaluación: utilizar un conjunto de datos diferente. La idea sería dividir los datos de entrenamiento en dos partes: una parte se utilizaría para aprender el conjunto de modelos base, y otra parte para aprender el meta-modelo. A este segundo conjunto se le llama “validación”. Otra opción sería utilizar validación cruzada.

Para clasificar un nuevo ejemplo con *stacking*, se clasifica con cada uno de los modelos base y las clases seleccionadas son introducidas al meta-modelo, quien decide la clase final.

Finalmente, *stacking* puede ser también utilizado en problemas de regresión. En este caso se mantendría el mismo esquema básico, pero los modelos base y el meta-modelo serían modelos de regresión.

18.4.2 Cascading

Cascading [Gama & Bradzil 2000] es un método que permite mejorar las características de los árboles de decisión mediante la incorporación de nuevas particiones basadas en otras técnicas de aprendizaje, como por ejemplo regresión lineal o redes neuronales.

Como se vio en la Sección 11.2.1, los árboles de decisión clásicos son univariantes, es decir sólo permiten realizar particiones sobre un único atributo de entrada al mismo tiempo. Estas particiones sólo permiten dividir el espacio de representación en espacios cuadriculares, (véase Figura 11.4, en la página 286). Esta limitación en la representación limita la capacidad de los árboles de decisión para ajustarse a los datos de entrenamiento.

La estrategia de *cascading* es utilizar otros métodos de aprendizaje (regresión lineal, análisis discriminante) sobre algunos de los atributos para crear nuevos atributos artificiales. Estos nuevos atributos son también utilizados por el árbol de decisión para crear modelos. Los atributos artificiales representan conceptos intermedios, y permiten que el algoritmo de aprendizaje extienda la representación cuadriculada mediante la inclusión de divisiones oblicuas en el espacio de representación. A este tipo de árboles de decisión se le llaman multivariantes [Brodley & Utgoff 1995].

Un esquema de cascading puede verse en la Figura 18.6. Aunque en este caso representamos la inclusión de atributos artificiales sólo en la fase inicial del aprendizaje del árbol, existen otras versiones que permiten la inclusión de nuevos atributos durante la construcción del árbol, es decir, en los nodos internos del árbol de decisión.

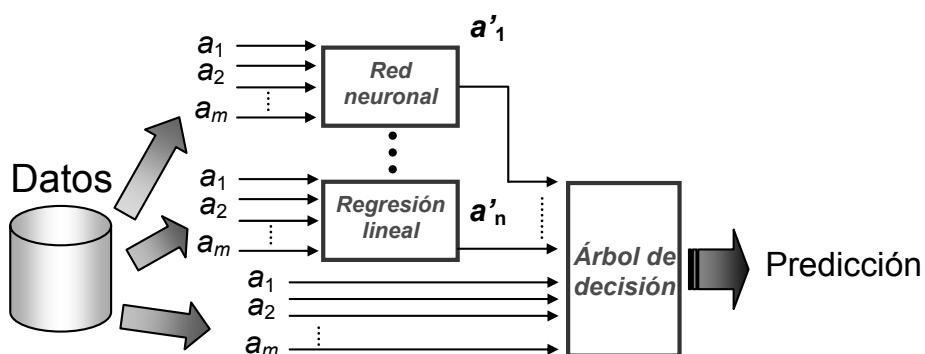


Figura 18.6. Esquema de general de cascading.

En realidad, la diferencia entre *cascading* y *stacking*, cuando en el *stacking* se utiliza un árbol de decisión como meta-modelo, es bastante tenue como se puede observar si comparamos la Figura 18.5 y la Figura 18.6. Al igual que otros métodos anteriores de combinación, *cascading* puede también utilizarse para regresión.

Ejemplos

Por último, vamos a mostrar con un ejemplo el uso de las técnicas desarrolladas en esta sección. Para ello utilizamos los paquetes de minería de datos WEKA y SPSS Clementine. El problema que vamos a tratar es el “German Credit”, que se encuentra descrito con mayor detalle en el Apéndice B. Este conjunto de datos recoge 1.000 ejemplos de clientes de una entidad de préstamos. Todas las instancias están etiquetadas con los valores “Bueno” o “Malo”, indicando si el cliente se considera como fiable o no. El objetivo es realizar un modelo que nos permita predecir si serán fiables los futuros demandantes de créditos según este conjunto de ejemplos anteriores etiquetados manualmente.

A parte de la clase (indicador de rentabilidad) existen 20 atributos. Estos 20 atributos (siete numéricos y 13 nominales) representan indicadores de los clientes, tales como: propósito del crédito, cantidad solicitada, situación laboral, etc.

Comenzaremos con WEKA (versión 3.2). Esta herramienta proporciona gran cantidad de métodos de aprendizaje, entre ellos varios basados en combinación de modelos, o métodos híbridos. En principio, vamos a utilizar tres métodos simples: árboles de decisión (C4.5 en su versión de java J48, paquete *WEKA.classifiers.J48.j48*), aprendizaje bayesiano (Bayes simple, paquete *WEKA.classifiers.NaiveBayes*), y aprendizaje basado en instancias (el vecino más cercano, paquete *WEKA.classifiers.IB1*). La siguiente tabla contiene la precisión media de un total de diez ejecuciones de validación cruzada de diez particiones, es decir un total de 100 iteraciones. Utilizamos para ello el entorno *experimenter* que permite automatizar de manera sencilla la realización de experimentos.

NaiveBayes	IB1	J48
74,98%	72,38%	71,98%

Los resultados indican que el mejor método es en este caso el aprendizaje bayesiano, seguido por el aprendizaje basado en instancias, y por último, árboles de decisión.

Vamos a incrementar la precisión de los árboles de decisión mediante el uso de métodos multiclasicadores. WEKA provee la implementación (entre otros) de los métodos *bagging* (paquete *WEKA.classifiers.bagging*) y *boosting* (paquete *WEKA.classifiers.AdaBoostM1*). En concreto, se ha implementado la variante *AdaBoost* de *boosting*.

La Figura 18.7 representa cómo se comporta la precisión obtenida dependiendo del número de iteraciones con que se ejecutan los métodos multiclasicadores. Como se puede observar, los dos métodos permiten incrementar considerablemente la precisión. En este problema, *bagging* obtiene mejores resultados. El incremento en ambos casos no es constante con respecto al número de iteraciones, sino que se produce con mayor rapidez en las primeras iteraciones, obteniéndose el pico de precisión en 80 iteraciones (75,01 por ciento *bagging* y 74,74 por ciento *boosting*).

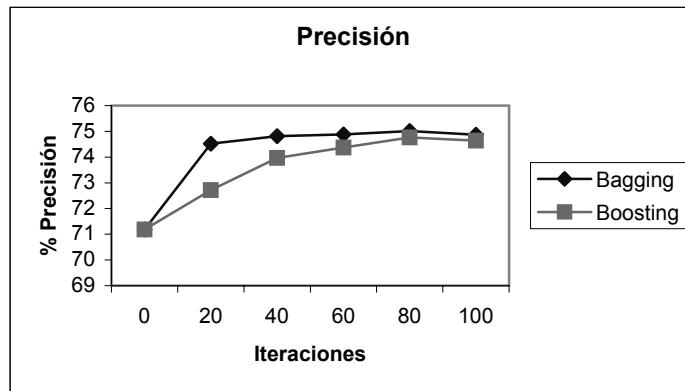


Figura 18.7. Comparación de precisión entre boosting y bagging.

La Figura 18.8 contiene una comparación en el tiempo de aprendizaje para los dos métodos multiclasicadores. Como es de esperar, *bagging* presenta un tiempo de aprendizaje totalmente lineal con respecto al número de iteraciones. *Boosting* podría no tener este comportamiento, ya que el algoritmo está diseñado de manera que se contemple un criterio de parada determinado por el error del último modelo aprendido. Sin embargo, observando el comportamiento asintótico en la figura, se puede concluir que el algoritmo ha realizado todas las iteraciones en la mayoría de los casos.

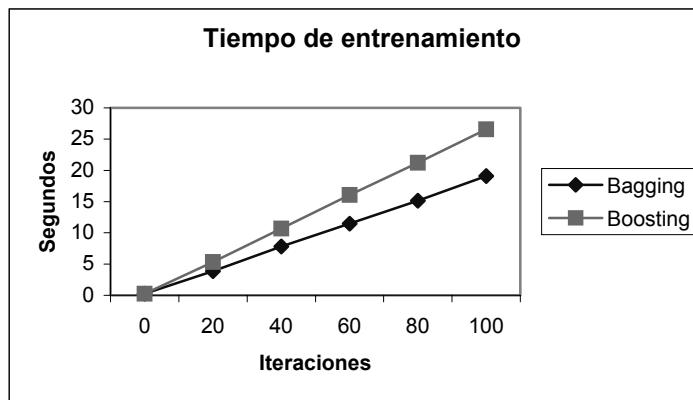


Figura 18.8. Comparación de tiempo de entrenamiento entre boosting y bagging.

Por último, vamos a utilizar un método híbrido: *stacking*. WEKA suministra para esta técnica el paquete WEKA.Classifiers.Stacking. Para la configuración de este paquete debemos especificar qué algoritmos de aprendizaje vamos a utilizar para construir los modelos de base y con qué técnica vamos a aprender el meta-modelo. Para este experimento utilizamos para los modelos de base las tres técnicas usadas anteriormente (*IB1*, *NaiveBayes*, y *J48*). Para el meta-modelo utilizamos *J48*. Los resultados se muestran en la siguiente tabla. Se muestra la precisión media de diez ejecuciones de una validación cruzada de diez particiones. Los resultados son bastante satisfactorios para la técnica *stacking*, ya que es capaz de mejorar la precisión media de todos los métodos por separado.

NaiveBayes	IB1	J48	Stacking
74,98%	72,38%	71,98%	75,24%

Veamos a continuación cómo se pueden utilizar métodos multiclasicadores o métodos híbridos en el sistema Clementine. Este sistema, con respecto a estas características, no es tan completo como el WEKA, ya que en el conjunto de técnicas básicas no se encuentra ninguna técnica de este tipo. Sin embargo, dado la capacidad de Clementine de derivar nuevas técnicas de aprendizaje mediante la utilización de las herramientas que proporciona, es posible que el usuario construya flujos que implementen métodos multiclasicadores o métodos híbridos.

Una excepción al discurso anterior la encontramos en la implementación de C5.0 incluida en Clementine. Si desplegamos la ventana de configuración de este componente en la versión en español, veremos una opción “Utilizar aumento”. Esta opción se refiere a *boosting*. Si marcamos esta opción, se habilita una caja de edición donde podemos indicar el número de iteraciones que deseemos. La Figura 18.9 corresponde a la evolución de la precisión obtenida de la ejecución de *boosting* sobre C5.0 dependiendo del número de iteraciones. La precisión es muy similar, como es lógico, a la obtenida en la Figura 18.7.

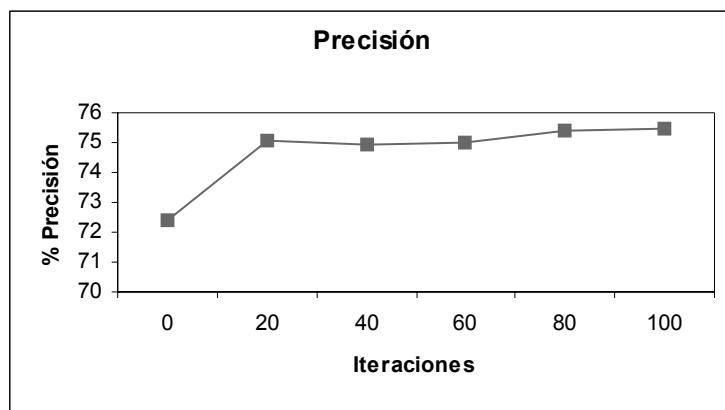


Figura 18.9. Evolución de la precisión de boosting sobre C5.0.

Como se ha comentado previamente, Clementine permite la construcción de métodos multiclasicadores mediante la utilización de componentes básicos del sistema: filtros, modelos, operaciones con registros, etc. Veamos un ejemplo. Deseamos construir tres modelos con técnicas de aprendizaje diferentes y posteriormente combinar las predicciones de estos tres modelos para incrementar el porcentaje de aciertos. Para la construcción de los modelos empleamos los siguientes componentes: dos algoritmos de aprendizaje de árboles de decisión (CART y C5.0), y regresión logística. Para la evaluación de los modelos dividimos el conjunto de datos en dos subconjuntos de idéntico tamaño. Uno se utilizará para el entrenamiento y el otro para el test.

La Figura 18.10 muestra el diseño de este experimento. Existen dos flujos diferentes. El primer flujo construye los tres modelos a partir de los datos de entrenamiento. El segundo corresponde a la evaluación de los modelos y es bastante más complejo, ya que es donde se combinan las predicciones de los tres modelos. Veamos paso a paso este flujo. Primero unimos las tres clases con el componente “combinar” de “operaciones con registros”. A continuación, y dado que la clase tiene dos valores 1 (bueno) y 2 (malo), creamos un nuevo

atributo que representa la suma de las tres clases. Este paso se realiza utilizando el componente “derivar” del conjunto de componentes de “operaciones con campos” (en la figura aparece con el nombre “suma”). Este componente permite crear nuevos atributos a partir de atributos existentes. Para poder sumar correctamente las clases, hemos de darles un tipo numérico, esto es lo que hace el componente “tipo” situado antes de la derivación. En esta situación, tendremos un campo que representa la suma de las tres clases, por lo tanto el valor mínimo será 3, y el máximo 6. De esta forma, si queremos implementar la votación mayoritaria, el valor frontera que marca qué clase se selecciona es 4,5. El siguiente componente (“combinada”) añade un nuevo campo que toma valor 2 si la suma es mayor que 4,5, y valor 1 en otro caso. El componente (“filtro”) elimina todos los campos a excepción de la clase combinada y la clase original. Finalmente, se indica mediante un componente “tipo” cuál es la entrada y salida, y se analizan los resultados.

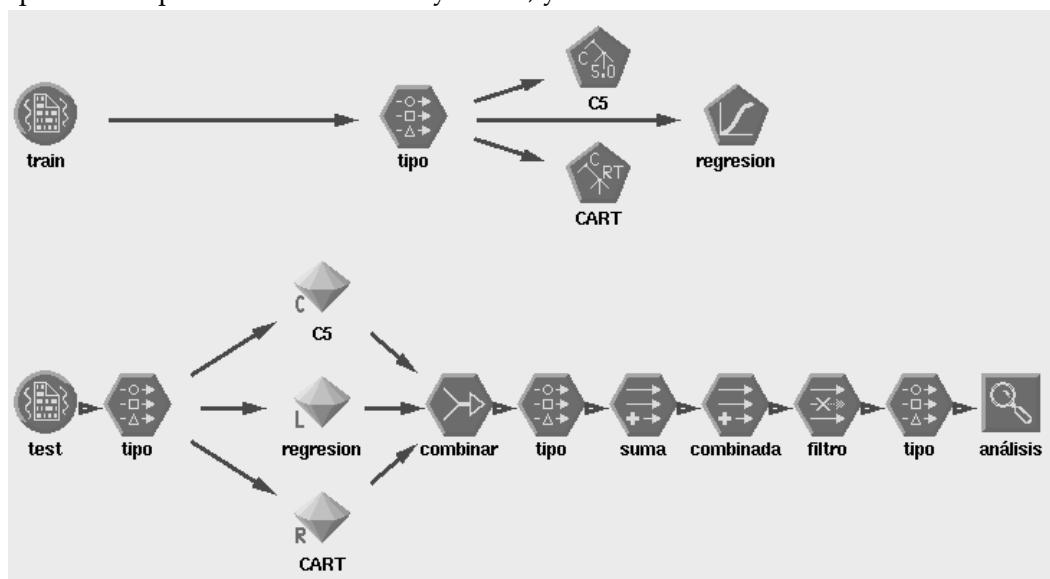


Figura 18.10. Combinación de modelos con Clementine.

Se ha optado por una votación mayoritaria como método de fusión, sin embargo, también es posible definir estrategias más complejas, como por ejemplo votación ponderada. Para ello sólo debemos modificar los componentes que manipulan la clase combinada, pudiendo utilizar los operadores que suministra Clementine en el lenguaje *Clementine Language for Expression Manipulation* (CLEM).

La siguiente tabla muestra los resultados de los experimentos. En este caso, y tal y como se esperaba, la combinación realizada por votación mayoritaria mejora la precisión de cada uno de los modelos por sí solos.

CART	C5.0	Reg. Logística	Combinación
69,80%	69,60%	72,20%	72,80%

En la siguiente matriz podemos observar la correlación⁵⁷ entre los tres métodos por separado. Podemos ver que la correlación es positiva pero no demasiado importante:

⁵⁷ Correlación calculada mediante el método de Pearson [StatSoft 2004].

	Reg. Logística	CART
C5.0	0,292	0,377
Regresión Logística	-	0,470

Por último, vamos a ver cómo se podría implementar la técnica de *stacking* con Clementine. Dada la complejidad intrínseca de esta técnica, el diseño no es simple ya que se necesitan tres flujos: uno para el aprendizaje de los modelos base, otro para el aprendizaje del meta-modelo, y por último, un flujo para la evaluación. Para este ejemplo, vamos a utilizar los tres métodos de aprendizaje vistos en el caso anterior para los modelos base: CART, C5.0 y regresión logística. Para el meta-modelo utilizamos C5.0. La Figura 18.11 muestra el diseño del *stacking*.

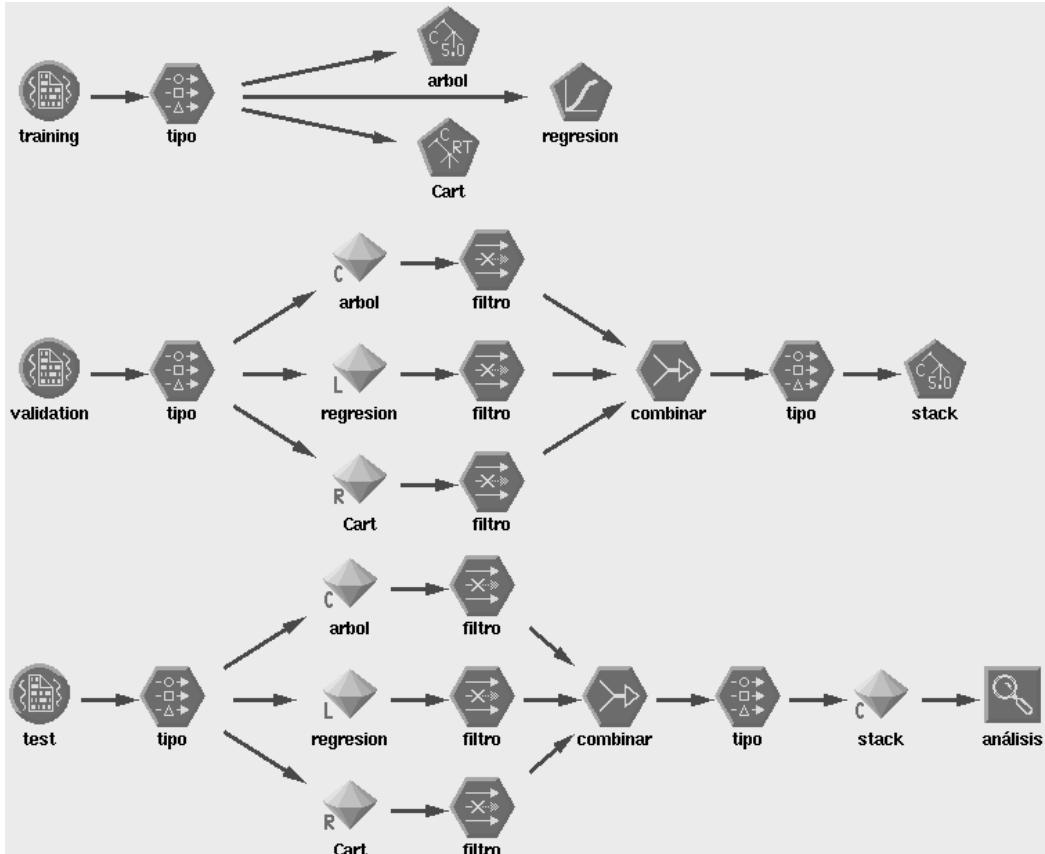


Figura 18.11. Implementación de Stacking en Clementine.

Para el diseño del *stacking* hemos dividido el conjunto de 1.000 datos en tres subconjuntos: entrenamiento (400 datos), validación (200 datos) y test (400 datos). Tal y como hemos comentado, el diseño de *stacking* debe contener tres flujos. El primero (parte superior de la figura) es el más simple, siendo su objetivo el aprendizaje de los modelos base a partir del conjunto de datos de entrenamiento.

El segundo flujo (situado en la parte central) se encarga del aprendizaje del meta-modelo. Para ello debemos, primero filtrar todos los campos a excepción de las clases, y posteriormente unir los cuatro campos (la clase original más la predicha por los tres

modelos) mediante el componente combinar. Es importante renombrar la clase original en el componente filtro, ya que en otro caso, habría conflictos en la parte de evaluación, al existir nombres de campos duplicados. Finalmente, definimos los tipos (como salida la clase original, y como entrada el resto) y se aprende el meta-modelo con los datos de validación.

El último flujo corresponde a la fase de evaluación. Este flujo es similar al anterior, a excepción de la última parte, ya que aquí se utiliza el meta-modelo para clasificar los datos de test.

La siguiente tabla contiene los resultados del experimento. La precisión de *stacking* mejora la precisión obtenida de los árboles de decisión, pero no llega a la precisión obtenida por el modelo de la regresión logística. Este bajo rendimiento del *stacking* podría ser debido a que el número de datos es bastante limitado para realizar una partición en tres subconjuntos.

CART	C5.0	Reg. Logística	Stacking
69,25%	68,50%	72,50%	71,75%

Para concluir el capítulo, comentamos que el diseño de nuevos métodos multiclasicadores es una de las áreas que ha despertado interés en los últimos tiempos en la comunidad de aprendizaje automático. Igualmente, la combinación de métodos de aprendizaje para crear nuevos métodos híbridos es otra de las áreas de investigación más activas. Todo este esfuerzo suscita que cada año se desarrollos nuevos métodos cada vez más efectivos.

Capítulo 19

INTERPRETACIÓN, DIFUSIÓN Y USO DE MODELOS

La fase de minería de datos construye un conjunto de modelos que explican de manera general los datos de entrenamiento. Normalmente, incluso después de la evaluación y validación pertinentes los modelos aprendidos no se pueden utilizar directamente, sino que necesitan una fase de refinamiento que permita concretar cuál es el conocimiento que aportan y, en esa situación, cómo se puede utilizar en la toma de decisión final. Este capítulo se centra en la transformación, difusión y aplicación de los modelos generados como resultado de la fase de aprendizaje, dentro del proceso de extracción de conocimiento.

En este capítulo veremos cómo extraer reglas comprensibles a partir de modelos no comprensibles, cómo exportarlos en formato estándar, como es el PMML, para integrarlos en otros sistemas y aplicaciones, cómo se integran los modelos en la toma de decisiones, y el diseño de campañas. Finalmente veremos brevemente el problema de la actualización de modelos y de su uso para simulación.

19.1 Introducción

En el Capítulo 2 introdujimos el proceso de la extracción de conocimiento, presentando para ello, cada una de las fases de este proceso. En este proceso, seguramente la fase más importante es la correspondiente a la de minería de datos, fase en la que realmente se produce el descubrimiento o extracción de los patrones existentes en los datos. Sin embargo, la realización de una buena fase de minería de datos puede ser inútil si no se aplican correctamente las dos siguientes fases: la fase de evaluación / interpretación y la fase de difusión / uso.

La fase de evaluación / interpretación (vista en el Capítulo 17) se encarga de medir la calidad de los modelos aprendidos, así como de introducir técnicas, como por ejemplo la

visualización de modelos, o visualización posterior, que permitan interpretar a los usuarios finales el conocimiento que aportan los modelos aprendidos.

La última fase se denomina fase de difusión / uso, y tiene como fin el empleo de forma correcta del modelo aprendido en el contexto de la aplicación real y de los usuarios para los cuales se inició el proceso de extracción de conocimiento.

En este capítulo introducimos varios conceptos que tienen que ver con estas fases del proceso de la extracción de conocimiento desde bases de datos. Vamos a discutir principalmente aspectos relacionados con el uso de modelos en situaciones reales: diseño de campañas de marketing, estándares de intercambio o representación de información, integración con sistemas de ayuda a la toma de decisiones, etc. Para ello intentaremos tomar un enfoque eminentemente práctico, dadas las características propias de esta fase donde en muchas ocasiones, son los propios usuarios quienes tienen un papel importante o incluso deben realizar estas tareas.

19.2 Extracción de reglas comprensibles

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

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

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

El problema de la pérdida de comprensibilidad afecta también a los métodos multiclasicadores o a los métodos híbridos. Estos métodos (vistos en el Capítulo 18) combinan o fusionan diferentes técnicas de aprendizaje con el fin de incrementar la precisión de los modelos aprendidos.

Por lo tanto, la comprensibilidad no puede ser obviada a la hora de elegir los métodos de minería de datos. En el caso de que este factor sea importante, se deben seleccionar

métodos de aprendizaje de reglas (reglas de decisión, ILP), o equivalentemente, métodos que aprendan modelos que puedan ser expresados como reglas (árboles de decisión, métodos difusos). También los modelos estadísticos simples (por ejemplo la regresión lineal) suelen ser bastante comprensibles. En todo caso, es también importante incluir técnicas que permitan reducir la complejidad de los modelos como por ejemplo podar reglas o filtrar atributos no relevantes.

Existen casos en los cuales los métodos de aprendizaje basados en reglas no son capaces de aprender modelos con los requisitos requeridos (precisión, tiempo de aprendizaje...). Para estos casos, se puede utilizar la siguiente estrategia: aprender un modelo con el método de aprendizaje que obtenga los mejores resultados, y entonces intentar modelar el comportamiento del modelo con un conjunto de reglas. A esta estrategia se le llama extracción de reglas comprensibles.

La extracción de reglas comprensibles se ha utilizado fundamentalmente desde redes neuronales. Por ejemplo, el sistema TREPAN [Craven 1996] construye un árbol de decisión basándose en una red neuronal. Para ello utiliza un criterio de partición basado en la fidelidad con respecto a la red neuronal, un criterio de parada especial y consultas a la red neuronal.

Otra estrategia que permite capturar el comportamiento de un modelo no comprensible en un conjunto de reglas es el método CMM (*Combined Multipled Models*). La descripción de este método puede encontrarse en [Domingos 1997]. Esta técnica es simple y general, ya que se puede aplicar a cualquier tipo de modelo independientemente del método de aprendizaje que se haya empleado. Dado un modelo no comprensible *A*, la idea consiste en utilizar un conjunto de ejemplos sin clase (o no etiquetados) generados aleatoriamente, que son etiquetados con el modelo *A* de manera que reflejan o capturan su comportamiento. Cuanto mayor sea el tamaño del conjunto de ejemplos mejor se plasmará el comportamiento del modelo. A partir de este conjunto de ejemplos podemos utilizar un algoritmo de aprendizaje que genere un meta-modelo comprensible *B*, de manera que tendrá una semántica similar a la del modelo *A* no comprensible. Probablemente, y debido al propio hecho de que utilizar reglas limita el poder de representación, el nuevo meta-modelo *B* obtendrá una precisión ligeramente inferior al modelo original *B*. Podemos ver un esquema del método en la Figura 19.1.

Generar el conjunto de datos de manera aleatoria tiene la ventaja (si el problema no lo limita) de poder tener tantos componentes como se deseé. No obstante, es importante que para la generación de los ejemplos aleatorios se conserve la distribución original de los datos de entrenamiento, ya que muchas técnicas de aprendizaje son bastante sensibles a cambios en la distribución en los datos. Por otra parte, se debe también utilizar el conjunto de datos de entrenamiento junto con el conjunto de datos inventado para aprender el meta-modelo, ya que no olvidemos que es la única referencia que se posee del problema real.

En [Domingos 1997] se estudia experimentalmente este algoritmo en 26 datasets, utilizando *bagging* como método de aprendizaje no comprensible, y C4.5 como método donde aplicar *bagging*, y posteriormente, para aprender el meta-modelo comprensible. Los resultados demuestran que CMM es capaz de retener el 60 por ciento de la mejora obtenida por *bagging* con respecto a C4.5. Referente a la comprensibilidad (medida usando el número de atributos y reglas) CMM obtiene modelos que son de dos a seis veces más complejos que los modelos aprendidos con C4.5.

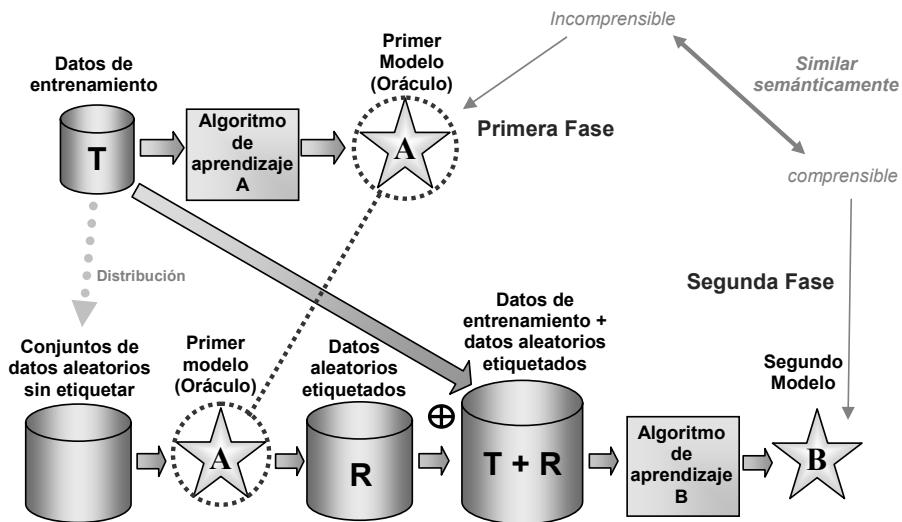


Figura 19.1. Esquema del método CMM.

Una ventaja muy importante de este método es que se puede implementar de manera sencilla. Incluso se puede realizar utilizando las interfaces o componentes que suministran algunos de los paquetes de minería de datos. Por ejemplo, podemos utilizar la técnica CMM para extraer un conjunto de reglas comprensibles utilizando para ello el paquete Clementine. Para un ejemplo del uso de esta técnica en Clementine, se puede consultar [Estruch et al. 2003a].

19.3 Visualización posterior

La tarea de minería de datos produce una serie de modelos cuya fácil interpretación por parte del usuario, tal y como hemos comentado en la sección anterior, es clave para el éxito final del proceso de extracción de conocimiento desde bases de datos. Con este fin, se han definido diferentes métodos y técnicas que permiten la visualización de los resultados de la etapa de aprendizaje. La visualización de modelos permite que los usuarios puedan identificar fácilmente y de manera directa los patrones más significativos que ha descubierto el modelo. También permiten representar los modelos junto a los datos. Asimismo, muchos de los métodos de visualización permiten que los propios usuarios modifiquen los modelos para refinarlos o adaptarlos según su conocimiento o circunstancias del ámbito de aplicación.

Gran parte de los métodos de minería de datos producen modelos que pueden ser expresados con lenguaje natural o bien mediante expresiones matemáticas. Sin embargo, dado que una representación gráfica permite, por lo general, una mejor interpretación y comprensión por parte de los seres humanos, se han estudiado diferentes representaciones visuales de modelos resultantes de fases de aprendizaje. En concreto, se han definido representaciones gráficas para (entre otros) árboles de decisión, reglas de asociación y redes bayesianas. En otros casos, en especial si sólo hay dos o tres dimensiones, se pueden mostrar los patrones sobre los mismos datos originales, como hemos visto en muchas figuras de la Parte III de este libro.

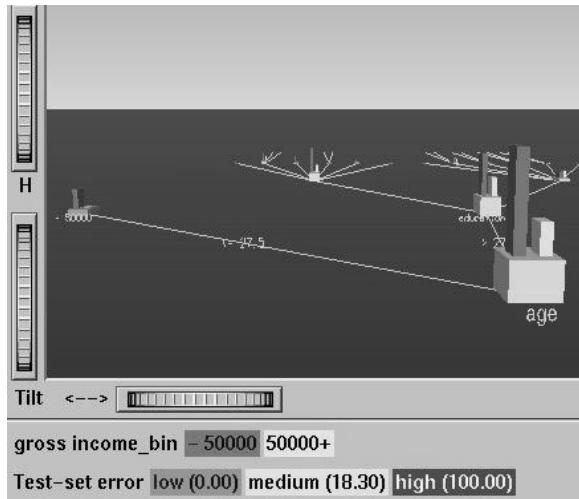


Figura 19.2. Visualización del árbol de decisión en MLC++ mediante la herramienta MineSet.

Los árboles de decisión son, seguramente, los modelos que permiten una representación visual más clara gracias a la propia estructura en árbol de los modelos. Un árbol de decisión puede verse como un grafo parcialmente ordenado donde los nodos sólo tienen un parente. Dado que un árbol puede ser de gran tamaño, muchas herramientas permiten mostrar segmentos parciales del árbol, empezando por las ramas superiores, y desplegar las partes que el usuario seleccione hasta llegar hasta las hojas. La Figura 19.2 muestra el árbol de decisión que genera la herramienta Mineset (véase el Apéndice A).

Las redes bayesianas representan el conocimiento cualitativo de un modelo mediante un grafo dirigido acíclico, por lo que su representación gráfica es directa. El grafo expresa las relaciones de dependencia/independencia entre los diferentes atributos de un problema. En el Capítulo 10 se trata con detenimiento las redes bayesianas. Un ejemplo de una red bayesiana para los datos del archivo "Ingresos.d" (descrito en la Sección 10.3, página 260) lo podemos ver en la Figura 19.3 (que es la misma Figura 10.7).

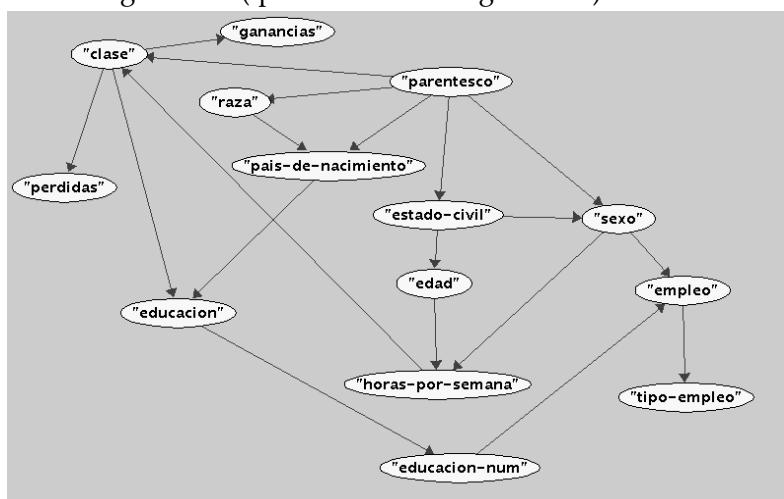


Figura 19.3. Red bayesiana aprendida en el dominio "Ingresos.d".

En esta red bayesiana podemos observar las relaciones directas e indirectas entre las variables correspondientes al dominio utilizado. Entre ellas, por ejemplo, podemos destacar las relaciones entre "Educación", "Educación-num", "Empleo" y "Tipo de empleo" que se observan conectadas en la red resultante; también podemos observar el camino existente entre las variables "Pérdidas", "Ganancias" y la variable "Clase" (Ingresos).

Aunque las reglas de asociación no son directamente representables gráficamente, sí es posible expresar mediante una representación visual llamada malla las relaciones entre los ítems o los conjuntos de ítems. Por ejemplo, Clementine proporciona este tipo de gráfico. Un ejemplo de una malla puede verse en la Figura 19.4 (que es la misma que la Figura 9.4). Detalles sobre la interpretación de estos gráficos se puede encontrar en la Sección 9.6.

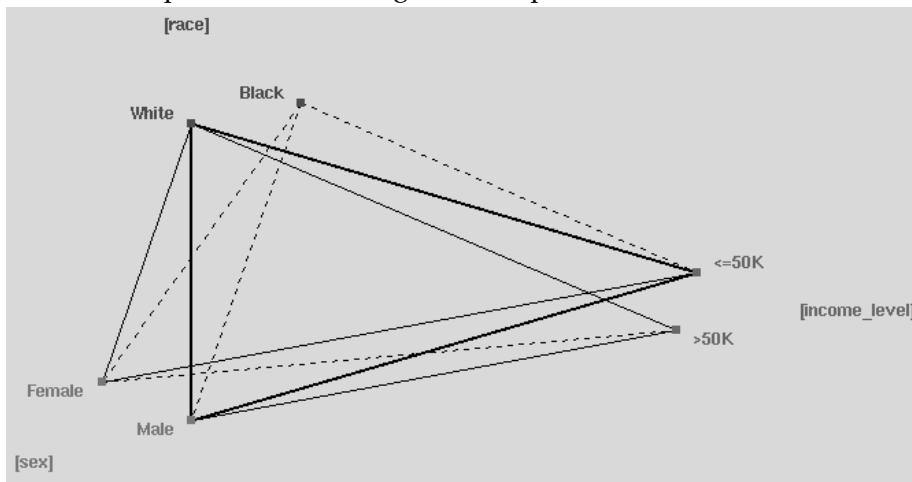


Figura 19.4. Gráfico de malla.

Por otra parte, también es posible representar gráficamente modelos de regresión o clasificación, siempre que los problemas tengan menos de tres dimensiones (atributos), mostrando directamente el modelo en el espacio formado por los atributos. Incluyendo los datos en la representación, podremos observar la calidad de los modelos para ajustarse a los datos. Por ejemplo, la Figura 19.5 muestra una representación simplificada de un modelo con dos discriminadores lineales (como los vistos en la Sección 7.8.3).

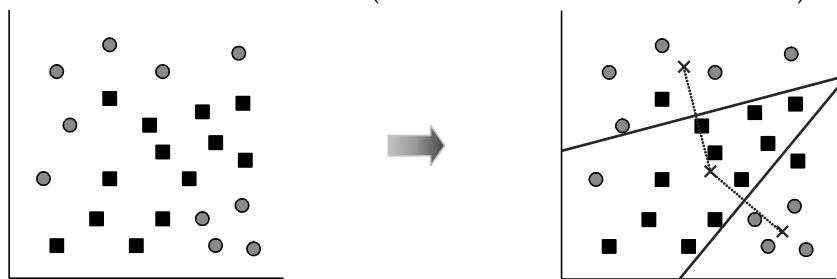


Figura 19.5. Representación simplificada de un modelo con dos discriminadores lineales.

Una estrategia similar se puede adoptar en modelos de agrupamiento, pero en este caso en vez de los modelos, se visualizan los centros de los grupos aprendidos. Un ejemplo de esta representación lo podemos ver en la Figura 19.6 (que es la misma que la Figura 16.8).

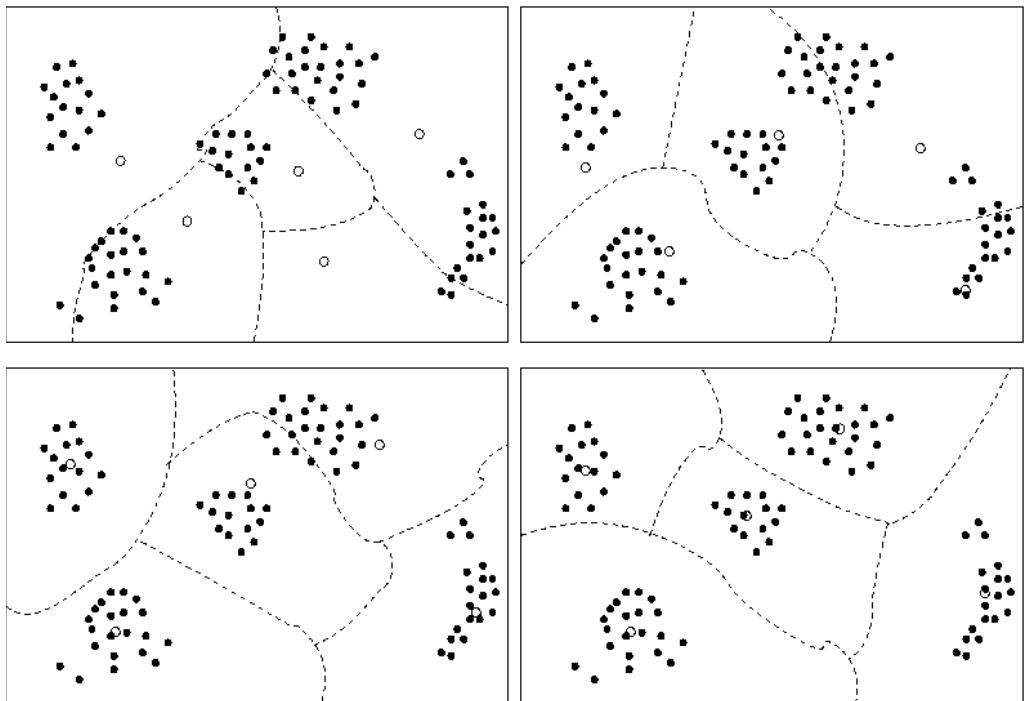


Figura 19.6. Ejemplo de evolución de los prototipos y grupos formados con el método K medias.

En este caso podemos ver la evolución de los grupos formados por el algoritmo K medias (véase la Sección 16.2.2) dependiendo del número de iteraciones. En las dos últimas representaciones la restricción es el número de dimensiones; cuando el número de dimensiones (atributos) es mayor que tres, estas representaciones no son posibles directamente.

Por otra parte, dado que los métodos jerárquicos de agrupamiento (véase la Sección 16.2.3) se basan en la construcción de un árbol, se puede representar este árbol en un gráfico llamado dendograma. La Figura 19.7 (que es la misma que la Figura 16.10) representa un dendograma.

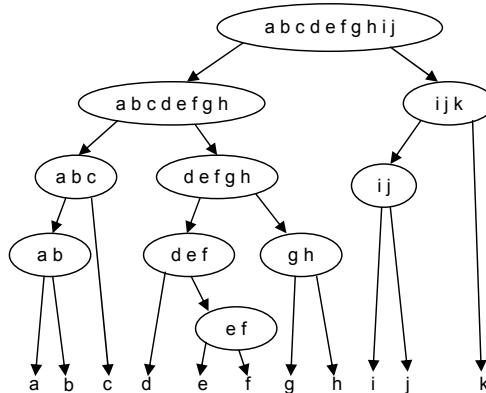


Figura 19.7. Ejemplo de un árbol de agrupamiento (dendograma).

Una descripción detallada de la visualización y presentación posterior de modelos puede encontrarse en el Capítulo 11 de libro [Berthold & Hand 2003] y el Capítulo 20.1 del libro [Klösgen & Zytkow 2002]

19.4 Intercambio y difusión de modelos: estándares de representación

Recientemente existe un interés creciente en la definición y uso de estándares para intercambiar información entre aplicaciones de distintos tipos, o bien, distintos fabricantes. La definición del lenguaje XML (*eXtensible Markup Language*) ha impulsado su utilización para la creación de lenguajes estándar para intercambio de información. XML es un lenguaje de marcas definido por el consorcio *World Wide Web Consortium* (W3C, <http://www.w3.org/>), basado en el estándar ISO SGML (*Standard Generalized Markup Language*). XML puede considerarse como un meta-lenguaje ya que permite la definición de lenguajes de marcas para diferentes tipos de documentos. Con ese fin XML utiliza las DTD (*Document Type Definition*) para describir las características (elementos u otros constructores) de cada tipo de documentos.

En el área de minería de datos este interés se ha plasmado en la definición y uso de estándares de representación y uso de conocimiento o más concretamente, de modelos estadísticos o de minería de datos. La motivación de este interés es clara, facilitar el intercambio de conocimiento. La importancia del intercambio de información puede verse con el siguiente ejemplo. Supongamos que una empresa A (por ejemplo una agencia de seguros) ha recogido información y ha obtenido patrones y modelos a partir de ellos sobre el riesgo de los clientes durante los últimos cinco años. Esta información es muy importante a la hora de determinar la cuantía de los seguros. Una nueva empresa B de la misma rama se establece en una zona de características similares. El comercio B no se puede permitir esperar unos años para recabar esa información y mejorar la rentabilidad de los seguros, por lo que decide comprar ese conocimiento a la empresa A. Sin embargo, existen multitud de sistemas de minería de datos o sistemas de ayuda a la toma de decisiones desarrollados por empresas diferentes. Todos ellos con formatos diferentes e incompatibles. La cooperación o integración entre estos sistemas es imposible sin la existencia de estándares independientes que definan la forma de expresar la información a compartir.

Otra motivación para la definición de estándares de intercambio es facilitar el intercambio de datos y conocimiento, ya sean conjuntos de ejemplos, matrices de coste, modelos, etc., entre aplicaciones diferentes de minería de datos. Por ejemplo, supongamos que tenemos una aplicación A que permite realizar la combinación de modelos mediante *stacking* (véase el Capítulo 18). Por otra parte, tenemos una aplicación B que aprende redes neuronales a partir de ejemplos. La única manera de que podamos integrar una red neuronal generada por la aplicación A en un modelo *stacking* de la aplicación B, es que la A pueda exportar las redes neuronales en un estándar que la aplicación B puede comprender y utilizar.

```
<?xml version="1.0" ?>
<PMML version="2.1" >
<Header copyright="www.dmg.org" description="A small binary tree model"/>
<DataDictionary numberOfFields="5" >
```

```

<DataField name="temperature" optype="continuous"/>
<DataField name="humidity" optype="continuous"/>
<DataField name="windy" optype="categorical" >
  <Value value="true"/>
  <Value value="false"/>
</DataField>
<DataField name="outlook" optype="categorical" >
  <Value value="sunny"/>
  <Value value="overcast"/>
  <Value value="rain"/>
</DataField>
<DataField name="whatdo" optype="categorical" >
  <Value value="will play"/>
  <Value value="may play"/>
  <Value value="no play"/>
</DataField>
</DataDictionary>
<TreeModel modelName="golfing" functionName="classification">
  <MiningSchema>
    <MiningField name="temperature"/>
    <MiningField name="humidity"/>
    <MiningField name="windy"/>
    <MiningField name="outlook"/>
    <MiningField name="whatdo" usageType="predicted"/>
  </MiningSchema>
  <Node score="will play">
    <True/>
    <Node score="will play">
      <SimplePredicate field="outlook" operator="equal"
        value="sunny"/>
      <Node score="will play">
        <CompoundPredicate booleanOperator="and" >
          <SimplePredicate field="temperature"
            operator="lessThan" value="90" />
          <SimplePredicate field="temperature"
            operator="greaterThan" value="50" />
        </CompoundPredicate>
        <Node score="will play" >
          <SimplePredicate field="humidity"
            operator="lessThan" value="80" />
        </Node>
        <Node score="no play" >
          <SimplePredicate field="humidity"
            operator="greaterOrEqual" value="80" />
        </Node>
      </Node>
      <Node score="no play" >
        <CompoundPredicate booleanOperator="or" >
          <SimplePredicate field="temperature"
            operator="greaterOrEqual" value="90"/>
          <SimplePredicate field="temperature"
            operator="lessOrEqual" value="50" />
        </CompoundPredicate>
      </Node>
    </Node>
    <Node score="may play" >
      .....
    </Node>
  </TreeModel>

```

```

</Node>
</Node>
</TreeModel>
</PMML>

```

Figura 19.8. Representación parcial en PMML de un árbol de decisión para el problema de jugar a tenis.

Existen varias iniciativas para establecer estándares de intercambio de información en minería de datos. Pero, sin duda alguna, la iniciativa más destacada es PMML (*Predictive Model Markup Language*). La definición de este lenguaje (entre otras tareas similares) está siendo llevada a cabo por el denominado *Data Mining Group* (<http://www.dmg.org/>). Este grupo es un consorcio de empresas importantes en el campo de la minería de datos (y del software en general) tales como: IBM, Microsoft, Oracle, SAP, SAS, SPSS...

PMML se basa en XML. El lenguaje soporta, entre otros, los siguientes tipos de modelos: regresión polinomial y general, árboles de decisión, agrupamiento basado en centros y en distribuciones, reglas de asociación y secuencias, y redes neuronales. Cada tipo de modelo se expresa mediante un documento DTD o un XML Schema.

Veamos un ejemplo. La Figura 11.8 contiene parte de la descripción en PMML de un árbol de decisión para un problema de si es posible jugar a tenis dependiendo de las condiciones climáticas. Este problema está inspirado en el problema visto en la página 30. En realidad, y tal y como se puede ver en la figura, el esquema XML propuesto para los árboles de decisión permite definir estructuras de decisión más generales, como por ejemplo particiones con condiciones compuestas y operadores lógicos.

Del texto de la figura podemos distinguir tres partes: cabecera, diccionario de datos y el modelo del árbol. La cabecera contiene información respecto al archivo PMML. En la siguiente parte *<DataDictionary>* se incluyen los atributos del modelo así como sus posibles valores. Finalmente se encuentra la parte exclusiva del modelo a describir, en este caso *<TreeModel>*. Con la etiqueta *<MiningSchema>* define qué atributos se van a utilizar en el aprendizaje, y cuál será la clase. El árbol se construye de manera recursiva utilizando componentes nodos *<Node>*. Cada nodo contiene información sobre condiciones que debe satisfacer el ejemplo para entrar en ese nodo junto con una predicción de la clase.

La definición de PMML constituye un paso importante en el camino de facilitar el intercambio de información entre aplicaciones y usuarios en la minería de datos. Al ser una iniciativa relativamente reciente, las aplicaciones no han incorporado masivamente la opción de importar/exportar información en PMML. Sin embargo, y fundamentalmente por la cantidad e importancia de las empresas que están detrás del *Data Mining Group*, es de esperar que las nuevas versiones incorporen cada vez con más frecuencia estas posibilidades.

19.5 Integración con la toma de decisiones

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

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

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

Otro tipo de modelo que incorpora mejor la información probabilística sobre las relaciones entre las variables del contexto del problema son las redes de decisión o diagramas de influencia [Zhang et al. 1994]. Estas redes pueden considerarse como una extensión de las redes bayesianas (véase el Capítulo 10). Las redes de decisión contienen tres tipos de nodos: nodos aleatorios (en forma circular u ovoidal) para expresar variables aleatorias, nodos de decisión (rectángulos) que indican las opciones a considerar y los nodos de utilidad (rombos) que representan una función sobre las variables. Una solución a una red de decisión es un conjunto de estrategias que permiten maximizar la utilidad esperada. La Figura 19.9 contiene una red de decisión sobre la elección entre un tratamiento o la realización de una prueba secundaria. Anteriormente se ha realizado una primera prueba, y entonces se puede optar entre pasar directamente al tratamiento o bien asegurar el diagnóstico mediante la realización de una prueba adicional. La figura muestra las interacciones entre las variables, las decisiones y los nodos de utilidad; no representa un diagrama de flujo de dichas decisiones. Por tanto, está más cercano a un red bayesiana que a un árbol de decisión.

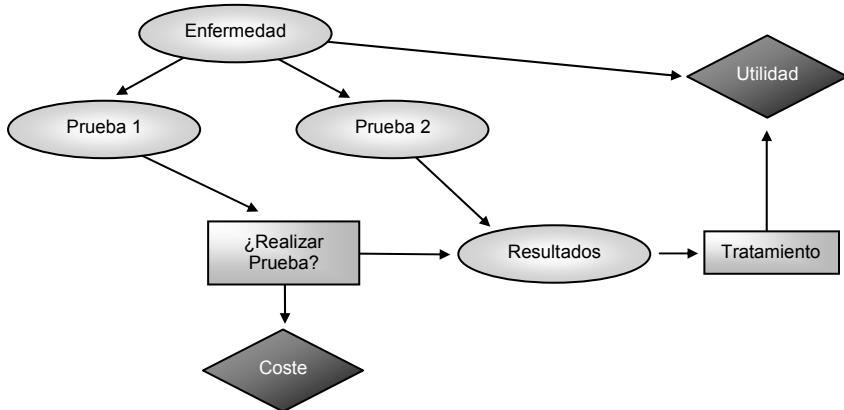


Figura 19.9. Red de decisión.

En los últimos años se ha popularizado el uso de la minería de datos en los sistemas de ayuda a la toma de decisiones. Como comentamos en el Capítulo 3, los sistemas de ayuda a la toma de decisiones que iban incorporando técnicas como cubos de datos, análisis

estadístico, almacenes de datos, OLAP, etc., han añadido también a esta lista componentes basados en la minería de datos.

Normalmente, el típico escenario para aplicar técnicas de minería de datos para ayudar en la toma de decisiones se sitúa en un contexto empresarial. Por ejemplo, si queremos determinar a qué cliente debemos enviar propaganda de un determinado producto de manera que se maximice el número de ventas con respecto al gasto en envío de publicidad. El primer paso consiste en trasladar el problema al ámbito de la minería de datos, es decir generar un conjunto de datos con características de los clientes y su decisión con respecto a la compra. Si somos capaces de encontrar este conjunto de datos, podemos emplear una herramienta de aprendizaje para generar un modelo que nos prediga si un cliente comprará el producto a partir de los datos del cliente. El siguiente paso consiste en trasladar el modelo al ámbito de la aplicación, para que apoye al comercial o automatice la decisión de enviar o no, propaganda a un determinado cliente.

19.5.1 Integración de los costes

Una situación más realista para este tipo de ejemplo, es el aprendizaje sensible al coste véase la Sección 17.2.2. En este tipo de aprendizaje no todos los errores tienen el mismo peso, sino que se suele utilizar la matriz de coste, donde se expresa el coste asociado que tiene cada una de las posibles combinaciones que se dé en la predicción de un caso. Por ejemplo, una matriz de coste para el ejemplo del envío de propaganda, podría ser:

		Real	
		No compra	Compra
Estimado	No compra	0 €	3 €
	Compra	10 €	-100 €

El caso más positivo se da cuando el sistema predice que el cliente va a comprar el producto, por lo tanto se le envía propaganda, y posteriormente lo compra. Esta situación representa un beneficio de 100 euros (coste negativo). El peor caso es cuando el sistema envía propaganda (se determina el cliente como potencial), y no se produce la compra (diez euros). Destaca el hecho de que si el sistema no predice un cliente como potencial comprador cuando posteriormente realiza la compra, en este caso sí tiene asociado un coste (tres euros). Obviamente, para la empresa esta situación es la ideal, ya que se ahorra el coste de la propaganda y además realiza la venta. Sin embargo, no debemos olvidar que estamos introduciendo los costes estimados para el entrenamiento del modelo, y al fin y al cabo esa situación representa un error del modelo por lo que debe ser penalizado.

Si disponemos de la matriz de coste (la mejor situación), lo más adecuado es utilizar esta matriz en el aprendizaje del modelo. Una aproximación sencilla es la estratificación, por submuestreo (*oversampling*) o submuestreo (*undersampling*) [Stolfo & Chan 1998]. Este método permite adaptar un algoritmo de aprendizaje al contexto de coste sin modificar el propio algoritmo, alterando la frecuencia de las clases como vimos en la Sección 17.2.2. Otro método similar, es decir, que no necesita modificar el algoritmo de aprendizaje, es *metacost* [Domingos 1999].

A parte de los métodos genéricos, existen, por lo general, versiones para el aprendizaje sensible al coste de cada método de aprendizaje. Por ejemplo, en los árboles de decisión se ha estudiado modificar el criterio de partición de manera que tenga en cuenta el coste. Sin

embargo, para este algoritmo de aprendizaje, se ha comprobado [Bradford et al. 1998] que una solución bastante efectiva consiste en aprender el árbol de decisión de la manera habitual, es decir utilizando los criterios de partición más conocidos, y posteriormente, para cada hoja del árbol asignar la clase de manera que, en vez de reducir el número de los errores de la hoja, se reduzca el coste asociado a esa hoja.

No obstante, conocer la matriz de costes de un determinado problema no es, por desgracia, la situación más habitual. La estimación de costes supone en la mayoría de los casos un estudio previo que requiere de la inversión de tiempo y dinero extra. La situación más común es conocer detalles de la matriz de coste en cuanto se entra en la fase de la aplicación del modelo. Cuando desconocemos la matriz de coste en tiempo de aprendizaje, podemos aplicar el análisis ROC (véase la Sección 17.2.2). El análisis ROC provee técnicas que, dado un conjunto de clasificadores, nos permiten seleccionar el subconjunto de clasificadores que tendrá un comportamiento óptimo (menor coste) para un determinado contexto de coste. El análisis ROC permite también evaluar los modelos aprendidos de manera más independiente respecto al contexto de coste mediante el valor AUC (área bajo la curva ROC), también visto en la Sección 17.2.2. Por lo tanto, si desconocemos la matriz de coste cuando se esté aprendiendo el modelo, podemos hacer lo siguiente: aprendemos un conjunto de modelos, ya sea utilizando diferentes técnicas de aprendizaje, o bien variando la configuración del algoritmo de aprendizaje; seguidamente seleccionamos el subconjunto de clasificadores que tienen un comportamiento óptimo para cualquier contexto de coste, eliminando los que siempre van a ser subóptimos mediante el análisis ROC; de este subconjunto de clasificadores utilizaremos el modelo con mayor AUC para empezar a aplicarlo con casos reales; si durante la aplicación del modelo podemos estimar la matriz de costes, cambiaremos el modelo a utilizar por el modelo que obtenga mejores prestaciones para la matriz de coste estimada de entre el subconjunto de modelos óptimos.

Finalmente, una vez hemos pasado la fase de aprendizaje y se ha seleccionado el modelo a ser aplicado con casos reales, antes de llevar a cabo ese paso, es recomendable que un experto evalúe la validez del modelo, es decir que el conocimiento extraído es coherente⁵⁸. Consideremos el siguiente caso en el cual utilizamos una base de datos sobre los clientes que poseen un libro sobre álgebra para determinar a qué otros clientes se envía propaganda sobre ese determinado libro. Una buena parte de los ingenieros en informática han comprado el libro, por lo que el modelo podría determinar que los ingenieros en informática son buenos candidatos a comprar el libro. Sin el análisis de un experto podríamos recomendar el libro de álgebra a todos los clientes que son ingenieros en informática. Sin embargo, probablemente este colectivo compró el libro en sus años de estudio para aprobar las asignaturas relacionadas con el álgebra, y probablemente tengan ese libro con varios dedos de polvo. Por tanto, muy probablemente pocos ingenieros en informática comprarán el libro. Este tipo de reglas, aparentemente buenas pero falsas en la práctica, sólo se pueden eliminar o bien corregir (en este caso sería mejor considerar los estudiantes de informática, no los titulados) mediante la supervisión de un experto.

Como conclusión a este apartado, hemos de destacar que si se desea llevar los resultados de la minería de datos a la práctica se deben tener en cuenta las características reales

⁵⁸ Para ello es necesario que el modelo sea fácilmente comprensible, por ejemplo un árbol de decisión.

del problema. Es decir, los datos de entrenamientos deben ser reales y representar al máximo el problema a resolver. El modelo debe tener en cuenta el contexto de coste en el cual se va a aplicar, y debe contemplar las dificultades que pueden tener los datos reales: atributos faltantes, datos desfasados. Además, es necesario que personas expertas en el problema asesoren y confirmen que el conocimiento aprendido es válido y útil.

En cuanto a bibliografía recomendada para ampliar conocimientos sobre este tema, podemos destacar el libro [Mladenic et al. 2003] que está centrado en la integración de minería de datos con la ayuda a la toma de decisiones, así como el Capítulo 21 de [Klösgen & Zytkow 2002].

19.5.2 Diseño de campañas

Una de las aplicaciones de la minería de datos más extendidas, y por lo tanto, más estudiadas, es el diseño de campañas para el envío de publicidad o de ofertas (normalmente por correo ordinario) a clientes de empresas o instituciones. En esta aplicación, el uso de técnicas de minería de datos puede prever el comportamiento de los clientes, es decir su interés o no sobre el producto determinado. La utilización de estas previsiones puede optimizar la realización de las campañas, invirtiendo dinero en publicidad en los clientes que el modelo prediga que pueden ser receptivos para la campaña.

En esta sección vamos a detallar con un ejemplo varias técnicas para configurar una campaña de manera óptima a partir de un modelo de estimación de la probabilidad de respuesta. Como referencia de este tema en particular, o en general del tema del tratamiento de las relaciones con los clientes o CRM (*Customer Relationship Management*) y el uso de la minería de datos en este problema puede consultarse [Berry & Linoff 2000].

Supongamos que una empresa de venta de productos informáticos por catálogo posee una base de datos de clientes, con datos sobre estos clientes. Esta empresa ha incorporado un nuevo producto a su catálogo, un mando de piloto para ser utilizado en programas de simulación de vuelo. La empresa, dado que este producto es innovador, ha editado un folleto sobre el producto para realizar una campaña de promoción mediante el envío de correo a sus clientes y, de esta manera, estimular su venta.

Una primera solución sería enviar el folleto a todos sus clientes. Sin embargo, dado que este producto tiene unas características especiales, no sería la mejor solución. Por ejemplo, no sería muy útil enviar propaganda de este producto a restaurantes. Por el contrario, podría molestarles, además del gasto del folleto y del correo.

Por lo tanto, una mejor opción sería una campaña selectiva. Para ello utilizaremos técnicas de minería de datos para poder predecir la respuesta de un determinado cliente al envío de la propaganda y utilizar esta información para optimizar el diseño de la campaña.

El primer paso es la selección de una muestra de clientes de la base de datos. Para no aprender un modelo sesgado, la muestra ha de ser aleatoria, y de tamaño suficiente. Una vez seleccionada la muestra, se envía la propaganda del producto a un grupo de clientes, y posteriormente, pasado un tiempo prudencial, se etiqueta cada cliente con la clase correspondiente (ha comprado el producto o no).

Tras etiquetar la muestra, podemos utilizar alguna de las técnicas de aprendizaje para generar un modelo de respuesta a partir del cliente. En esta aplicación nos interesa utilizar técnicas de clasificación suave, es decir que asignen a cada ejemplo (cliente) no la clase

predicha, sino una estimación de la probabilidad de respuesta de ese cliente. Por ejemplo, el “Restaurante Tío Canya” puede tener un valor 0,1 mientras que el “Playcenter Galaxy” puede tener un valor de 0,85.

La información sobre las estimaciones de respuesta de los clientes puede ser utilizada para la generación del llamado gráfico de respuesta acumulada. Estos gráficos nos indican qué porcentaje de las posibles respuestas vamos a obtener dependiendo del porcentaje de envíos que realicemos sobre la población total. Para construir estos gráficos se ordenan los ejemplos por su *ranking* o prorrato. A continuación se realiza una suma acumulativa entre los ejemplos y se divide cada ejemplo por la suma total de probabilidades. Finalmente cada ejemplo se multiplica por 100. El proceso para diez ejemplos lo podemos ver en la siguiente tabla:

Ejemplos	Probabilidad	Suma Acum.	Normalizada	Norm.*100
0	0	0	0	0
1	0,9	0,9	0,24456522	24,4565217
2	0,87	1,77	0,48097826	48,0978261
3	0,6	2,37	0,64402174	64,4021739
4	0,4	2,77	0,75271739	75,2717391
5	0,3	3,07	0,83423913	83,423913
6	0,25	3,32	0,90217391	90,2173913
7	0,2	3,52	0,95652174	95,6521739
8	0,1	3,62	0,98369565	98,3695652
9	0,05	3,67	0,99728261	99,7282609
10	0,01	3,68	1	100

Esta información también se puede representar gráficamente. Para ello, el número de ejemplos se normaliza sobre 100 y la serie se incluye en un gráfico de porcentajes. Para el ejemplo anterior podemos ver el resultado en la Figura 19.10. El eje *x* indica el porcentaje de clientes a los cuales se les envía la propaganda. El eje de la *y* indica el porcentaje de clientes que van a comprar el producto sobre el total que comprarán el producto, o equivalentemente, el porcentaje sobre las ventas totales estimadas (siempre refiriéndonos a la población que recibe la propaganda, en este caso 3,68). En el gráfico podemos ver dos series: sin modelo y utilizando el modelo de predicción de respuesta. Obviamente los resultados sin modelo corresponden a una selección al azar, por lo tanto si enviamos propaganda al 10 por ciento de los clientes, recibirán la propaganda el aproximadamente el 10 por ciento de los clientes que comprarán el producto. Con el modelo de predicción de respuesta se mejora considerablemente el resultado. Por ejemplo, enviando tan sólo al 30 por ciento de los clientes con mayor *ranking*, alcanzaríamos el 65 por ciento de las ventas totales. Cabe destacar que utilizando sólo el gráfico de respuesta acumulada es imposible conocer la cantidad de ventas, ya que el gráfico indica porcentaje pero no totales. La estimación de ventas se puede aproximar utilizando la suma de la probabilidad estimada para los clientes. Para el ejemplo que estamos viendo, las ventas estimadas serían 3,68.

Podemos plantear aún mejor la campaña si conocemos algunos datos extra sobre los costes del problema. En concreto, necesitamos determinar tres costes: el coste de planificación de la campaña, el coste de envío de publicidad por unidad y finalmente el beneficio por unidad vendida. El primer coste, coste de planificación de la campaña, recoge

todos los costes iniciales de la campaña independientemente del número de envíos: coste de diseño del folleto, estudio de mercado, etc. Con estos costes estimados, y un conjunto de clientes prorrateados por el modelo de estimación de respuesta, podemos realizar un cálculo sobre los costes o beneficios dependiendo del número de envíos.

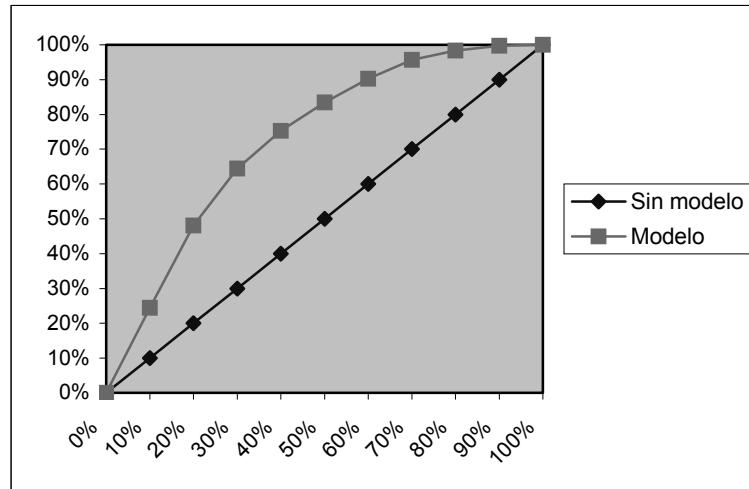


Figura 19.10. Gráfico de respuesta acumulada.

Veamos un ejemplo. Consideremos que tenemos una base de datos de 120.000 clientes en nuestra empresa de venta de catálogos informáticos. Utilizamos 20.000 seleccionados al azar para aprender un modelo que nos estime la respuesta (campaña exploratoria). Con este modelo calculamos la probabilidad de respuesta de cada uno de los restantes 100.000 clientes y los ordenamos según su *ranking*. Supongamos que el gráfico de respuesta acumulada que se obtiene de esta ordenación es igual al de la Figura 19.10. Enviando propaganda al 20 por ciento de los clientes obtendremos el 48 por ciento de las ventas totales. Supongamos también que de los 20.000 clientes el 30,68 por ciento compraron el producto. El coste inicial de la campaña lo estimamos en 10.000 euros, y el coste de envío de cada folleto 1,5 euros. Por cada producto vendido ganamos tres euros (sin tener en cuenta el gasto del envío del folleto). Con estos datos podemos realizar una estimación de los gastos, ingresos y beneficios de la campaña dependiendo del número de clientes a los que envíemos propaganda. El estudio lo tenemos en la siguiente tabla:

Envíos	Compras	Gastos	Ingresos	Beneficios
0	0	10.000 €	0 €	-10.000 €
10000	9000	25.000 €	27.000 €	2.000 €
20000	17700	40.000 €	53.100 €	13.100 €
30000	23700	55.000 €	71.100 €	16.100 €
40000	27700	70.000 €	83.100 €	13.100 €
50000	30700	85.000 €	92.100 €	7.100 €
60000	33200	100.000 €	99.600 €	-400 €
70000	35200	115.000 €	105.600 €	-9.400 €
80000	36200	130.000 €	108.600 €	-21.400 €
90000	36700	145.000 €	110.100 €	-34.900 €
100000	36800	160.000 €	110.400 €	-49.600 €

El beneficio estimado dependiendo del número de envíos puede mostrarse en un gráfico. De esta manera podemos apreciar la evolución del beneficio y seleccionar la mejor configuración. El gráfico muestra la evolución de los beneficios con dos configuraciones de coste: la estimación de coste comentada anteriormente (configuración 1), y otra configuración similar donde el coste de campaña es de 20.000 euros, el coste de envío es de 0,8 euros, y el beneficio por venta 2,5 euros (configuración 2). Para la configuración 1, el mayor beneficio se obtiene enviando publicidad al 30 por ciento de los clientes con más probabilidad de compra. Sin embargo, en la configuración 2, el pico se encuentra enviando al 40 por ciento. Finalmente, hay que destacar que estos datos son bastante sensibles al modelo, ya que si por ejemplo, las ventas finales fuesen del 30 por ciento en vez del 36 por ciento previsto por el modelo, los beneficios estimados para el envío al 30 por ciento de los clientes con mejores expectativas pasarían de 16.100 euros a 2.962 euros. No obstante, si el modelo es fiable esta sencilla técnica permite considerar campañas que a priori (para todos los clientes) no serían rentables, además de la ventaja de que no saturan de propaganda a los clientes.

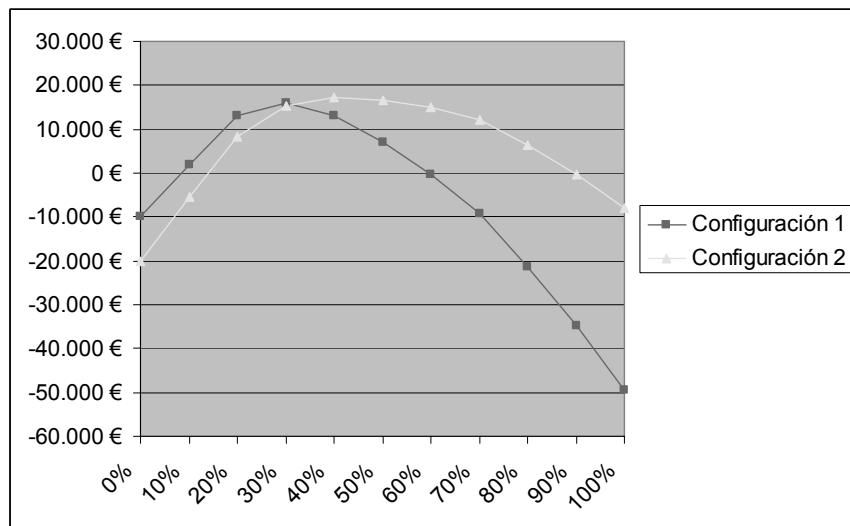


Figura 19.11. Gráfico de beneficios de la campaña.

Por último, es necesario remarcar que si no podemos estimar los costes asociados a la campaña, podemos utilizar el análisis ROC (véase la Sección 17.2.2) que dado un conjunto de clasificadores, permite seleccionar un subconjunto de clasificadores que tenga un comportamiento de coste óptimo para cualquier contexto de costes.

19.5.3 Simulación

Vamos a introducir brevemente el uso de técnicas de minería de datos para mejorar modelos de simulación de sistemas. La simulación es un tipo de modelización por el que se trata de representar la realidad de una manera simplificada. En muchos casos, las simulaciones se llevan a cabo para conocer los efectos que supondría la introducción de ciertos cambios en el sistema.

Para que una simulación obtenga resultados útiles debe representar con la máxima precisión posible el sistema a modelizar. Es en este aspecto donde las técnicas de minería

de datos pueden ayudar, dado que existen muchos sistemas cuyo comportamiento es tan complejo y conlleva tal cantidad de información, que sin la ayuda de técnicas de aprendizaje automático, sería prácticamente imposible representar su comportamiento de manera mínimamente fiable.

Las técnicas de minería de datos pueden también aplicarse a modelos de simulación ya existentes con el fin de refinar su comportamiento de manera que reflejen mejor la realidad. Por ejemplo, [Pyle 2003] cita el caso de una compañía de venta de materiales de los Estados Unidos; esta empresa había realizado un modelo de simulación de ventas utilizando el conocimiento y experiencia de sus empleados. Este modelo ayudó a mejorar el diez por ciento de las respuestas de la empresa a demandas de servicio por parte de los clientes de la empresa. Sin embargo, el modelo presentaba todavía algunas deficiencias, ya que en algunos casos no predecía correctamente los acontecimientos. Con el fin de aumentar las prestaciones del modelo de simulación, se utilizó minería de datos para descubrir pautas de comportamiento. Estas pautas fueron incorporadas al modelo, y gracias a ellas, se pudieron descubrir varios problemas en el modelo de simulación anterior. Esta corrección del modelo permitió que se mejoraran cerca del 30 por ciento de las demandas.

Cabe resaltar que probablemente en el ejemplo anterior no se habrían conseguido esas altas cotas de efectividad en la simulación sin la participación en su construcción de las dos partes: experiencia humana y técnicas de minería de datos. La clave del éxito para construir un modelo de simulación lo más ajustado a la realidad estriba en la correcta combinación de ambas fuentes de conocimiento.

19.6 Actualización y revisión de modelos

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

Por otra parte, en muchas ocasiones, las bases de datos desde donde extraer información con técnicas de minería de datos están formadas por un número tan alto de registros (filas), o bien de columnas (atributos), que hace inviable su tratamiento directamente por parte de los algoritmos de aprendizaje, ya que desborda la capacidad de los sistemas. Como hemos comentado anteriormente, una posible solución es reducir la dimensionalidad trabajando con una muestra de los ejemplos (véase la Sección 5.4.1), o bien limitando el número de atributos que se utilizan en el aprendizaje (véase la Sección 5.4.2).

Otra aproximación diferente es la utilización de algoritmos incrementales. Un algoritmo incremental es capaz de aprender modelos utilizando tan sólo una parte de la información disponible, y posteriormente, utilizar el resto para actualizar o revisar los modelos aprendidos. Si el algoritmo utiliza los ejemplos gradualmente, se denomina incrementalidad vertical. En el caso de que el algoritmo emplee todos los ejemplos, pero la incrementalidad se realice con respecto al número de atributos, se llama incrementalidad horizontal. Aunque existen dos tipos de incrementalidad, la primera suele ser la más extendida, y en muchos casos, el término incrementalidad se refiere sólo a la primera.

En verdad, un algoritmo incremental es una aproximación más realista que un algoritmo no incremental, ya que en muchos casos es imposible tener ejemplos de todos los casos

(o bien conocer todos sus atributos) en el momento de realizar aprendizaje. En esta situación, es mejor aprender un modelo inicial para posteriormente aplicarlo para predecir nuevos casos, y cuando se conozca la clase real de los nuevos casos (siempre que sea posible), utilizar esta información para realimentar el modelo. Esta realimentación puede suponer la actualización del modelo mediante la modificación o supresión de parte del modelo, o bien la inclusión de nuevas partes.

Debemos tener en cuenta que, en muchos problemas el comportamiento no es estático, sino que va evolucionando constantemente, por lo que utilizar un algoritmo de aprendizaje incremental permite que el modelo vaya evolucionando al mismo tiempo que el problema. Por ejemplo, podríamos aprender un modelo para la predicción del abandono de los alumnos de primer curso de una facultad. Este modelo podríamos aplicarlo durante varios años para estimar la matrícula del segundo curso. Obviamente, aunque el comportamiento de los alumnos tendrá algún patrón común durante estos años, existen otros factores que pueden afectar al comportamiento (cambio de plan de estudios en la enseñanza secundaria, incremento en la nota mínima de acceso a la facultad...), y que supondrían una pérdida de precisión del modelo, en caso de que no se efectúe una actualización del mismo.

Existen versiones incrementales para la mayoría de los algoritmos de aprendizaje. Cabe reseñar que, de manera lógica, se ha estudiado más esta extensión en algoritmos que no son particularmente escalables para el tratamiento de grandes volúmenes de datos, como por ejemplo los árboles de decisión (véase la Sección 11.7).

Finalmente, si por alguna limitación no tenemos a nuestra disposición un algoritmo incremental es posible realizar algún tipo de revisión/actualización del modelo de manera manual y poco eficiente. En este contexto, cuando, tras el transcurso de cierto tiempo, tenemos nuevos casos podemos comprobar si el modelo se comporta bien con estos nuevos casos. Podemos decidir realizar una revisión total cuando el número de nuevos casos problemáticos supere un umbral mínimo. Para ello, se realiza el aprendizaje partiendo de cero y utilizando los ejemplos iniciales junto con todos los nuevos casos conocidos posteriormente, incluidos los ejemplos problemáticos. En general, no habría de esperarse a que el modelo tenga mal comportamiento, ya que hay que intentar detectar esta situación con anterioridad. Al primer signo de que el modelo se comporta de manera errónea, hay que intentar analizar las causas, y cómo se pueden rectificar.

PARTE V

MINERÍA DE DATOS

COMPLEJOS

En esta parte se analiza la minería de datos estructurados pero heterogéneos (espaciales, temporales, secuenciales y multimedia), no estructurados (documentos de texto y de la web), así como los semiestructurados (XML). También se trata la minería de estructura web y la minería de uso web.

CAPÍTULOS

- 20. Minería de datos espaciales, temporales, secuenciales y multimedia**
- 21. Minería de web y textos**

Capítulo 20

MINERÍA DE DATOS

ESPACIALES, TEMPORALES, SECUENCIALES Y MULTIMEDIA

Con este capítulo iniciamos la parte del libro dedicada a la extracción del conocimiento desde repositorios de datos que contienen datos complejos y/o heterogéneos. En este capítulo abordamos la minería de datos desde bases de datos espaciales, temporales o con contenido multimedia. En el siguiente capítulo se trata la minería de datos desde textos, documentos en la web, y documentos XML. Dado el importante crecimiento que están teniendo estos tipos de bases de datos, están apareciendo multitud de técnicas de minería de datos que tratan específicamente estos tipos de problemas.

20.1 Introducción

Gran parte de las bases de datos contienen algún tipo de información que ha sido almacenada cronológicamente en períodos de tiempos constantes, o bien, presenta información que puede ser considerada como una secuencia de eventos. Un ejemplo de este tipo de repositorios es una base de datos con la evolución de la cotización de valores en bolsa, o bien una base de datos con información de las compras realizadas a lo largo de un período por un grupo de clientes.

Por otra parte, otro tipo de información compleja es aquella en la que el espacio juega un papel fundamental: una base de datos sobre información de enfermedades infecciosas de países del continente africano, o los datos del censo de todas las localidades de una determina región.

Finalmente, otro tipo de información que se está almacenando masivamente en los últimos años es la información multimedia. La principal peculiaridad de la información multimedia es que contiene simultáneamente varios medios de información: audio, vídeo, texto, imagen, etc.

Cada uno de estos tipos de información contiene una serie de características especiales, como por ejemplo relaciones intrínsecas (espaciales o temporales) entre los datos, o bien propiedades no detectables directamente. Estas características, en general, no pueden ser tratadas directamente por los algoritmos tradicionales de extracción de conocimiento desde bases de datos. Este hecho limita profundamente la calidad del conocimiento que se obtiene con la aplicación de las técnicas de minería de datos tradicionales a estos tipos de datos. Por ejemplo, consideremos el problema de predecir la cotización del valor en bolsa de una determinada empresa de venta de helados. Atributos como el estado financiero de la empresa, o la temperatura media influirán sin lugar a duda en el comportamiento en bolsa. Sin embargo, afectará con mayor importancia la evolución que ha tenido ese valor en el último período, las fluctuaciones, los ciclos, etc.

Motivados por estas limitaciones han surgido subáreas específicas del área de la minería de datos que tratan de desarrollar técnicas concretas para cada uno de estos tipos especiales de información. Cada subárea supone un campo muy activo y prolífico de investigación donde se están desarrollando trabajos que permiten aplicar con éxito nuevas técnicas de minería de datos en estos tipos de información. Podemos clasificar los trabajos en dos grandes grupos. El primer grupo se compone de las investigaciones sobre técnicas para transformar la información a un formato tabular de manera que salgan a la luz características “ocultas” de los datos, y de esta forma, puedan ser tratados directamente por algoritmos de aprendizaje no específicos. La otra familia la forman los trabajos que desarrollan técnicas específicas de minería de datos que permiten capturar las propiedades intrínsecas de los datos sin transformación alguna.

En este capítulo vamos a abordar parte de la minería de datos desde bases de datos con características especiales. Expondremos las problemática específicas de cada tipo, así como varias de las técnicas que se han propuesto para su solución. Hemos dividido el capítulo en cuatro secciones según el tipo de relación que tengan los datos entre sí. En el primer punto se tratan las bases de datos con información relacionada espacialmente, o bases de datos espaciales. La siguiente sección contiene un resumen sobre la minería de datos temporales, centrándonos en el análisis de series temporales. La siguiente sección está muy relacionada con la minería de datos temporales, ya que en este caso se trata de bases de datos que almacenan datos que están relacionados con el tiempo u otro tipo de secuencia. Finalmente, la última sección introduce la subárea de minería de datos sobre bases de datos con información multimedia.

20.2 Minería de datos espaciales

Una base de datos espacial es una base de datos que contiene datos pertenecientes a un determinado espacio. Ejemplos de bases de datos espaciales son: una base de datos sobre los inmuebles de una ciudad, las habitaciones de un hotel, los cultivos de un término, la distribución de las moléculas en una proteína, etc. Este tipo de bases de datos son muy empleadas en sectores como la arquitectura, la navegación, la agricultura, el medio ambiente, la química, etc.

Un concepto clave en las bases de datos espaciales es, lógicamente, la dimensión espacio. Esta dimensión establece el marco de referencia donde ubicar los datos. Ejemplos de espacios en dos dimensiones son: el mapa del término de una localidad, el mapa urbano

de una ciudad, un diseño de una placa VLSI (*very large scale integration*), etc. En cuanto a tres dimensiones: la galaxia, un modelo del cerebro, una proteína, etc.

Una base de datos espacial requiere que sea capaz de tratar con una gran cantidad de datos, así como de tecnología para tratar datos espaciales.

Un sistema de información espacial o geográfica SIG (en inglés GIS *Geographic Information Systems*) es un conjunto de métodos, herramientas y datos que permiten capturar, almacenar, analizar, transformar y presentar toda la información geográfica y de sus atributos almacenada en una base de datos espacial.

Asociados a los SIG, se han desarrollado métodos de visualización de la información contenida en las bases de datos espaciales de acuerdo con su distribución espacial. Por ejemplo, en [Andrienko & Andrienko 1999] se presenta la herramienta *Descartes*. Esta herramienta permite la generación de gráficos con información geográfica de manera sencilla y personalizada a partir de bases de datos espaciales. En la Figura 20.1 podemos ver un gráfico generado con la versión *applet* de esta herramienta disponible en la página web (<http://borneo.gmd.de/descartes/IcaVisApplet/index.html>).

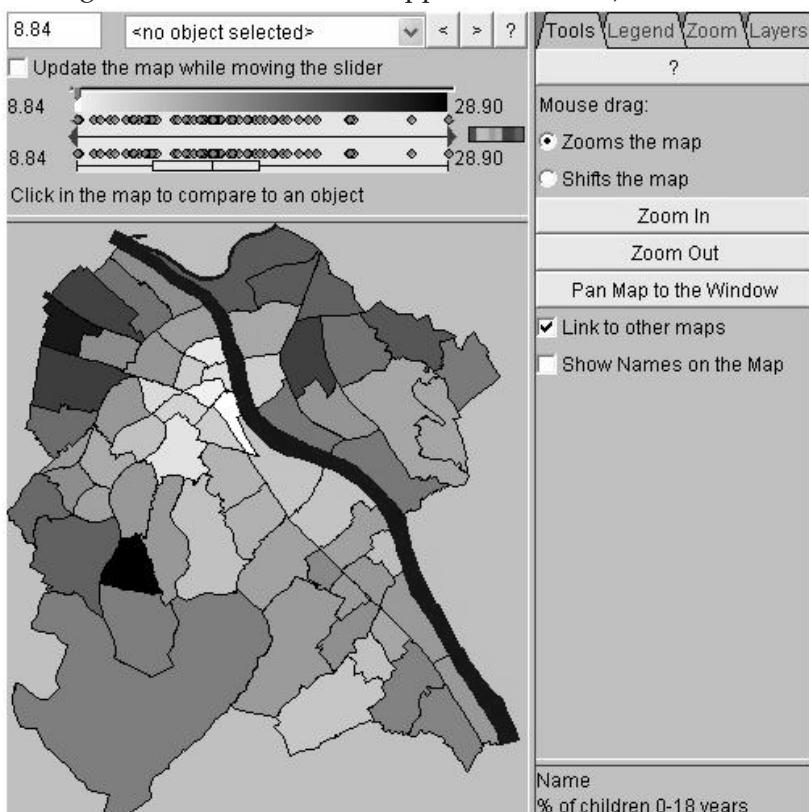


Figura 20.1. Distribución del porcentaje de menores por barrios de Berlín⁵⁹.

La Figura 20.1 incluye un mapa de Berlín dividido por distritos. La coloración de los distritos indica el porcentaje de jóvenes y niños (edad 0-18 años) de entre la población de ese distrito.

⁵⁹ © N. & G. Andrienko (<http://borneo.gmd.de/and/>), GMD SET.KI (<http://set.gmd.de/>).

Los SIG permiten tratar consultas del tipo: ¿Qué dos barrios limítrofes entre sí de la ciudad de Berlín tienen índices de población menor más diferentes? ¿Qué ciudad costera de Colombia tuvo mayores precipitaciones en 2003? ¿Cuántos kilos de chufa produjeron el año 2002 las parcelas que están al norte de la acequia de Moncada? ¿Cuál es el camino más corto para viajar desde Castellón a Cádiz? Es decir, permiten extraer información desde la base de datos espacial de maneras más flexibles, respecto al espacio, que un sistema tradicional.

El número de base de datos espaciales está creciendo constantemente, gracias al rápido desarrollo de las áreas asociados a este tipo de bases de datos: medicina, geología, química, astronomía, etc. Hay estimaciones de que cerca del 80 por ciento de la información tratada por instituciones y empresas públicas o privadas tienen en alguna medida relación con datos espaciales. El porcentaje de SIGs todavía está muy por debajo de ese porcentaje.

Dadas las características especiales de este tipo de información, se han desarrollado estructuras de datos que permiten un mejor almacenamiento y tratamiento. Una de las estructuras más conocidas y utilizadas para almacenar información con estructura espacial son los “árboles r” (*r-trees*, de *region trees*) [Guttman 1984]. Este tipo de árboles permiten que los datos sean almacenados en la base de datos con respecto a la posición que ocupan en el espacio real. En los árboles r las hojas están formadas por punteros a los datos, mientras que los nodos intermedios son los rectángulos mínimos que contienen todos sus sub-nodos y los punteros a los nodos hijos. Todas las hojas están al mismo nivel del árbol. La Figura 20.2 muestra un ejemplo de cómo se almacenan unos datos espaciales en un árbol-r. En este espacio hay un total de ocho objetos, los nodos *a*, *b*, *c* y *d* son nodos hoja. Por ejemplo el nodo *a* contiene el rectángulo mínimo que envuelve los objetos 1 y 2, y el nodo *B* contiene el rectángulo mínimo que envuelve a los nodos *c* y *d*.

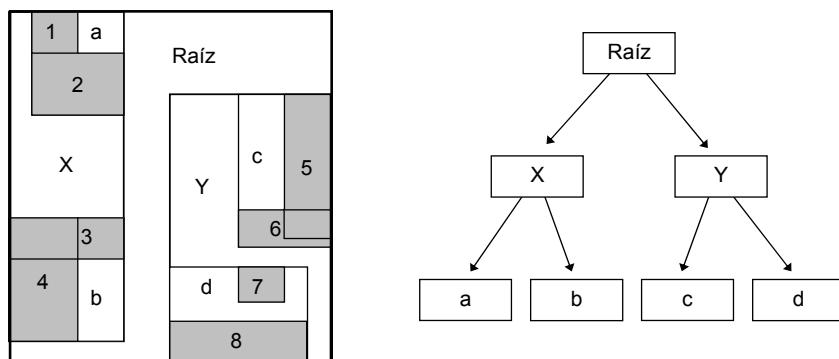


Figura 20.2. Ejemplo de un “árbol r” para dos dimensiones.

La ventaja de utilizar este tipo de estructura para almacenar información espacial es que permite la realización eficiente de operaciones tales como la intersección entre dos elementos. Esta operación es bastante común en bases de datos espaciales y tiene un coste elevado en otros tipos de estructuras de datos. Existen extensiones de estas estructuras tales como los árboles R+ [Sellis et al. 1987] que evitan regiones solapadas, y los árboles R* [Beckmann et al. 1990] que mejoran la eficiencia de las operaciones de inserción y desdoblamiento.

En cuanto a la extracción de conocimiento desde bases de datos con información de carácter espacial, el número de las técnicas de minería de datos desarrolladas específica-

mente para este tipo de información es bastante escaso. Los atributos de los datos cercanos al dato específico estudiado pueden ser en muchos casos de vital importancia. Por ejemplo, al nivel de ruido de una calle le afecta de manera importante que por la calle paralela pase una carretera, o bien que haya un aeropuerto a dos kilómetros. Existen varios métodos que tienen en cuenta estas características especiales de la información de una base de datos espaciales. A continuación describimos algunos de estos métodos separados por la tarea de minería de datos que desarrollan.

20.2.1 Clasificación y regresión de datos espaciales

Como hemos explicado en el punto anterior, una característica especial de los objetos emplazados en un sistema espacial es que, en muchas ocasiones, las propiedades de los objetos cercanos en distancia (vecinos) afectan al propio objeto. Esta particularidad es tomada en cuenta en el algoritmo de aprendizaje de árboles presentado en [Ester et al. 2000]. Este algoritmo se basa en método clásico de aprendizaje de árboles de decisión ID3 (véase la Sección 11.2), pero además de considerar los atributos del propio objeto a clasificar, los árboles de decisión también utilizan los atributos de los datos vecinos. Es necesario establecer un límite de hasta dónde buscar vecinos, para ello el usuario establece una distancia máxima. Un ejemplo de un árbol de decisión de este tipo para datos espaciales lo podemos ver en la Figura 20.3.

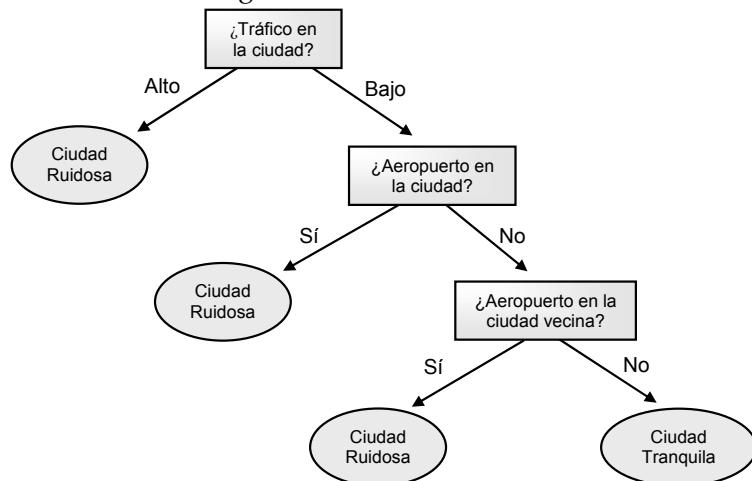


Figura 20.3. Ejemplo de un árbol de decisión para datos espaciales.

Otro trabajo para el aprendizaje de reglas de decisión desde base de datos espaciales es [Koperski et al. 1998]. En este caso se utiliza la información de una determinada zona para seleccionar los atributos que van a ser empleados en la construcción del árbol. Brevemente, el algoritmo establece un área de estudio, selecciona los objetos dentro de esa área y los separa por clase. A continuación ordena los atributos, favoreciendo aquellos atributos que tienen idéntico valor si coincide la clase, y restando importancia a aquellos atributos que varían su valor para una misma clase. Al final, selecciona para el aprendizaje sólo los atributos que superen cierto umbral.

20.2.2 Reglas de asociación de datos espaciales

Las reglas de asociación (véase el Capítulo 9) pueden también emplearse para expresar pautas de comportamiento en bases de datos espaciales. En este caso, una regla de asociación se representa como una regla de la forma **SI** α **ENTONCES** β , donde α y β son conjuntos de predicados espaciales o no espaciales. Un ejemplo de regla de asociación espacial sería:

Si es_un(X, "país") **Y** cerca (X, "Caribe") **ENTONCES** tiene(X, "turismo") Cob=0,1 Conf=0,75

Esta regla indica que si un país está cerca del mar Caribe (y lo cumplen el diez por ciento de los países analizados), se cumple al 75 por ciento que ese país recibe turistas. Una regla de asociación espacial considera relaciones topológicas entre objeto: *solapa*, *intersecta*, etc.; de orientación espacial: *a_la_izquierda*, *al_norte*, etc.; o bien de información sobre distancias: *cerca*, *lejos*, etc.

En [Koperski & Hand 1995] puede encontrarse un algoritmo de búsqueda de reglas de asociación espaciales. Este algoritmo se ejecuta en dos pasos: el primer paso busca de manera aproximada y eficiente patrones en toda la base de datos; el segundo paso se centra en refinar los patrones encontrados que parezcan más prometedores.

20.2.3 Agrupamiento de datos espaciales

Otra tarea de minería de datos que vamos a comentar es el agrupamiento. Esta tarea consiste en encontrar grupos de elementos de manera que los componentes de los grupos sean lo más parecidos entre ellos y lo más diferentes con respecto de los elementos externos al grupo. En realidad muchos algoritmos de agrupamiento se basan en distancias entre los componentes, como vimos en el Capítulo 16, por lo que pueden tener directamente en cuenta el componente espacial. Comentamos aquí algunos de estos métodos (algunos de ellos también comentados en el Capítulo 16).

Una primera aproximación fue propuesta en el algoritmo PAM (*Partitioning Around Medoids*) [Kaufman & Rousseeuw 1990]. Este método se basa en encontrar k grupos en un conjunto formado por n objetos. Para ello elige un elemento representativo o "medioide" por cada grupo, eligiendo el elemento que se halle más centrado en cada grupo. Este algoritmo es poco eficiente para valores altos de n y k .

Para paliar parcialmente este problema, los mismos autores diseñaron CLARA (*Clustering LARge Applications*). Se mejora la eficiencia del algoritmo trabajando con una muestra aleatoria de los datos, en vez de con todos los datos.

Finalmente, [Ng & Han 1994] han propuesto CLARANS (*Clustering LARge Applications based upon Randomized Search*) como una modificación del algoritmo CLARA. En este caso, en cada iteración se trabaja con una muestra diferente. En el mismo trabajo comprueban experimentalmente que su método es bastante más eficiente que CLARA y PAM.

El problema que en ocasiones surge cuando se quieren utilizar estos algoritmos con datos temporales, es que encontramos que los datos no sean geométricos directamente, sino que se dividan en regiones y no en puntos. En este caso existen relaciones de adyacencia. En esta situación es necesario realizar algún tipo de adaptación sobre los datos.

20.2.4 Detección de tendencias espaciales

Por último lugar, vamos a comentar una tarea propia de las bases de datos espaciales, la detección de tendencias espaciales. Tradicionalmente, el análisis de tendencias tiene que ver con el estudio de los cambios a lo largo del tiempo. Las tendencias espaciales reemplazan el espacio por el tiempo. [Ester et al. 1998] definen una tendencia espacial como un cambio regular de uno o más atributos no espaciales cuando nos desplazamos desde un determinado objeto de inicio. Por ejemplo, podríamos estudiar el nivel de empleo a partir de la distancia de un centro industrial, o bien la variación de clima a partir de la distancia al mar. Dos algoritmos de aprendizaje de tendencias espaciales se han definido en [Ester et al. 1998].

Para terminar este punto vamos a citar varios textos que pueden servir de referencia para conocer más técnicas desarrolladas para la minería de datos espaciales: [Koperski et al. 1997], el Capítulo 16.7 de [Klösgen & Zytkow 2002], el Capítulo 9.2 de [Han & Kamber 2001] y el libro [Miller & Han 2001].

20.3 Minería de datos temporales

Un tipo de datos que ha despertado interés especial en su investigación por la gran cantidad de aplicaciones que pueden derivarse son los datos que tienen componente temporal. Muchas bases de datos están formadas por series con observaciones de carácter cronológico que normalmente se realizan de forma repetida y con la misma frecuencia. Este tipo de series se denominan series temporales.

Existen multitud de ejemplos de series temporales. Por ejemplo, la evolución diaria de la cotización en bolsa de un determinado valor, el número de vehículos que han transitado por una determinada carretera medida hora a hora, la evolución diaria de una epidemia de gripe medida en el número de personas afectadas por día, etc. La previsión de cómo evolucionarán las series en el futuro tiene indudables aplicaciones. Para los casos comentados previamente, una correcta predicción nos puede ayudar a la toma de decisión sobre la compra o venta del determinado valor, o bien adaptar el funcionamiento de los semáforos que controlan la carretera de acuerdo con las predicciones, o para el tercer ejemplo, contratar más personal sanitario en un hospital por la predicción de un incremento masivo en la llegada de pacientes enfermos de gripe.

Existen dos grandes objetivos que han impulsado el estudio de las series temporales: identificar la naturaleza del sistema que genera la secuencia de los datos, y predecir los valores futuros que tomará la serie temporal.

Para conocer el comportamiento que tiene una serie temporal normalmente se descompone en cuatro elementos o movimientos principales [Han & Kamber 2001]:

- **Movimientos a largo término o tendencias:** estos movimientos indican el comportamiento general de la serie en un período largo de tiempo. Ayudan a identificar cuál es la tendencia que sigue o ha seguido la serie. Por ejemplo, podemos ver que en tiempos de bonanza económica existe una tendencia al alza en el número de ventas de coches.
- **Variaciones cíclicas:** representan ciclos que presentan las series. Estas variaciones cíclicas pueden o no ser periódicas. Es decir, los ciclos pueden no ser completamente

te iguales después de períodos de tiempos idénticos. Un ejemplo de estos ciclos son las oscilaciones que presentan las cotizaciones en bolsa de los valores de una compañía. Normalmente, cuando se conoce una buena noticia⁶⁰ se produce un inmediato incremento en el valor de las acciones, que con el tiempo, se convierte en una bajada por la recogida de beneficios.

- **Movimientos estacionales:** estos movimientos se deben a eventos que ocurren con una frecuencia establecida y constante. Por ejemplo, el número de juguetes vendidos se incrementa considerablemente en el período de navidad, o el número de ingresos en urgencias es siempre más alto las noches de los fines de semana que las noches entre semana.
- **Movimientos aleatorios o irregulares:** estos movimientos representan el comportamiento de la serie debido a eventos aleatorios o semi-aleatorios. Por ejemplo, tormentas fuertes, atentados terroristas, irrupción de un virus desconocido, etc.

El análisis de series temporales es conocido también como la descomposición de series temporales en estos cuatro movimientos básicos.

20.3.1 Análisis de tendencias

Como hemos visto, uno de los componentes en los cuales se descompone una serie temporal es la tendencia. Existen varias técnicas que permiten extraer la tendencia de una serie intentando minimizar el efecto del resto de componentes de la serie. Vamos a introducir algunas técnicas que permiten conocer estas tendencias.

Probablemente, la técnica más popular para identificar tendencias de series temporales es suavizar la serie temporal mediante el cómputo de la media de un intervalo. Dado una serie de valores y_1, y_2, \dots, y_n , para calcular la serie media sobre un intervalo de tamaño x se transforma cada valor de la siguiente forma (si x es impar):

$$y'_i = \frac{y_{(i-(x-1)/2)} + y_{(i-(x-1)/2+1)} + \dots + y_{(i-1)} + y + y_{(i+1)} + y_{(i+(x-1)/2-1)} + y_{(i+(x-1)/2)}}{x+1}$$

Si x es par, existen dos opciones para calcular y'_i , o tomar el intervalo con x valores empezando por $y_{(i-x/2)}$ y terminado en $y_{(i+x/2-1)}$, o bien, comenzando en $y_{(i-x/2+1)}$ y terminando en $y_{(i-x/2)}$. Para los valores de los extremos (principio o final de la serie), dado que no pueden acceder a los datos suficientes para calcular las medias, se suelen reducir los intervalos con el número máximo que permita calcular la serie suavizada.

En la Figura 20.4 podemos apreciar la evolución del índice IBEX-35 de la bolsa de Madrid entre los meses de agosto y octubre de 2003. Además se han incluido dos series suavizadas para apreciar la tendencia del índice. Una calculada usando un intervalo de nueve sesiones, y otra utilizando 19. Obviamente, cuanto mayor es el valor utilizado para calcular la serie suavizada, mayor es su capacidad de ignorar oscilaciones temporales. Por ejemplo, podemos ver la importante caída y posterior recuperación que sucedió a finales de

⁶⁰ Para ser más precisos, deberíamos referirnos a una noticia más favorable de lo esperado, ya que en ocasiones el anuncio de beneficios millonarios repercute en la caída del valor, por ser beneficios menores de lo esperado.

septiembre y principios de octubre; el efecto sobre la serie calculada usando nueve valores es bastante más significativo que sobre la serie con 19 valores.

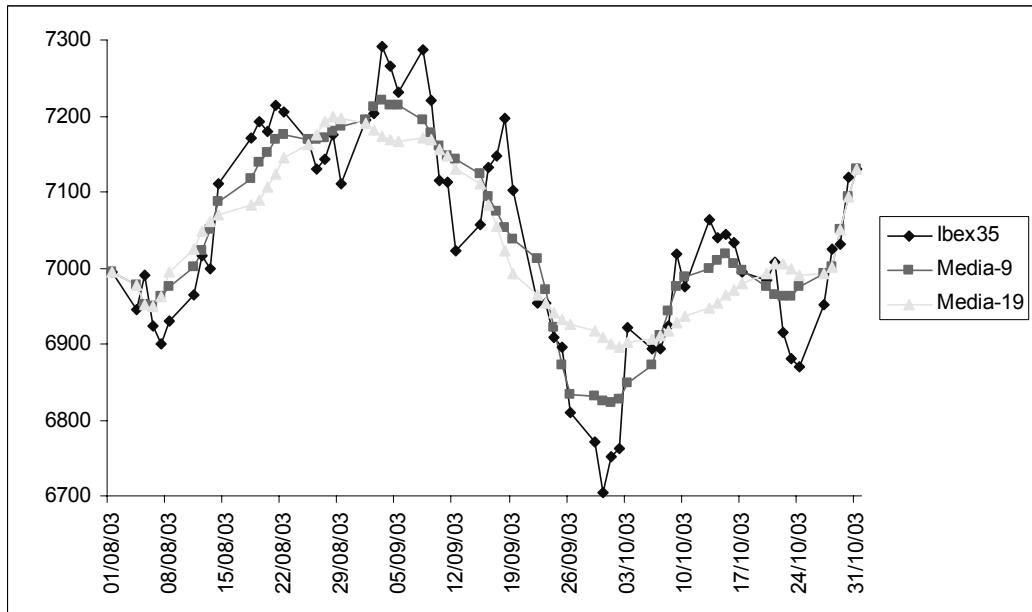


Figura 20.4. Evolución del índice IBEX-35 de agosto a octubre de 2003.

Existen otras opciones para realizar suavizados. Por ejemplo, podemos también variar el peso de los valores de las serie para calcular las medias, y por ejemplo, dar más valor a los valores más cercanos en tiempo que a los más lejanos.

Se pueden también utilizar otras aproximaciones para suavizar las series temporales y estudiar así las tendencias. Por ejemplo, utilizando alguno de estos suavizados es más sencillo realizar una regresión no lineal (entre las vistas en los capítulos 7 y 8) para aprender una función f , de manera que se minimice la distancia cuadrática total entre la función y la serie. Para calcular la distancia, empleamos la diferencia del valor de la serie y_i y el valor correspondiente de la función f .

Por otra parte, sería interesante el estudio de tendencias temporales y espaciales que se dan de manera simultánea. Hay muchos fenómenos que dependen claramente de la situación temporal y espacial simultáneamente. Por ejemplo, el nivel de agua de un río, o bien el comportamiento meteorológico. Aunque, por separado, sí se han desarrollado técnicas de minería de datos, hasta el momento, hay pocos trabajos que contemplen ambas dimensiones simultáneamente.

20.3.2 Análisis de movimientos estacionales

Un movimiento estacional se define formalmente como una dependencia correlacional de orden k entre cada uno de los i -ésimos elementos de la serie y el elemento k -ésimo posterior [Kendall 1975]. Esta dependencia normalmente se mide mediante autocorrelación (correlación entre dos valores desplazados de las mismas variables). El valor k se denomina usualmente como *lag* (retraso). Un movimiento estacional se aprecia visualmente como un patrón que se repite cada k elementos.

Como hemos comentado, el coeficiente de autocorrelación (o autocorrelación simple) permite estudiar la existencia de patrones estacionales en la serie. Si queremos analizar la relación entre elementos adyacentes, calcularemos la autocorrelación tomando $k=1$. Tomando $k=2$, medimos la relación entre los elementos separados por un valor entre ellos, tomando $k=3$ sería la correlación entre cada componente y el situado tres posiciones más atrás, y así sucesivamente.

Formalmente, dado un conjunto de n valores y_1, y_2, \dots, y_n tomados en los instantes x_1, x_2, \dots, x_n , la autocorrelación para un retraso de k se define como:

$$r_k = \frac{\sum_{i=1}^{n-k} (y_i - \bar{y})(y_{i+k} - \bar{y})}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

donde

$$\bar{y} = \frac{\sum_{i=1}^n (x_i)}{n}, \text{ y } \bar{x} = \frac{\sum_{i=1}^n (x_i)}{k}$$

Un diagrama muy útil para identificar los patrones estacionales en las series son los correlogramas de autocorrelación. Para dibujar un correlograma de autocorrelación se debe especificar un valor de k máximo (k_{max}), y calcular la autocorrelación para cada valor entre 1 y k_{max} , dibujando un diagrama con estos valores. En la Figura 20.5 se puede ver un correlograma de autocorrelación.

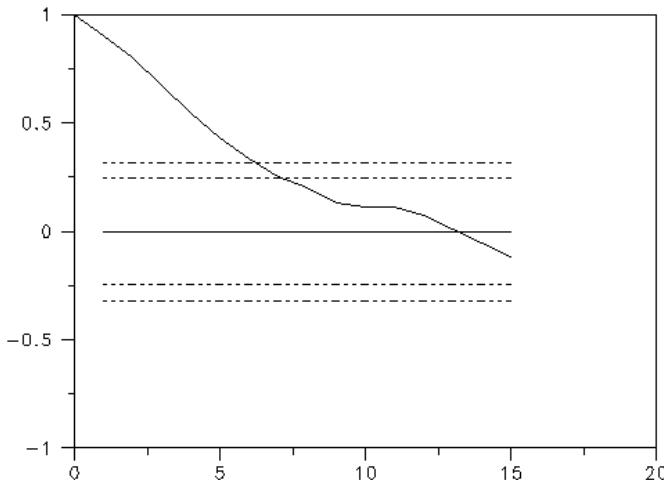


Figura 20.5. Correlograma de autocorrelación del índice IBEX-35 de agosto a octubre de 2003.

Hemos de tener en cuenta al observar un correlograma de autocorrelación que existen dependencias entre autocorrelaciones en valores continuos de k . Por ejemplo, si detectamos dependencia entre el primer y el segundo elemento, y entre el segundo y el tercero, existirá, por tanto, algún tipo de relación entre el primero y el tercero.

Esta reflexión nos lleva a una medida similar, es la autocorrelación parcial. En este caso, cuando calculamos este coeficiente para un k determinado (por ejemplo 4), se calcula de tal

manera que se eliminan los efectos de las autocorrelaciones de $k = 1, 2$ y 3 . De la misma manera que a partir del coeficiente de autocorrelación podemos dibujar un correlograma de autocorrelación, de la correlación parcial podemos generar una correlograma de autocorrelación parcial. Se pueden consultar algoritmos para calcular autocorrelaciones parciales en [Box et al. 1997].

A partir de estos gráficos podemos conocer propiedades interesantes de una serie temporal. Por ejemplo, si todos los valores de autocorrelación están significativamente cercanos a 0 , la serie temporal se puede considerar aleatoria. Si detectamos en el correlograma de autocorrelación algún pico que se repite cada k valores (por ejemplo cada siete valores para patrones semanales), indica que la serie presenta un patrón estacional de valor k .

En la Figura 20.5 podemos ver el correlograma de autocorrelación de la serie temporal que representa el índice IBEX-35 entre los meses de agosto y octubre de 2003. En el eje de las x se representa el *lag*, y el eje de las y se representa la autocorrelación (Figura 20.5). El diagrama contiene unas líneas discontinuas paralelas al eje de las x . Estas líneas indican las bandas de confianza del 95 por ciento y del 99 por ciento de la existencia de relación. El gráfico ha sido generado con la herramienta *dataplot*⁶¹. La gráfica indica que existe dependencia entre valores cercanos (*lag* inferior a 5).

Una vez hemos descubierto algún tipo de movimiento estacional con período k en la serie temporal podemos transformar la serie para el eliminar el efecto del movimiento estacional. Esta transformación se realiza por varios motivos. Eliminar el efecto de un movimiento estacional permite que se puedan descubrir otro tipo de movimientos estacionales secundarios ocultos inicialmente. Otro motivo es que eliminar el efecto de un movimiento estacional transforma la serie temporal en una serie *estacionaria*⁶², lo que permite que se puedan aplicar herramientas de análisis de series temporales como ARIMA.

ARIMA [Box et al. 1997] es una metodología que permite analizar series de datos temporales para realizar predicciones sobre el comportamiento de estas series. ARIMA es una técnica bastante compleja que requiere poseer experiencia en su utilización para obtener resultados fiables. Esta metodología se basa en la aplicación de dos procesos denominados proceso autoregresivo y proceso de desplazamiento de medias.

20.3.3 Búsqueda de similitudes en series temporales

Otra técnica utilizada en series temporales para predecir el comportamiento de series, es buscar secuencias similares a la presente ocurridas en el pasado, para observar cuál fue la evolución de la serie y extrapolar esa evolución al presente.

Frecuentemente se emplea la distancia euclídea para medir la similitud entre dos secuencias de series temporales desplazadas un retraso k . Sin embargo, suele ser más habitual aplicar transformaciones sobre la serie de datos ya que muchas técnicas de análisis

⁶¹ Herramienta estadística de visualización y modelización multi-plataforma de dominio público disponible (<http://www.itl.nist.gov/div898/software/dataplot/index.htm>).

⁶² En análisis de series temporales, una serie estacionaria se define como una serie que tiene media, varianza y autocorrelación constante a lo largo del tiempo. Para conseguir que una serie sea estacionaria se deben eliminar los movimientos estacionales.

de señales requieren que los datos se encuentren en el dominio de las frecuencias. La distancia entre dos señales en el dominio del tiempo es similar a la distancia en el dominio de la frecuencia. Dos transformaciones de este tipo son la transformación discreta de Fourier (*Discrete Fourier Transform*, DFT) y la DWT (*Discrete Wavelet Transform*). La transformación DFT captura de manera adecuada la señal en los primeros coeficientes, por lo que podemos utilizar estos coeficientes para calcular de manera aproximada la distancia.

Una vez transformada la serie completa al dominio de las frecuencias mediante la transformación DFT se pueden almacenar los primeros coeficientes de la transformada. De esta manera, dada una nueva secuencia de una serie si deseamos buscar una secuencia similar en la serie almacenada, aplicamos la transformación sobre la nueva secuencia y entonces buscamos la secuencia almacenada que más se le parezca de acuerdo con los coeficientes. Finalmente, una vez encontrada la secuencia candidata se puede volver al dominio del tiempo y calcular la distancia real para comprobar que la selección ha sido correcta.

Para concluir esta sección vamos a referir varios textos donde ampliar los conocimientos sobre análisis temporales. Existe un capítulo bastante completo y además accesible desde su página web en [Statsoft 2004]. Un libro específico de análisis de series temporales es [Box et al. 1997]. Finalmente, el Capítulo 9.4 de [Han & Kamber 2001] contiene una introducción bastante adecuada del área.

20.4 Extracción de patrones secuenciales

La minería de datos secuenciales se define como la extracción de patrones frecuentes relacionados con el tiempo u otro tipo de secuencia. A diferencia de las series temporales, el momento preciso no es tan relevante, sino que estos eventos se producen secuencialmente. Además, el objetivo no suele ser cómo seguirá una serie sino analizar muchos individuos que tienen un comportamiento secuencial. Un ejemplo de patrón secuencial podría ser “un cliente que se compra una bicicleta, probablemente se comprará un bombín en menos de tres meses”. Gran cantidad de datos pueden considerarse como datos secuenciales: transacciones comerciales, acceso a páginas web por un cliente, recorrido de un cliente por las secciones de un supermercado, etc. En esta sección analizamos algunas de las técnicas de minería de datos existentes para datos secuenciales.

20.4.1 Clasificación con datos secuenciales

En un contexto de datos secuenciales, los datos de entrenamiento representan una secuencia de ejemplos (x, y) donde datos contiguos presentan algún tipo de relación. Un ejemplo de datos secuenciales es el reconocimiento de caracteres escritos. En este caso, cada ejemplo lo compone una letra que el modelo debe reconocer. Las letras forman una secuencia que representa las palabras en un texto. Se debe tener en cuenta que se cumplen una serie de reglas que pueden ser muy útiles para el correcto reconocimiento de un carácter. Por ejemplo, detrás de dos consonantes en castellano la mayoría de las veces encontramos vocales.

Otro ejemplo de datos secuenciales podría ser la predicción del resultado de un encuentro de fútbol entre dos equipos. Podemos utilizar atributos tales como la clasificación de los dos equipos, el campo donde se realiza el encuentro, etc. Sin embargo,

un factor muy importante para predecir con fiabilidad el resultado es el estado de forma, y este elemento lo podemos deducir del resultado de ambos equipos en los últimos encuentros.

El problema clasificación secuencial puede ser formulado de la siguiente manera [Dietterich 2002]. Sea $\{(x_i, y_i)\}$, $1 \leq i \leq N$ un conjunto de N ejemplos. Cada ejemplo es un par de secuencias (x_i, y_i) , donde $x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, \dots, x_{i,T_i} \rangle$, y $y_i = \langle y_{i,1}, y_{i,2}, y_{i,3}, \dots, y_{i,T_i} \rangle$. Por ejemplo, el caso del reconocimiento de la estructura de frases de acuerdo con el tipo de las palabras. En castellano es bastante extraño encontrar una frase con la siguiente estructura (adjetivo adjetivo verbo verbo). En este caso del reconocimiento de la estructura de una frase, podríamos tener $x_i = \langle \text{Juan come naranjas dulces} \rangle$, y $y_i = \langle \text{nombre verbo nombre adjetivo} \rangle$. El objetivo del aprendizaje supervisado secuencial es la inducción de un modelo h que pueda predecir una secuencia "clase" $y = h(x)$ dada una secuencia de entrada x . Es decir, la clase no es un atributo sino una secuencia de atributos.

Debemos relacionar el aprendizaje supervisado secuencial con dos técnicas similares: el análisis de series temporales y la clasificación de secuencias. La primera técnica (vista en la sección anterior), consiste en dada una serie $\langle y_1, y_2, y_3, \dots, y_T \rangle$ se debe predecir el elemento y_{T+1} . Por otra parte, en la clasificación de secuencias el objetivo es aprender una clase y (ahora un solo atributo) desde toda una secuencia $\langle x_1, x_2, x_3, \dots, x_T \rangle$. Un ejemplo de clasificación de secuencias es el reconocimiento como correo *spam* de un correo electrónico. En este caso la secuencia sería la secuencia de caracteres escritos y la clase un valor binario que indica si el correo electrónico puede considerarse *spam*. Debemos tener en cuenta que estas tres técnicas están muy relacionadas por lo que, a menudo, un determinado problema puede ser abordado por varias técnicas al mismo tiempo.

Examinemos ahora algunas técnicas utilizadas para el aprendizaje supervisado secuencial. Una primera técnica que se ha utilizado es la denominada de ventanas deslizantes. Este método permite convertir un problema de aprendizaje supervisado secuencial en un problema clásico de aprendizaje supervisado. Para ello, se utiliza un clasificador ventana h_v que asigna a una ventana de entrada una salida individual y_i . Un posible ejemplo de ventana de tamaño w sería $\langle x_{i,t-d}, x_{i,t-d+1}, \dots, x_{i,t}, \dots, x_{i,t+d-1}, x_{i,t+d} \rangle$ donde $d = ((w-1)/2)$. A partir de esa ventana h_v predeciría el valor y_i . El clasificador ventana se entrena convirtiendo cada ejemplo secuencial (x_i, y_i) en ventanas para después poder aplicar un algoritmo de aprendizaje supervisado. Finalmente, para clasificar una secuencia nueva x' , se divide esta secuencia en ventanas, se aplica el clasificador ventana produciendo un valor y_i' por cada ventana, y por último se concatenan estos valores para formar la secuencia de salida y' . Un algoritmo de este tipo se ha utilizado, entre otras aplicaciones, para detectar fraudes en el uso de teléfonos móviles [Fawcett & Provost 1997]. Otra técnica similar se basa en utilizar ventanas recurrentes [Bakiri & Dietterich 2002]. En una ventana recurrente, el valor predicho en $y_{i,t}$ se utiliza también para realizar la predicción de $y_{i,t+1}$. El concepto de n-gramas no es más que la particularización del concepto de ventana a documentos de texto. Este concepto lo veremos en el capítulo siguiente.

Otras aproximaciones se basan en la utilización de modelos ocultos de Markov (HMM, *Hidden Markov Models*) [Rabiner & Juang 1986]. Estos modelos representan modelos probabilísticos de cómo se generan las secuencias x_i e y_i . En [Diligenti et al. 2003] podemos encontrar ejemplos de aplicación de técnicas basadas en HMM que obtienen buenos

resultados en problemas de aprendizaje supervisado secuencial y problemas de clasificación de secuencias.

Finalmente, indicar que un buen resumen de esta área puede consultarse en [Dietterich 2002].

20.4.2 Agrupamiento de patrones secuenciales

La tarea de agrupamiento se define como la tarea de separar en grupos a los datos, de manera que los miembros de un mismo grupo sean muy similares entre sí, y al mismo tiempo, sean diferentes a los objetos de otros grupos. Para el caso de datos secuenciales, la tarea de agrupamiento se convierte en la búsqueda de grupos de secuencias con alta cohesión. Existen muchos datos con componentes secuenciales: secuencias de proteínas, registros sobre los accesos a páginas web, transacciones comerciales, etc. Utilizar técnicas de agrupamiento para identificar grupos significativos puede conllevar aplicaciones importantes. Por ejemplo, agrupar secuencias de transacciones comerciales puede ayudar a identificar diferentes grupos de clientes de acuerdo a sus compras. O bien, encontrar grupos con secuencias de proteínas similares puede ayudar a identificar secuencias de idéntica funcionalidad.

Uno de los primeros pasos a la hora de diseñar un algoritmo de agrupamiento de secuencias es establecer una medida de similitud entre secuencias [Gunsfield 1997]. Una aproximación consiste en encontrar el alineamiento óptimo entre dos secuencias, y después comparar una a una las posiciones de la secuencia.

Tras establecer métodos para medir la similitud entre secuencias es posible utilizar algoritmos de agrupamiento tradicionales para separar los grupos. Sin embargo, existe un problema de eficiencia, ya que la estimación de la similitud entre secuencias es bastante más costosa que medir LA similitud entre datos. Este problema restringe en la práctica la utilización de estos métodos de agrupamiento a problemas de tamaño limitado.

Teniendo en cuenta esta carencia, en [Guralnik & Karypis 2001] se propone un algoritmo escalable de agrupamiento de datos secuenciales. Para ello, se selecciona un conjunto de componentes que capturan las características principales de varias secuencias de objetos. Es decir, se introduce una proyección del espacio de las secuencias a un nuevo espacio cuyas dimensiones son los componentes capturados. La ventaja de esta transformación es que se pueden aplicar los algoritmos de identificación de grupos de una manera bastante eficiente.

20.4.3 Reglas de asociación con datos secuenciales

Existe gran interés desde los últimos años en el desarrollo de algoritmos de aprendizaje de reglas de asociación desde datos secuenciales. La motivación de este interés es la gran cantidad de aplicaciones prácticas que tienen estas técnicas. Por ejemplo, podemos detectar patrones del tipo “si un cliente compra un reproductor de DVD, es probable que el mes siguiente compre varias películas de DVD”, o bien, “el 40 por cien de las personas que consultan la página web de información sobre la cartelera, visitan en menos de dos días la página web de compras de entradas de cine”. El tema de las reglas de asociación con datos secuenciales lo hemos tratado con detenimiento en el Capítulo 9 de reglas de asociación, concretamente en la Sección 9.5.

20.5 Minería de datos multimedia

Uno de los tipos de información cuyo almacenamiento masivo ha crecido recientemente, gracias a la proliferación de aparatos electrónicos de audio y vídeo, entre otras causas, es la información multimedia. La información multimedia contiene simultáneamente varios medios de información: audio, vídeo, texto, imagen, etc. Las técnicas básicas de minería de datos trabajan sobre información almacenada en forma de datos con formato simple tales como texto o números, existiendo pocos métodos específicos para información multimedia.

La propia naturaleza de la información multimedia, que contiene la información de formas diversas, complica de manera considerable su análisis. Sin embargo, dada la importancia creciente que está tomando este tipo de contenidos, están apareciendo nuevas técnicas que posibilitan la aplicabilidad de la minería de datos en este campo.

Principalmente, se han distinguido dos tipos de problemas fundamentales para el tratamiento de datos multimedia. Primeramente, la carencia de técnicas efectivas que permitan extraer información desde secuencias de vídeo o audio que pueda ser directamente analizada. Este hecho limita en gran medida el descubrimiento de patrones válidos. Por ejemplo, podríamos desear aprender un modelo de clasificación de fotos deportivas insertadas en una página web de información deportiva según su deporte. Si el sistema de análisis de la imagen no es capaz de distinguir entre un jugador de fútbol y un jugador de baloncesto, es difícil que obtengamos un modelo con buena precisión. Por otra parte, otro problema inherente a los datos multimedia es que la información viene por varios medios, por lo que es necesario establecer mecanismos que permitan unir la información de manera coherente. Por ejemplo, si queremos aprender un modelo de clasificación de noticias de un programa de televisión según su tipo, hemos de tener en cuenta (al menos) el audio y el vídeo, ya que, por ejemplo, pueden aparecer imágenes de un avión en la sección de internacional (accidente aéreo), en economía (fusión de dos compañías de líneas aéreas) o incluso en deportes (llegada del Valencia C.F. al aeropuerto de Buenos Aires en su gira americana).

A pesar de todos estos inconvenientes, varias aplicaciones están demostrando que vale la pena intentar la aplicación de minería de datos en bases de datos multimedia. Veamos a continuación algunos ejemplos de aplicaciones con éxito.

La prestigiosa universidad de Carnegie Mellon está desarrollando desde 1994 la iniciativa *informedia* (<http://www.informedia.cs.cmu.edu/>). En esta iniciativa caben proyectos relacionados con la comprensión automática de información con formato vídeo, audio, etc., para la realización de búsquedas de información, categorización, emisión de resúmenes, etc. Podemos citar varios proyectos, como por ejemplo la creación de una biblioteca digital que almacene documentos multimedia clasificados según su contenido. La idea es investigar métodos que permitan realizar esta clasificación de forma automática, para su correcta ubicación en la biblioteca. Además, se han investigado métodos de búsqueda inteligente que permitan encontrar documentos multimedia indagando en su contenido de acuerdo a las peticiones del cliente.

Otro proyecto interesante es el denominado *Aquaint*. Para situar el proyecto consideremos el siguiente dato: según la CIA (*World Factbook 2000*) existen 33.071 emisoras de televisión en el mundo. Emitiendo 16 horas al día, estas estaciones transmiten un total de 193 millones de horas de programación televisiva al año. Y esto sólo sin considerar otros

flujos de información como radio o comunicación por teléfono. Indudablemente, estos flujos de datos representan una fuente de información inmensa cuya utilización de manera automática tendría importantes aplicaciones (comerciales, militares, etc.). El propósito del proyecto *Aquaint* es el desarrollo de herramientas que permitan la captación automática de información que permita a su vez ayudar en la resolución de problemas⁶³.

Un proyecto también similar es MoCA (*Automatic Movie Content Analysis*) (<http://www.informatik.uni-mannheim.de/informatik/pi4/projects/MoCA/>). El objetivo de este proyecto es la extracción de información desde películas cinematográficas. Aunque este es el objetivo primordial, se han desarrollado subproyectos que permiten la extracción de objetos en movimiento desde secuencias de vídeo, la detección, segmentación, y reconocimiento de texto en secuencias de vídeo, etc. Uno de los subproyectos que tienen más relación con la minería de datos es el del reconocimiento automático del género de una película [Fisher et al. 1995].

Otra de las áreas de aplicación de la minería de datos en datos multimedia es el control de seguridad física. Consideremos el caso de la vigilancia de un complejo mediante cámara de seguridad. Dada la gran cantidad de información que es generada mediante la utilización de cámaras de seguridad, en muchas ocasiones es bastante costoso mantener un alto número de personas encargadas en observar las pantallas. Además, dada la monotonía de la tarea, los vigilantes tienden a relajar la atención sobre los monitores, incrementando el riesgo de una intrusión no detectada. En este contexto, se están investigando técnicas basadas en reconocimientos de patrones y minería de datos, que permitiesen la detección automática de intrusos a partir de las imágenes.

Finalmente, otra aplicación de gran relevancia es la minería de datos desde datos multimedia en entornos médicos. La detección automática de tumores u otras malformaciones a partir de imágenes como radiografías es un campo de investigación activo. Otra aplicación en esta área es el descubrimiento de relaciones entre áreas del cerebro con actividades del cuerpo humano utilizando imágenes fMRI (*Functional Magnetic Resonance Imaging*) del cerebro [Tsukimoto et al. 2002].

Una parte fundamental en la minería de datos con formato multimedia es el preprocesamiento de los datos. En esta parte y en este contexto, el objetivo es procesar los datos para intentar capturar toda la información posible en un formato que pueda ser utilizado de manera directa por las técnicas de minería de datos no especializadas en datos multimedia. Por ejemplo, podemos tratar imágenes de una entrada a un centro comercial para intentar conocer si hay personas en la entrada, o bien el número exacto de personas. Para este propósito se emplean técnicas de reconocimiento de formas (*pattern recognition*). El reconocimiento de formas [Duda et al. 2001] es un área muy cercana al aprendizaje automático definida como el área de investigación que estudia el diseño de sistemas que reconocen patrones desde datos. Para ello utiliza métodos de estadística, aprendizaje automático y otras áreas. Aplicaciones típicas de reconocimiento de formas son reconocimiento automático del habla, categorización de textos, reconocimiento de caracteres en texto escrito (OCR, *optical character recognition*), reconocimiento de caras

⁶³ Probablemente se halle detrás de este proyecto la enigmática red de captación de información *Echelon*.

humanas desde fotografías, etc. Los dos últimos ejemplos son del campo del análisis de imágenes, campo especializado en el tratamiento de imágenes para recuperar información.

A continuación resumimos brevemente algunos métodos de minería de datos empleados en las aplicaciones detalladas previamente y otras aplicaciones con datos multimedia. Para ello sepáramos los métodos según la tarea que realizan.

20.5.1 Descubrimiento de similitud de datos multimedia

Conocer si dos archivos son similares de acuerdo a su contenido tiene grandes aplicaciones prácticas. Por ejemplo, podemos utilizar la última imagen del satélite Meteosat⁶⁴ y buscar en la base de datos de períodos anteriores la imagen más similar para estudiar cómo evolucionó en aquella ocasión el tiempo, y utilizar esa información para la predicción actual. O bien, en una empresa que vende música a través de páginas web, si un usuario busca información sobre un determinado grupo, podemos sugerirle otros grupos que componen música que suena de manera similar.

Establecer la similitud entre dos archivos multimedia es bastante dependiente del medio que estemos comparando. Incluso en un mismo medio pueden variar los valores a considerar según el contenido. Por ejemplo, en archivos de audio con música se puede utilizar el ritmo, tono, etc. Mientras que en archivos de audio con voz se suele extraer la información a texto mediante herramientas de reconocimiento del habla, y a partir del texto podemos utilizar técnicas de minería de datos desde documentos de texto como las que veremos en el capítulo siguiente.

Para comparar imágenes existen varias aproximaciones [Han & Kamber 2001] basadas en la compresión de la imagen en un vector de características. Este vector se genera de acuerdo con unos criterios, y son utilizados directamente para comparar las imágenes. Algunos criterios para generar vectores de características son:

- Histogramas de color: estas aproximaciones utilizan únicamente la composición de colores de la imagen. Por lo tanto, ignoran cualquier otro tipo de información. Son eficientes pero demasiado simples.
- Múltiples características: en este caso se combina información de diferentes fuentes: color, formas, textura, etc. Puede incluso emplearse una distancia por cada fuente y calcular la distancia final mediante la fusión de las distancias parciales.
- Utilización de transformaciones DWT: las transformaciones DWT (*Discrete Wavelet Transform*) permiten comprimir en un solo vector la información de la imagen. Es más efectivo que el criterio anterior, ya que aunque utiliza la información de diversas fuentes, sólo genera un vector de características. Sin embargo, los resultados suelen ser peores.

20.5.2 Aprendizaje supervisado desde datos multimedia

Gran parte de las aplicaciones de la minería de datos multimedia utilizan técnicas del aprendizaje supervisado, principalmente la clasificación. Por ejemplo, se puede aprender un modelo que permita detectar si hay algún tumor en el cerebro utilizando imágenes del

⁶⁴ Satélite utilizado en Europa para la predicción meteorológica.

cerebro a partir de imágenes previamente etiquetadas. Otro ejemplo de aplicación es la clasificación automática de objetos estelares (estrellas, galaxias, etc.) empleando imágenes tomadas de los objetos [Fayyad 1995], o bien detectar automáticamente volcanes en la superficie de Venus [Burl et al. 1997].

Probablemente, en este contexto, debido a la complejidad inherente de los datos multimedia, es casi más importante la preparación de los datos que la propia fase de descubrimiento de conocimiento. La fase de preparación de los datos debe analizar los objetos multimedia e intentar extraer la máxima información posible de manera que pueda ser tratada por las técnicas del aprendizaje supervisado directamente. Es importante utilizar técnicas de reconocimiento de formas para intentar capturar el número máximo de componentes de manera adecuada. Por ejemplo, para trabajar con archivos de video de un informativo podemos utilizar técnicas de reconocimiento de formas en imágenes para conocer qué objetos aparecen en pantalla, pero, por otra parte, podemos utilizar técnicas de reconocimiento de voz para convertir el discurso del presentador en texto, que siempre se puede tratar más fácilmente. A lo largo del libro hemos utilizado otros ejemplos relacionados con datos multimedia. Por ejemplo, el conjunto de datos *Wisconsin Diagnostic Breast Cancer* (véase el Apéndice B) proviene de la conversión a atributos desde imágenes de mamas afectadas o no con cáncer. Otro ejemplo es la base de datos *isolet (Isolated Letter Speech Recognition)* (véase el Apéndice B). Esta base de datos está compuesta por la pronunciación de letras por 150 personas diferentes. Cada ejemplo contiene parámetros sobre la letra pronunciada como: características sonoras, coeficientes espectrales, etc.

20.5.3 Reglas de asociación en datos multimedia

Los trabajos de investigación en el aprendizaje de reglas de asociación que se han centrado en datos multimedia, lo han hecho sobre todo en el aprendizaje de reglas desde imágenes. Existen dos aproximaciones principales: reglas de asociación basadas en el contenido de la imagen y reglas de asociación basadas en la descripción de la imagen.

En el segundo grupo podemos ubicar el sistema *Multimedia Miner* [Zaïane et al. 1998]. En este prototipo, entre otras herramientas, se incluye un módulo denominado *MM-Associator* que permite aprender reglas de asociación desde datos multimedia. Para ello utiliza características relacionadas a las imágenes: tamaño, color, descripción, etc., no teniendo en cuenta los objetos que incluyen las imágenes. Un ejemplo de este tipo de regla sería: “si en una foto predomina el color azul probablemente la imagen mostrará el mar”.

En cuanto a la segunda aproximación podemos destacar el trabajo [Ordonez & Omiecienski 1999]. En este trabajo se describe un algoritmo que es capaz de describir reglas de asociación entre imágenes utilizando el contenido de las mismas. El algoritmo se divide en cuatro partes. La primera parte se encarga de dividir una imagen en segmentos, donde cada segmento probablemente contendrá un objeto; la segunda parte trata de identificar objetos, comparándolos entre ellos para conocer si dos segmentos representan el mismo objeto; la tercera parte genera imágenes auxiliares con los objetos identificados que ayuden a interpretar las reglas aprendidas en la siguiente fase; la cuarta y última fase aplica un algoritmo de aprendizaje de reglas de asociación para aprender reglas con los objetos identificados. Con este algoritmo podemos aprender reglas del tipo “si en la imagen hay un triángulo, probablemente también encontraremos un cuadrado”.

Para finalizar esta sección citamos alguna bibliografía de referencia sobre la minería de datos multimedia: el Capítulo 39 de [Klösgen & Zytkow 2002] y el Capítulo 9.3 de [Han & Kamber 2001]. Además, cabe destacar que desde 1999 se celebra un *workshop* anual sobre minería de datos multimedia (*International Workshop on Multimedia Data Mining*). Cada año se publica un artículo resumen de los temas tratados en la edición correspondiente. El último disponible es el de la edición de 2002 [Simoff et al. 2003].

Capítulo 21

MINERÍA DE WEB Y TEXTOS

En este capítulo describimos la minería web, es decir, el problema de extraer información a partir de documentos de la web. Las técnicas de minería web difieren significativamente de las técnicas vistas hasta ahora en el libro ya que la web es un repositorio de gran tamaño donde los documentos contienen datos de muy diverso tipo (texto, imágenes, audio, etc.) que son, por tanto, *no estructurados* o *semi-estructurados*, a diferencia de las bases de datos. Además, los documentos son *hipertexto* o *hipermedia*, al hacer referencias a otros documentos a través de hipervínculos. Estos hipervínculos pueden ser recorridos o no por distintos usuarios, según las secuencias de *navegación* por la web. Esta diversidad permite minar la web basándose en tres conceptos: el **contenido**, la **estructura** y el **uso**.

Como veremos, la minería del contenido web reutiliza todas las técnicas de la minería de *textos* y mucha de la *recuperación de información*. De hecho, la minería de textos y de documentos de marcas la veremos en este capítulo englobada dentro de la minería de contenido web. A lo largo del capítulo introduciremos las nociones básicas de las tres modalidades (contenido, estructura y uso) e incluiremos descripciones y referencias de algunas de las técnicas empleadas en cada una de ellas.

21.1 Introducción

La *World Wide Web* es el repositorio más grande y ampliamente conocido de hipertexto. Un documento hipertexto es una colección de caracteres (texto) que puede contener, a través de los hipervínculos, referencias a otros documentos distribuidos en la web. Estos documentos o páginas web están escritos en una gran diversidad de idiomas y abarcan todos los tópicos del conocimiento humano.

La web ha experimentado un crecimiento exponencial desde su aparición en 1990. El código inicial fue escrito por Berners-Lee en el Laboratorio de Física de Altas Energías (CERN) en Suiza. Como él mismo afirmó: "el principal objetivo de la web fue tener un espacio de información compartido a través del cual máquinas y personas pudieran comunicarse". Estaba especialmente interesado en que se pudieran comunicar máquinas y *software* de diferentes tipos. Para ello, desarrolló un identificador de recursos universal (*Uniform Resource Locator, URL*) para poder referirse a cualquier documento (u otro tipo de recurso) en el universo de información. Asimismo, en lugar del protocolo de transferencia de archivos utilizado en ese momento para el intercambio de información, creó a partir de él un protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol, HTTP*) más rápido que el primero y un lenguaje de marcas para hipertexto (*HyperText Markup Language, HTML*).

Actualmente, Internet (incluyendo dentro de este término también el correo electrónico) es el medio más popular e interactivo de difundir información. Pero esta situación hace que a menudo los usuarios tengamos una sobrecarga de información. Según [Kosala & Blockeel 2000] algunos de los problemas con los que nos encontramos cuando interactuamos con la web son:

- Encontrar información relevante: cuando un usuario utiliza servicios de búsqueda para encontrar una información específica en la web, normalmente introduce una pregunta con las palabras clave y obtiene como respuesta una lista de páginas ordenadas según su similitud con la pregunta. Sin embargo, estas herramientas de búsqueda tienen, por lo general, una precisión bastante baja debido a la irrelevancia de muchos de los resultados de la búsqueda. A esto se une su limitada memoria que las hace incapaces de indexar toda la información disponible en la web, por lo que se hace incluso más necesario encontrar la información relevante a la pregunta.
- Crear nuevo conocimiento: la relevancia de la información obtenida en las consultas a la web es un problema estrechamente relacionado con el de crear nuevo conocimiento a partir de la información disponible en la web, es decir, una vez obtenidos los datos tras el proceso de búsqueda probablemente queramos extraer coincidencias, resúmenes, patrones, regularidades y, al fin y al cabo, conocimiento a partir de estos datos. Podemos decir, que si encontrar información en la web es un proceso orientado a la recuperación, la obtención de conocimiento útil es un proceso orientado a la minería de datos.
- Personalización de la información: a menudo se asocia este problema con la presentación y el tipo de la información, ya que los diferentes usuarios suelen tener gustos distintos a la hora de preferir ciertos contenidos y presentaciones cuando interactúan con la web. Muy relacionado con este problema está el de aprender de los usuarios, es decir, saber qué es lo que los usuarios hacen y quieren. Esto permite personalizar la información incluso para un usuario individual (diseño de portales web, de herramientas *software*, filtros de correo, etc.).

La enorme cantidad de información disponible hace de la web un área fértil para la minería de datos cuyas técnicas pueden resolver los problemas que acabamos de mencionar. Para ello, la minería web se nutre de técnicas de otras áreas de investigación como las bases de datos, la recuperación de información (*Information Retrieval, IR*) [Salton & McGill 1983; Baeza-Yates & Ribeiro-Neto 1999], el procesamiento del lenguaje natural (*Natural Language Processing, NLP*)

Processing, NLP) [Manning & Schütze 1999] y la inteligencia artificial [Russell & Norvig 2002], especialmente el aprendizaje automático.

Sin embargo, a diferencia de las bases de datos relacionales que poseen una estructura bien definida, la web es poco estructurada por naturaleza. Esto significa que muchas de las técnicas de minería de datos vistas hasta ahora no pueden aplicarse directamente, deben modificarse o, incluso, deben definirse nuevas técnicas. De hecho, tradicionalmente, la minería de datos se ha aplicado a las bases de datos, ya que era un formato de fácil procesamiento por los computadores, mientras que la información en la web reside en documentos enfocados al consumo humano tales como páginas personales, publicitarias, información general o catálogos de productos. Más aún, mucha de esta información se presenta como un texto en lenguaje natural, o bien como anotaciones HTML que estructuran la representación visual de las páginas web pero que proporcionan una escasa idea acerca de su contenido. Otras formas de estructurar la web incluyen ciertas convecciones lingüísticas y tipográficas, clases de documentos semi-estructurados como XML (*eXtensible Markup Language*) cada día de más uso para representar datos con cierta estructura, como los catálogos o los índices y directorios web.

Otros datos de interés residen en los archivos log, en los que los servidores registran información sobre las visitas que se efectúan a la web, y en las bases de datos que se generan a partir de otra información como, por ejemplo, la proporcionada por las *cookies*. Existen algunas herramientas de análisis de la web que pueden ser de utilidad al proporcionar respuestas a preguntas como ¿cuál es el orden más habitual al visitar los enlaces de nuestras páginas?, ¿cuántos nuevos visitantes tuvimos el mes pasado? ¿cuál es la media de visitas de un cliente?, ¿cuánto tiempo está un cliente en nuestras páginas? Estas herramientas analizan y monitorizan el tráfico de la web y analizan los archivos log (normalmente transformando los datos a un formato inteligible como resúmenes agregados o grafos). En [Mena 1999] se incluye una comparación entre algunas de estas herramientas.

Las herramientas de análisis y estadísticas de sitios web que se proporcionan junto a los servidores web (frecuentemente mal etiquetadas bajo el término “minería web”) proporcionan vistas y resúmenes de los datos de un modo similar a las herramientas clásicas de representación y summarización estadísticas y las herramientas OLAP (comentadas en los capítulos 4, 5 y 6). Al igual que éstas, son buenas para generar informes agregados o gráficas, lo cual puede ser de gran interés para diseñar, administrar y manipular webs, pero no permiten realizar otras actividades, como la extracción de patrones sobre el comportamiento de los usuarios, o bien estudiar la relevancia y clasificación de páginas y documentos.

Las *verdaderas* herramientas de minería de datos pueden proporcionar al administrador de la web información adicional para responder a cuestiones mucho más sofisticadas, como, por ejemplo, ¿cuáles serían los visitantes más adecuados para una nueva línea de productos?, ¿cuál es el perfil de mis visitantes?, ¿qué organización del portal favorece las compras?, ¿qué páginas web fomentan el abandono del sitio web? Si bien es cierto que estas cuestiones podrían responderse con herramientas de análisis a base de tratar de definir criterios y perfiles y ver cuándo se cumplen y cuándo no (siguiendo un método de prueba y error), en un entorno tan dinámico como la web sería un proceso temporalmente muy costoso (cuando no imposible) y susceptible de cometer errores. Sin embargo, con herramientas de minería de datos, encontrar, por ejemplo, grupos de clientes a partir de

archivos log es casi inmediato usando técnicas de agrupamiento, o categorizar documentos sobre ciertos temas utilizando técnicas de clasificación, o determinar qué páginas llevan a comprar qué producto mediante reglas de asociación secuenciales, etc.

A continuación, definiremos el concepto de minería web, estudiaremos su relación con otros conceptos relacionados y presentaremos una clasificación de la minería web atendiendo al tipo de datos que se minan (el contenido, la estructura o el uso).

21.2 Minería web

Etzioni [Etzioni 1996] definió la minería web como el uso de técnicas de minería de datos para descubrir y extraer información automáticamente desde el *World Wide Web*.

21.2.1 El proceso de minería web

La minería web puede descomponerse en las siguientes subtareas:

1. Descubrimiento de las fuentes: localizar los documentos y servicios en la web.
2. Selección y pre-procesado de la información: extraer automáticamente información específica desde las fuentes web descubiertas.
3. Generalización: descubrir patrones generales desde los sitios web individuales así como desde múltiples sitios.
4. Análisis: validación y/o interpretación de los patrones minados.

La primera tarea hace referencia al proceso de recuperar los datos desde las fuentes textuales de la web, tales como los correos y los boletines electrónicos, los grupos de noticias, el texto en los documentos HTML (una vez procesadas las etiquetas) o cualquier otro tipo de documento hipertexto (pdfs, xml, etc.). El trabajo de descubrir las fuentes se centra principalmente en el uso de índices de documentos web. Esto ha dado lugar al desarrollo de una serie de herramientas (los llamados buscadores) que recuperan documentos relevantes, usando normalmente técnicas de recuperación basadas en palabras claves (una técnica clásica de la IR). La lista de documentos recuperados suele priorizarse de acuerdo a diferentes criterios de relevancia. Algunos de los índices más populares han sido creados por robots web como Google (<http://www.google.com/>), AltaVista (<http://www.altavista.com/>), Excite (que comercializa WebCrawler, <http://www.webcrawler.com/>), Lycos (<http://www.lycos.com/>) y Yahoo! (<http://www.yahoo.com/>). Aunque en sí estos buscadores no hacen actividades de minería de datos, su funcionalidad puede extenderse para incluir otras actividades del tipo minería.

La segunda tarea incluye cualquier proceso de selección y/o transformación de los datos originales obtenidos en la etapa anterior. Para lo primero, existen algunos sistemas de extracción de información, como por ejemplo Harvest [Brown et al. 1994], que es entre otras cosas capaz de encontrar el título y el autor de documentos Latex, o FAQ-Finders [Hammond et al. 1995], que extrae las respuestas a las preguntas más frecuentes (*Frequently Asked Questions*) a partir de archivos FAQ disponibles en la Web. La transformación de los datos seleccionados comprende cualquier pre-procesamiento tanto los orientados a eliminar las palabras de fin, las etiquetas, etc., como los destinados a obtener la representación deseada, por ejemplo en forma de frases, en lógica de primero orden, etc.

La tercera etapa, la de generalización, es la etapa central de la minería web y es en la que se realiza el proceso de minería en sí. Para ello, la minería web ha adaptado técnicas de la minería de datos (como las reglas de asociación, el agrupamiento, etc.), de la IR (como algunas técnicas para la categorización y la clasificación de textos) y ha desarrollado algunas técnicas propias, como por ejemplo el análisis de caminos (*web paths*) usado para extraer secuencias de patrones de navegación desde archivos log.

La última etapa se ocupa de desarrollar técnicas y herramientas que permitan el consumo humano del conocimiento minado (a menudo éste no es directamente utilizable por los analistas). Estas herramientas deben incorporar métodos estadísticos (para manipular los patrones), de visualización (para facilitar su análisis) así como el conocimiento explícito que sobre el dominio del problema posee el analista (para contrastar el conocimiento minado con el que se poseía anteriormente sobre el problema). La tecnología de los agentes inteligentes podrían ser un buen medio para construir herramientas automáticas de este estilo.

Todas estas tareas recuerdan a las tareas que componen el proceso general de extracción de conocimiento KDD visto en el Capítulo 2. De hecho, la definición de minería web es idéntica a la del proceso KDD salvo que aquí la fuente de los datos es la web.

21.2.2 Disciplinas relacionadas

Además de con el proceso de KDD, la minería web se asocia a los procesos de recuperación de la información (IR, del inglés *Information Retrieval*) y de extracción de la información (IE, del inglés *Information Extraction*), aunque no son exactamente lo mismo. Algunas de las diferencias apuntadas en [Kosala & Blockeel 2000] son:

- Minería web e IR: la IR tiene como objetivo principal el indexado de texto y la búsqueda de documentos útiles en una colección, aunque actualmente la investigación en IR incluye la modelización, la clasificación y categorización de documentos, interfaces de usuario, visualización de datos filtrados, etc. Es decir, la IR ([Salton & McGill 1983; Baeza-Yates & Ribeiro-Neto 1999]) está interesada en seleccionar documentos relevantes. La tarea de la minería web más relacionada es la de la clasificación y categorización de documentos web, los cuales pueden usarse para la indexación.
- Minería web e IE: la IE tiene como objetivo principal la transformación de una colección de documentos en información para que sea más fácilmente comprendida y analizada. En otras palabras, la IE se centra en extraer hechos relevantes desde documentos. Básicamente, existen dos tipos de IE: desde textos no estructurados y desde datos semi-estructurados. Los métodos clásicos de IE (por ejemplo, [Cardie 1997; Wilks 1997]) tratan con textos (no estructurados) escritos en lenguaje natural y tienen su raíz en la comunidad del procesamiento de lenguaje natural. Estos métodos se basan en algún tipo de pre-procesado lingüístico, como el análisis sintáctico, el análisis semántico y el análisis del discurso. Con la creciente popularidad de la web, se ha puesto de manifiesto que los sistemas clásicos de IE no son apropiados para medios tan dinámicos y diversos como la web y que es necesario sistemas estructurales que extraigan información desde documentos semi-estructurados. Estos sistemas estructurales (por ejemplo, [Muslea 1999; Kushmerick et al. 1997; Hsu &

Dung 1998]) utilizan meta-information, como las etiquetas HTML o los delimitadores. Para su construcción se usan técnicas de minería de datos y de aprendizaje automático, ya que construir los sistemas manualmente no es apropiado para este medio.

21.2.3 Clasificación de la minería web

Generalmente, en la literatura se clasifica la minería web en tres áreas de interés en función de la parte de la web que se mina: minería del contenido, minería de la estructura y minería del uso (véase la Figura 21.1).

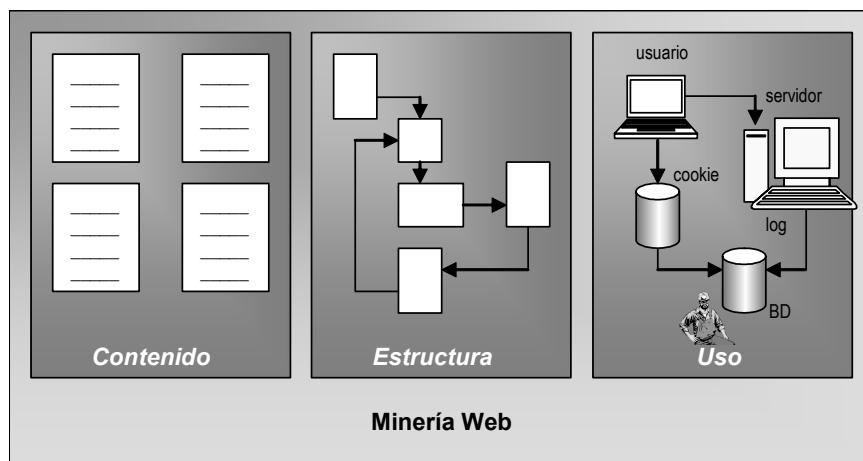


Figura 21.1. Clasificación de la minería web.

La **minería del contenido de la web** describe el descubrimiento de información útil desde los contenidos textuales y gráficos de los documentos web, y tiene sus orígenes en el procesamiento del lenguaje natural y en la recuperación de la información (tal y como hemos comentado en la sección anterior). Analiza, por tanto, documentos, más que los enlaces entre ellos. Los contenidos de la web han cambiado sustancialmente desde su origen. Al principio, Internet consistía en diferentes tipos de servicios y fuentes de datos, casi todos textuales y estáticos. Ahora, podemos encontrar una gran variedad de datos: librerías digitales accesibles desde la web, las bases de datos de muchas empresas que ofrecen electrónicamente sus negocios y servicios, aplicaciones y sistemas que están siendo migrados a la web o emergen en este entorno. De hecho, algunos de los datos en la web son ocultos ya que se generan dinámicamente o se obtienen como respuesta a preguntas cuyos datos residen en bases de datos privadas. Resumiendo, los contenidos en la web pueden ser de varios tipos: textual, imágenes, audio, vídeo, meta-datos e hipervínculos, y constan de datos no estructurados (texto), datos muy poco estructurados (como en los documentos HTML), datos semi-estructurados (como los documentos XML) y datos más estructurados (como los contenidos en bases de datos generadas desde páginas HTML). Sin embargo, como la mayoría del contenido corresponde a texto no estructurado, ésta es el área más investigada.

La **minería de la estructura de la web** trata de descubrir el modelo subyacente a la estructura de enlaces de la web y analiza, fundamentalmente, la topología de los hipervínculos (con o sin descripción de los enlaces). Este modelo puede usarse para

categorizar páginas web y es útil para generar información como la similitud y relación entre diferentes sitios web, así como para detectar páginas autoridades y páginas concentradores (que apuntan a páginas autoridades), estudiar topologías, etc.

La **minería de uso de la web** es el proceso de analizar la información sobre los accesos web disponible en los servidores web. A diferencia de las minerías de contenido y de estructura que usan datos reales sobre la web, la minería de uso mina datos secundarios derivados de la interacción de los usuarios mientras interactúan con la web. Estos datos incluyen los archivos de *logs* de acceso al servidor, *logs* del navegador, *logs* de los servidores *proxy*, perfiles de usuario, datos de registros, sesiones o transacciones del usuario, *cookies*, preguntas del usuario, pulsos del ratón y desplazamientos por las páginas, y en general cualquier otro dato fruto de la interacción.

21.3 Minería del contenido de la web

Dada la enorme cantidad de información disponible en la web y la gran diversidad de la misma, uno de sus principales usos es el de buscar información. La principal diferencia entre las técnicas de recuperación de información y las técnicas de minería del contenido de la web es que las primeras ayudan a los usuarios a encontrar documentos que satisfacen sus necesidades de información, mientras que las segundas permiten descubrir, reconocer o derivar información nueva a partir de uno o, generalmente, varios documentos.

La minería del contenido de la web ha sido principalmente aplicada para dos objetivos que corresponden a dos puntos de vista diferentes:

- visión desde IR: para asistir, mejorar o filtrar la información que los buscadores proporcionan a los usuarios a partir de los perfiles de los mismos (los cuales a su vez pueden haber sido inferidos o bien solicitados). Este objetivo también incluye los documentos que se reciben por correo, por grupos de noticias u otros medios diferentes de la navegación.
- visión desde Bases de Datos: modelar e integrar los datos encontrados en la web para permitir preguntas más sofisticadas que las búsquedas basadas en palabras clave. Muchas de las aplicaciones tienen por tarea la extracción de esquemas o la construcción de *DataGuides* ([Nestorov et al. 1998; Goldman & Widom 1999]), de las que hablaremos en la sección de minería de marcado (21.3.4).

Los diferentes tipos de datos contenidos en la web han dado lugar a diferentes técnicas de minería de datos para los diferentes formatos en los que éstos se presentan. Así, la aplicación de técnicas de minería a textos no estructurados se conoce como minería de textos (*Text Mining*), cuando se trata de texto semi-estructurado (XML, HTML, etc.) recibe el nombre de minería del marcado (*Markup Mining*), si se trata de datos multimedia hablamos de minería multimedia (*Multimedia Mining*), vista en el capítulo anterior, y, finalmente, si sólo nos referimos a los enlaces entre documentos o en el propio documento, pero sin tener en cuenta la estructura, recibe el nombre de minería de hipertexto (*Hypertext Mining*). De hecho, a los documentos HTML, al ser sus marcas fundamentalmente de formato y no de contenido, se les suele eliminar las marcas y se les trata como textos.

21.3.1 Minería de textos

El objetivo de la minería de textos es el descubrimiento de nueva información a partir de colecciones de documentos de texto no estructurado. Por no estructurado nos referimos a texto libre, generalmente en lenguaje natural, aunque también podría ser código fuente u otro tipo de información textual. La tarea de minería más habitual sobre estos datos es la categorización, la clasificación y el agrupamiento de los textos. Podemos decir que la categorización es la tarea que identifica las categorías, temas, materias o conceptos presentes en los textos, mientras que la clasificación es la tarea de asignar una clase o categoría a cada documento. Existen en la literatura otras definiciones diferentes para la categorización de textos, como la de [Dumais et al. 1998]: “la asignación de textos en lenguaje natural a una o más categorías predefinidas basadas en sus contenidos”. Otros autores tienden a ver la categorización como una parte de la clasificación, por lo que categorización y clasificación se usan como sinónimos. Nosotros aquí usaremos la siguiente taxonomía:

- agrupamiento de documentos: para organizar los documentos entorno a una jerarquía basándose en alguna medida de similitud.
- identificación de categorías: extracción de términos significativos (es muy parecido al análisis de relevancia de atributos y está relacionado con el agrupamiento).
- categorización: asignar una o más categorías a un documento (ésta es la que se usa en el resto del libro).
- clasificación: asignar una (y sólo una) clase a un documento.
- asociaciones: generalmente entre conceptos más que entre palabras.

Una aproximación muy usual a la categorización, si se tienen pocas categorías, digamos n , es convertir el problema en n problemas de clasificación binaria, en el que cada clasificador i se limite a decir si el documento es de la clase i o no.

La minería automática de textos juega un papel importante en una amplia variedad de tareas de manipulación de la información más dinámicas y personalizadas, como en la ordenación en tiempo real de correo electrónico o archivos en jerarquías de carpetas, en el filtro del correo electrónico, búsqueda estructurada y/o en los navegadores web, identificación de tópicos para soportar operaciones de procesamiento específicas a un tópico, catalogación de nuevos artículos y páginas web y en los agentes de información personal.

En la minería de textos lo primero a realizar es representar el texto en algún formato concreto que pueda ser adecuado para los algoritmos de aprendizaje. Esto se realiza en dos pasos. El primero consiste en usar una representación más abstracta, siendo las más habituales en IR las siguientes:

- bolsas de palabras (*bag of words* [Sahami et al. 1996; Lagus et al. 1999]): llamada también representación basada en vectores, ya que cada documento se representa como un vector de dimensión J , siendo J el número de palabras y en donde cada palabra constituye una componente del vector y representa una característica, la cual puede ser booleana (aparece o no en el documento) o basada en frecuencias (el número de veces que ha aparecido en el documento). Esta representación ignora el

orden de aparición de las palabras en el texto y es una de las más empleadas en el área de la IR.

- frases ([Frank et al. 1999]): esta representación consiste simplemente en considerar el documento como un conjunto de frases sintácticas, tal y como se hace en el análisis del procesamiento de lenguaje natural. Esta representación permite mantener el contexto en el que ocurre una palabra, hecho que se pierde en la representación anterior.
- *n-gramas* ([Kargupta et al. 1997a]): permiten usar la información sobre la posición de la palabra en el texto, ya que éste se representa mediante secuencias de palabras de longitud máxima n , llamadas *n-gramas*. Permiten un mejor tratamiento de las frases negativas como "... excepto ..." o "... pero no ..." que de otra forma tomarían como relevantes las palabras que les siguen.
- representación relacional ([Cohen 1995b]): la representación usando lógica de primer orden permite detectar patrones más complejos. Por ejemplo, cada palabra se puede representar mediante un átomo de la forma $w(d,p)$, el cual es cierto cuando la palabra w ocurre en el documento d en la posición p .
- categorías de conceptos ([Deerwester et al. 1990]): también llamado Indexación Semántica Latente (*Latent Semantic Indexing*) ya que tiene como objetivo la reducción de la dimensión del vector de palabras inicial, reduciendo las palabras a su raíz morfológica, es decir, las palabras "informando", "información", "informado" e "informador" se representarían por su raíz "inform" y sólo esta palabra se usaría como componente del vector. Esta reducción tiene que ser cuidadosa, ya que otras palabras aparentemente con la misma raíz pueden no tener relación con el término. En el ejemplo anterior, "informal" e "informática" tienen poca relación.

Casi todas estas representaciones se enfrentan al problema del vocabulario ([Furnas et al. 1987]), es decir, tienen errores semánticos debido a la sinonimia (diferentes palabras con el mismo significado), la quasi-sinonimia (palabras relacionadas con la misma materia, como declaración y comunicado), la polisemia (palabras iguales con diferente significado), los lemas (palabras con el mismo radical, como descubrir y descubrimiento), etc. Aunque se han realizado algunos estudios comparando las distintas representaciones (como por ejemplo [Scott & Matwin 1999]), en general no se han encontrado diferencias sustanciales en cuanto a las prestaciones de los algoritmos usando una u otra, aunque en un problema específico sí que pueden aparecer diferencias.

El segundo paso consiste en reducir el conjunto de características original (reducción de la dimensionalidad en el área del reconocimiento de patrones), ya que el conjunto de características que resultan de las representaciones descritas puede ser de cientos de miles, algo inabordable para muchos de los algoritmos de aprendizaje inductivos. La primera aproximación consiste en eliminar palabras con poca semántica, como son los artículos, preposiciones y conjunciones. En [Moulinier 1996] se describen dos maneras más elaboradas para reducir la dimensionalidad del vector basadas en el ámbito y en la naturaleza del problema. La reducción por ámbito tiene que ver con la universalidad del conjunto de características, mientras que la reducción por naturaleza describe cómo se seleccionan los atributos (por filtrado o por transformación, como se vio en los capítulos 4 y 5).

Se han empleado un gran número de técnicas del aprendizaje automático y estadísticas a la categorización de textos, incluyendo modelos de regresión multivariante ([Yang & Chute 1994]), clasificadores del vecino más próximo ([Yang 1994]), modelos bayesianos ([Joachims 1996; Lewis & Ringuette 1994]), árboles de decisión ([Lewis & Ringuette 1994]), redes neuronales ([Schütze et al. 1995]), aprendizaje de reglas simbólicas ([Cohen 1995b]) y máquinas de vectores soporte ([Joachims 1998]). A continuación vamos a ver algunas de estas técnicas haciendo uso de diferentes representaciones.

21.3.2 Clasificador de textos por palabras

Consideremos que disponemos de documentos de texto T , que pueden clasificarse en varias clases c , (por ejemplo, interesante vs. no-interesante) siendo Ω_c el conjunto de posibles clases. Usaremos como representación la bolsa de palabras la cual es equivalente a la representación atributo-valor del aprendizaje automático: un documento es un vector $\langle \omega_1, \omega_2, \dots \rangle$ donde ω_i es la i -ésima palabra del documento. Esto genera un problema con miles de atributos. En el capítulo de máquinas de vectores soporte (Capítulo 14) vimos una aproximación usando máquinas de vectores soporte, ya que es una técnica que permite tratar problemas de alta dimensionalidad.

Aquí, vamos a presentar la aplicación del clasificador probabilístico Naive Bayes (visto en el Capítulo 10) a la clasificación de textos, ya que Naive Bayes permite tratar también con problemas de alta dimensionalidad. Asumiremos que las palabras dentro de un documento son independiente entre sí (una suposición que aún no siendo cierta no es demasiado grave en general y nos permite utilizar este clasificador).

La representación elegida es la de los vectores de palabras, lo que significa que si el conjunto de palabras consideradas o vocabulario V consta de todas las palabras que ocurren en los documentos de entrenamiento, cada vector tendrá una longitud igual a $|V|$. De acuerdo a lo visto en la Sección 10.3, la clasificación con mayor precisión se obtiene a partir de la expresión:

$$c_{MAP}(d) = \arg \max_{c \in \Omega_c} p(c | d)$$

donde $p(c|d)$ es la probabilidad de que el documento d sea de la clase c . Si consideramos que la clase de un documento es independiente de su longitud y que la ocurrencia de una palabra sólo depende de la clase del documento y que es independiente de las otras palabras que también ocurren en el documento, o dicho de otra forma, que es independiente de la longitud del documento, entonces $p(c|d)$ puede aproximarse por

$$p(c | d) = \prod_{i=1}^{|V|} p(\omega_i | c)$$

donde ω_i son las componentes del vector de palabras del documento d . Ya que muchas $p(\omega_i | c)$ serán 0 hemos de adoptar un m -estimado (véase la Sección 5.4.1) para $p(\omega_i | c)$ con el objetivo de que el producto no sea 0:

$$p(\omega_i | c) = \frac{n_k + mp}{n + m} = \frac{n_k + |V| / |V|}{n + |V|} = \frac{n_k + 1}{n + |V|}$$

donde

- n_k es el número medio de ocurrencias de la palabra ω_i en los documentos de la categoría c .
- n es el número medio de palabras en los documentos de entrenamiento.
- p es la probabilidad de la palabra ω_i .
- aunque $|V|$ es el número de palabras del lenguaje considerado (inglés, castellano, etc.), también se puede usar un subconjunto eliminando las palabras más habituales, como preposiciones, artículos, verbos muy comunes, etc.

Finalmente, tenemos

$$c_{NB}(d) = \arg \max_{c \in \Omega_C} p(c) \cdot \prod_{i=1}^{|V|} p(\omega_i | c)$$

Resumiendo, la Figura 21.2 muestra el algoritmo Naive Bayes para la clasificación de textos.

```

ALGORITMO EntrenamientoNaiveBayes( $T$ :conjunto de documentos,  $\Omega_C$ : conjunto de clases  $c_1, c_2, \dots$ )
   $V :=$  todas las palabras ( $\omega_1, \omega_2, \dots$ ) y signos extraídos de  $T$ 
  // Comienza cálculo de  $p(c_i)$  y  $p(\omega_k | c_i)$ 
  PARA CADA clase  $c_i$  en  $\Omega_C$ 
     $docs_i :=$  documentos de  $T$  de la clase  $c_i$ 
     $p(c_i) := |docs_i| / |T|$ 
     $text_i :=$  concatenación de todos los elementos de  $docs_i$ 
     $n :=$  número total de posiciones de palabras distintas en  $text_i$ 
    PARA CADA palabra  $\omega_k$  en  $V$ 
       $n_k :=$  número de veces que la palabra  $\omega_k$  aparece en  $text_i$ 
       $p(\omega_k | c_i) := (n_k + 1) / n + |V|$ 
    FIN PARA
  FIN PARA
FIN ALGORITMO

```

Para clasificar un nuevo documento D se siguen los siguientes pasos:

1. obtener las palabras a_k de D
2. $posiciones :=$ todos los i tales que $\omega_i = a_k$ (palabras de V encontradas en D)
3. $c_{NB} = \arg \max_{\omega_j \in V} p(\omega_j) \prod_{j \in posiciones} p(a_i | \omega_j)$

c_{NB} es la clase del documento.

Figura 21.2. Algoritmo Naive Bayes para la clasificación de textos.

Como ejemplos de aplicación podemos nombrar que en [Joachims 1996] se hace uso de este clasificador para clasificar un artículo en 20 grupos de noticias. A partir de 1.000 artículos de cada grupo que se usaron de entrenamiento, la precisión de la clasificación para nuevos mensajes fue del 89 por ciento. En [Lang 1995] se presenta el sistema NewsWeeder que también usa un modelo probabilístico junto con métodos colaborativos (véase la Sección 21.5.2) para clasificar artículos y noticias dependiendo del interés que han creado en el usuario. Después de una etapa de entrenamiento, del 16 por ciento de artículos que el usuario había marcado como interesantes se pasó a un 59 por ciento de los que el sistema recomendaba. Otro ejemplo muy resaltante son los filtros de *spam* de la mayoría de lectores de correo que están basados en variantes del clasificador Naive Bayes.

21.3.3 Reglas de asociación para conceptos

Esta aproximación, en lugar de analizar las palabras, se basa en conceptos extraídos desde los textos. Los conceptos representan atributos del mundo real (como eventos, objetos, acciones, etc.) que ayudan a comprender las ideas o conceptos presentes en los textos. Por lo tanto, es necesaria una tarea previa de identificación de categorías para crear las definiciones de los conceptos, para luego aplicar otras tareas de minería que descubran patrones, analizando y relacionando los conceptos en una colección de textos.

Un hecho relevante a los conceptos es que éstos pertenecen al conocimiento extra-lingüístico sobre el mundo, es decir, dependen de quién hace el texto, con qué propósito y en qué contexto. Por ejemplo, en un entorno bancario uno querría poder identificar conceptos como “saldo”, “balance”, “crédito”, “interés”, “dinero”, etc. Los conceptos en un entorno empresarial podrían ser “ventas”, “compras”, “promoción”, “campaña publicitaria”, etc.

Como representación usaremos vectores ([Loh et al. 2000]) de tal forma que cada concepto se almacena como un vector de términos, los cuales pueden incluir sinónimos, quasi-sinónimos, variaciones léxicas, plurales, derivaciones verbales... Basándonos en la idea de que cada palabra de un concepto contribuye con cierta fuerza en la presencia de este concepto, cada término en un vector tiene asociado un peso, comprendido entre cero y uno, que indica la importancia relativa del término en el concepto.

Además, el mismo término puede pertenecer a la descripción de más de un concepto. Por ejemplo, el concepto *crédito* puede estar definido a través de los términos {*crédito*, *préstamo*, *créditos*, *prestamista*, *anticipo*, *acreedor*, *fiador*, *interés*}. Cada uno de estos términos participa de forma diferente a la presencia del concepto en el texto, a través de unos pesos, tal y como se ilustra en la siguiente tabla:

<i>crédito</i> →	1	<i>crédito</i>	0,9
	2	<i>préstamo</i>	0,8
	3	<i>créditos</i>	0,8
	4	<i>prestamista</i>	0,5
	5	<i>anticipo</i>	0,1
	6	<i>acreedor</i>	0,4
	7	<i>fiador</i>	0,3
	8	<i>interés</i>	0,5

Tabla 21.1. Definición del concepto *crédito* a través de un vector de términos y sus pesos asociados.

Como se observa en la Tabla 21.1, el término *anticipo* tiene un peso menor que el término *préstamo*. Esto significa que cuando en el texto aparece el término *anticipo* es menos probable que en el documento se encuentre el concepto *crédito* que cuando ocurre el término *préstamo*.

Una vez definidos los conceptos (esto puede hacerlo directamente el usuario, usando diccionarios, tesauros, etc., o bien con la ayuda de herramientas *software* para la determinación de los pesos, tales que presenten al usuario la lista de palabras y su frecuencia de aparición para que éste determine qué palabras son más comunes y en qué contexto), se comparan todos los textos contra cada concepto. Haciendo uso de los pesos de aquellas palabras que aparecen en el texto y en la definición del concepto se calcula el grado de relación entre el texto y el concepto, lo que nos permite calcular la probabilidad

relativa de que el concepto esté presente en el texto. Es decir, si en el texto aparecen palabras que describen un concepto, existe una alta probabilidad de que el concepto esté presente en el mismo. Para decidir si el concepto está o no presente en un texto se puede usar un umbral definido por el usuario (o bien establecido en una sesión de entrenamiento anterior).

El segundo paso es asociar conceptos con conceptos, a través de asociaciones, como se ve en el Capítulo 9. De esta forma se pueden descubrir reglas de la forma $A \rightarrow B$, con A siendo o un conjunto de conceptos o bien un único concepto, mientras que B es un concepto. La regla significa: “*si A está presente en el texto, entonces B está también presente con una cierta confianza y un cierto soporte*”. La *confianza* es la proporción de textos que tienen (A y B) en relación con el número de textos que tiene sólo A . El *soporte* es la proporción de textos que tienen (A y B) en relación con el número de textos totales. Por ejemplo, después de analizar diez documentos, podríamos haber obtenido una regla del tipo *crédito Y promoción* \rightarrow *compras* con un soporte = 0,5 (cinco documentos de los diez) y una confianza = 83,3 por ciento.

21.3.4 Minería de mercado

La naturaleza semi-estructurada de los datos de la web es inadecuada para el almacenamiento y la recuperación de información en forma de bases de datos relacionales. Ésta es la principal causa de que no existan lenguajes de interrogación, como los de las bases de datos, capaces de recuperar información basándose, por ejemplo, en la estructura de los documentos HTML. Para subsanar este problema se han introducido sistemas basados en mediadores o en almacenes de datos. Ambos tipos de sistemas usan esquemas o modelos de datos para representar los datos no estructurados o semi-estructurados. El resultado es un Sistema de Gestión de Bases de Datos que el usuario puede interrogar directamente, en lugar de interactuar directamente con la web. Mediadores y almacenes usan componentes *software*, llamados *wrappers*, para extraer datos desde la web. El propósito de los *wrappers* es filtrar y transformar los datos de la web a un formato estructurado (una base de datos) para que puedan utilizarse lenguajes de interrogación a bases de datos, que no se pueden utilizar directamente sobre los datos recuperados de la web por carecer éstos de estructura (o ser semiestructurados). En este apartado vamos a describir el sistema Lore (*Lightweight Object Repository* [McHugh et al. 1997]) que integra bases de datos con la web.

Lore es un sistema de gestión de bases de datos semi-estructuradas [Abiteboul et al. 1997]. El modelo de datos usado es OEM (*Object Exchange Model*), en el que los datos se representan como un grafo dirigido etiquetado donde los nodos son objetos. Un objeto consta de un identificador único (por ejemplo, *&1*), una etiqueta (por ejemplo, *nombre*), un tipo (por ejemplo, *string*) y un valor (por ejemplo, “*La Ilíada*”). Los objetos se descomponen en atómicos y complejos. Los atómicos tienen valor y se reconocen porque no tienen ejes salientes. Todos los demás objetos son complejos y se caracterizan porque tienen uno o más ejes salientes. A continuación se muestra a modo de ejemplo parte de los datos correspondientes a una biblioteca:

```

Biblioteca (&b)
  libro (&1)
    nombre (&2) "La Iliada"
    escrito-por (&3)
  
```

genero (&4) "novela épica"
libro (&5)
 nombre (&6) "La Fundación"
 escrito-por (&7)
 genero (&8) "ciencia ficción"
libro (&9)
 nombre (&10) "El baile de la Victoria"
 escrito-por (&11)
 genero (&12) "novela"
 premio (&13) "Planeta"
escritor (&3)
 nombre (&14) "Homero"
 autor (&1)
escritor (&7)
 nombre (&15) "Isaac Asimov"
 autor (&5)
 nacionalidad (&16) "Americana"
escritor (&11)
 nombre (&17) "Antonio Skármeta"
 autor (&9)
libro (&18)
 nombre (&19) "El cantar del mío Cid"

Estos datos se pueden representar gráficamente usando la notación OEM, tal y como se ilustra en la Figura 21.3.

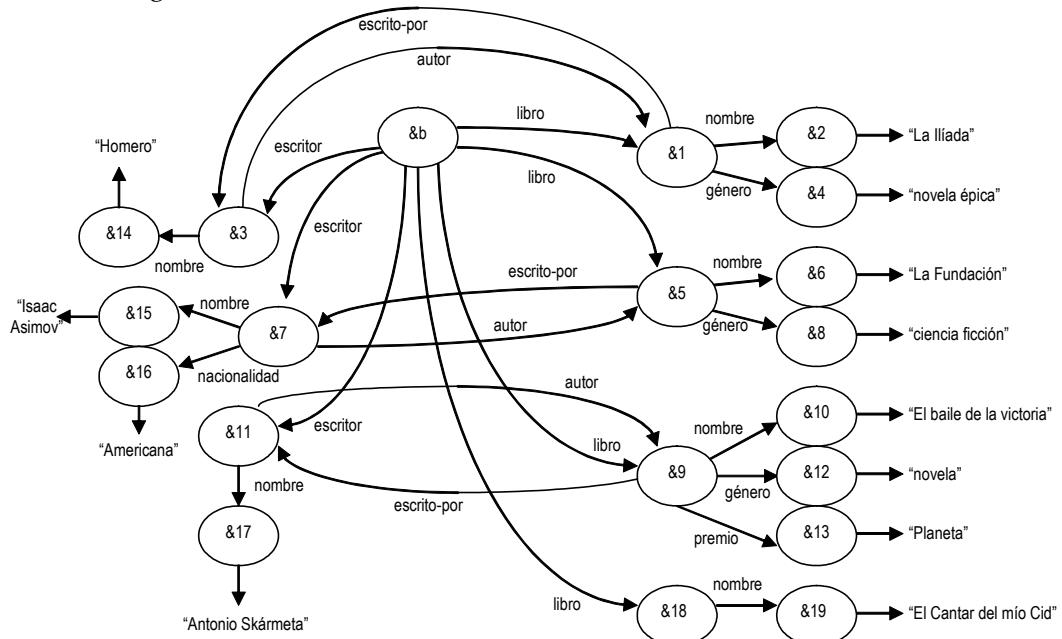


Figura 21.3. OEM para los datos de una biblioteca.

A partir del modelo OEM es posible obtener grafos de esquemas, los llamados *Dataguides* introducidos por primera vez en el proyecto Lore, que intentan ser un resumen preciso y conciso de los grafos de datos. La idea es recorrer todos los caminos posibles desde la raíz,

creando los nodos y los arcos necesarios para que todos tengan una representación en el esquema. Por ejemplo, desde el nodo raíz `&b` hay un camino, “libro”, que nos lleva a los nodos `&1, &5, &9, &18`; creamos entonces un nuevo nodo “Libro” y lo enlazamos al nodo raíz con la etiqueta “libro”. La siguiente figura muestra el *Dataguide* del grafo de la Figura 21.3.

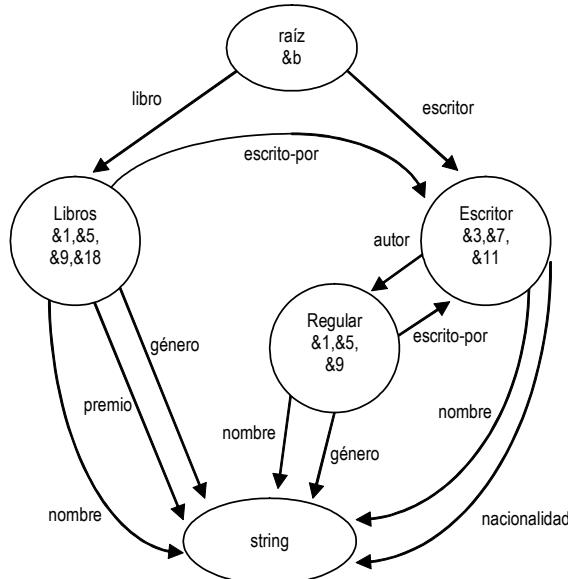


Figura 21.4. Dataguide correspondiente al modelo OEM de la Figura 21.3.

Lore usa un lenguaje de interrogación llamado Lorel que usa expresiones de caminos en el grafo de los datos, devolviendo el conjunto de nodos que satisfacen el camino. Por ejemplo, la siguiente pregunta busca libros escritos por Isaac Asimov:

```
select nombre: N
from libro L, libro.nombre N, libro_escrito-por A
where A="Isaac Asimov"
```

21.3.5 Minería de hipertextos

La información de la estructura de la página se refiere a la estructura interna del documento en HTML o XML, la cual se representa normalmente como un árbol (*Document Object Model, DOM*, <http://www.w3.org/DOM/>) [Chakrabarti 2000]. Las hojas del árbol contienen fragmentos de texto, mientras que los otros nodos son hipervínculos a otras páginas web o etiquetas HTML. Hacer uso de las etiquetas puede ayudar en la clasificación de las páginas. En realidad se trata de una mezcla de minería de contenido y de minería de estructura. Por ejemplo, consideremos el siguiente fragmento escrito en notación XML que describe la información disponible en una biblioteca sobre las revistas que recibe:

```
...
<publication>
  <name> Communications of the ACM</name>
  <type>Journal</type>
  <topic> Learning</topic>
  <topic> Software Engineering</topic>
  <topic> Data Mining</topic>
```

```

</publication>
<paper>
  <title> Learning to Personalize</ title >
  <author>H. Hirsh</author>
  <author>C. Basu </author>
  <author> B.D. Davison </author>
</paper>
...

```

Dependiendo de la tarea de clasificación, podría ser importante distinguir entre las dos ocurrencias de la palabra *Learning* (por ejemplo, para clasificar la revista por tópicos sería más importante el primero que el segundo). Esto puede hacerse fácilmente usando las etiquetas: *publication.topic.Learning* y *paper.title.Learning*. Aunque algunas veces esta forma de actuar es suficiente para alcanzar mejor precisión en la clasificación de la que se obtendría con clasificadores sólo de texto, en general es demasiado *ad-hoc* e inflexible.

Otra alternativa que nos proporciona una forma uniforme de codificar los hipertextos es el uso de predicados, es decir una representación relacional ([Craven et al. 1998]). Por ejemplo, podemos representar con predicados información diversa sobre un documento, como *has_word(A,Topic)* (que nos dice si el documento *A* contiene palabras del tema *Topic*), *classified(A,label)* que nos dice que el documento *A* es de la clase *label* o *link_to(A,B)* (que nos informa que existe un enlace desde el documento *A* al *B*).

Usando esta representación podemos definir reglas que definan relaciones entre los documentos web. Por ejemplo, una regla para ver si una página web contiene artículos online sobre un tópico podría ser

```

web_with_papers(A,Topic):- has_word(A,Topic),
  link_from(A,B),
  (URL_has_word(B,'ps'); URL_has_word(B,'pdf')),
  not is_html(B).

```

donde hemos usado la notación Prolog en la que ‘:-’ se lee como “si”, la coma representa la conjunción y el punto y coma la disyunción. Por tanto, la regla significa que la web contiene artículos de un tema si aparece el tema en la página y tiene un link a un documento ps o pdf.

21.4 Minería de la estructura de la web

Tal y como hemos comentado en la introducción, uno de los principales problemas cuando interactuamos con la web es la de encontrar información interesante. Los buscadores basados en índices (como Google, AltaVista, Yahoo!, Excite o InfoSeek) han sido unas de las primeras herramientas con las que han contado los usuarios para buscar información en la web. Si bien son útiles para usuarios experimentados o cuando se buscan páginas sobre un tópico muy concreto, pueden no ser tan adecuadas para un concepto muy general contenido en miles o millones de páginas, lo que obligaría al usuario a revisar un excesivo número de páginas. Por lo tanto, a la hora de buscar tópicos en la web cuyo resultado sea de un tamaño razonable para el ser humano, necesitamos identificar las páginas web más significativas o definitivas (autoridades) en el tópico. Esta noción de autoridad añade una segunda dimensión crucial a la noción de relevancia: deseamos no sólo localizar un conjunto de páginas relevantes sino que además sean de una alta calidad. En segundo

lugar, la web consta no sólo de páginas sino también de hipervínculos que conectan una página a otra. Estos hipervínculos representan la intención por parte del autor de “incluir” la página referenciada, lo cual puede ser de interés para inferir automáticamente la noción de autoridad y hacernos una buena idea de la relevancia y calidad de los contenidos de la web.

En el modelo más simple, el hipertexto se representa como un grafo (D, L) donde D es el conjunto de páginas o documentos y L el conjunto de enlaces. Dado que la teoría de las redes sociales estudia las propiedades relacionadas con la conectividad y las distancias en grafos, recientemente se ha aplicado este tipo de análisis y el análisis de citaciones al grafo de la web con el propósito de identificar las páginas más autoritativas con relación a la pregunta del usuario ([Chakrabarti 2003]) recogiendo la idea intuitiva de que el documento más citado o más referenciado es el más importante.

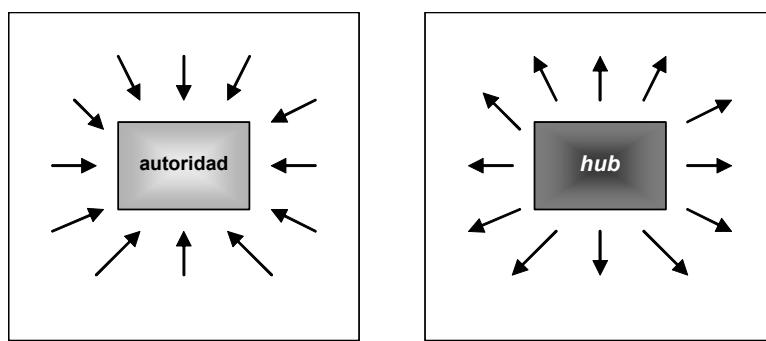


Figura 21.5. Representación de los vínculos a páginas autoridades y concentradores.

Veamos un caso concreto. En [Kleinberg 1999] se presenta un algoritmo denominado “búsqueda de tópicos inducidos por hipervínculos” (*hyperlink-induced topic search, HITS*) que analiza la topología de los hipervínculos para descubrir dos tipos interrelacionados de páginas: autoridades (*authorities*) y concentradores (*hubs*) (véase Figura 21.5). Las autoridades y los *hubs* tienen una relación de refuerzo mutuo, ya que un *hub* es mejor cuando enlaza a muchas buenas autoridades, y una autoridad también es mejor cuando es enlazada por buenos *hubs*. Este tipo de análisis recibe el nombre de análisis de conectividad. En la figura se muestra esquemáticamente que una autoridad recibe muchos enlaces mientras que un *hub* sirve como un resumen o lista de muchos enlaces a otros documentos autoridades.

En el primer paso del algoritmo HITS, el usuario hace una pregunta con el concepto o asunto sobre el que está interesado en encontrar autoridades y *hubs*. Esta pregunta se reenvía a un buscador (como AltaVista) que devuelve un subgrafo de la web cuyos nodos emparejan con la misma. El grafo se expande para incluir tanto las páginas citadas por una página como las páginas que la citan a ella. Cada nodo u en este grafo expandido tiene asociado dos pesos a_u y h_u , siendo el primero su peso de autoridad y el segundo el de *hub*. Estos pesos se inicializan a 1 y son iterativamente ajustados de la siguiente forma:

$$a_u = \sum_{v \rightarrow u} h_v$$

$$h_u = \sum_{u \rightarrow v} a_v$$

donde $x \rightarrow y$ denota un enlace desde la página x a la página y . Los pesos se normalizan a 1 después de cada iteración y se repite el proceso hasta que los valores a y h convergen a un conjunto estable de pesos de autoridad y *hub*. Finalmente, el algoritmo devuelve una lista de páginas (más corta que la inicialmente obtenida con el buscador) con las páginas con los mayores pesos de autoridad y *hub*.

Tal y como hemos comentado, otra técnica alternativa muy parecida al HITS, llamada *PageRank* y utilizada por el buscador Google (<http://www.google.com/>) y el sistema Clever (<http://www.almaden.ibm.com/cs/k53/clever.html>), se basa en el análisis de citaciones sobre la topología de la web [Page et al. 1998].

En *PageRank*, cada página de la web tiene una medida de prestigio que es proporcional a la suma de los valores de prestigio de las páginas que la enlazan. Supongamos que pudiéramos navegar por la web por un tiempo infinito, haciendo uso de los enlaces de las páginas seleccionando cada vez un enlace de forma aleatoria. Supongamos que comenzamos desde un nodo arbitrario u de acuerdo a una distribución p tal que p_u es la probabilidad de comenzar en el nodo u siendo $\sum_u p_u = 1$. Consideremos la matriz de adyacencia E sobre la web definida como

$$E[u,v] = \begin{cases} 1 & \text{si } u \rightarrow v \\ 0 & \text{en cualquier otro caso} \end{cases}$$

El grado de salida de un nodo u , es decir, el número de páginas que se pueden alcanzar desde u es $N_u = \sum_v E[u,v]$, es decir, la suma de la u -ésima fila de la matriz. Dado que la web no está fuertemente conectada, es decir, no puedo pasar siempre de una página a otra ya que incluso existen páginas sin enlaces de salida, el navegador en cada nodo hace primero una elección de entre estas dos posibilidades:

1. con probabilidad d , saltar a una página aleatoria.
2. con probabilidad $(1-d)$, pasar a una página vecina elegida aleatoriamente de entre las que son directamente accesibles a través de un enlace de la página actual.

Donde d es una constante generalmente comprendida entre 0,1 y 0,2. Esto significa que las páginas serán visitadas con diferentes ratios, siendo las páginas más populares con muchos enlaces hacia ellas las que se tenderá a visitar más frecuentemente. Formalmente, esta medida de popularidad se define recursivamente como

$$\text{PageRank}(v) = d/N + (1-d) \sum_{u \rightarrow v} (\text{PageRank}(u)/N_u)$$

donde N es el número total de nodos en el grafo de la web.

El Google usa esta medida para ordenar las páginas que ha seleccionado y que emparejan con la pregunta realizada por el usuario. No obstante, este mecanismo no es el único que usa el Google, que también se vale de técnicas basadas en el contenido para seleccionar las páginas que emparejan con la pregunta. Ahora bien, ya que PageRank es una medida independiente de la pregunta, el Google precomputa esta medida sobre el grafo de la web, de tal forma que, una vez seleccionadas las páginas que emparejan con las palabras clave introducidas en la pregunta, su ordenación de acuerdo a este criterio es inmediata. Esto hace del Google un buscador tan rápido como los basados en textos. Otros sistemas, como Clever, han incorporado modificaciones de los pesos del grafo que dependen del texto de la página, lo que mejora considerablemente los resultados [Chakrabarti et al. 1998].

21.5 Minería de uso web

La web tiene actualmente un papel importante en el mundo de los negocios. Cada vez más las empresas diseñan sus propios sitios web, que se están convirtiendo en el primer punto de contacto entre empresas y clientes. Esto está generando la necesidad de conocer cómo los usuarios interactúan con estos sitios web. La minería del uso de la web captura las actividades de los usuarios durante su conexión y extrae patrones de comportamiento que pueden ayudar a comprender las preferencias de navegación de los usuarios, el comportamiento de los clientes o a mejorar futuras páginas adaptando las interfaces de los sitios web a los usuarios individuales.

Cuando los usuarios interactúan con un sitio web, los datos que registran su comportamiento se almacenan en los logs de los servidores web. Éstos pueden llegar a almacenar, en un sitio web de tamaño medio, varios megabytes por día.

Existen dos aproximaciones principales para minar los patrones de navegación de los usuarios desde los datos log:

- transformar los datos a notación tabular y aplicar técnicas estándar de minería de datos, como las reglas de asociación [Chen et al. 1998].
- desarrollar técnicas *ad-hoc* para trabajar directamente con los datos log [Spiliopoulou et al. 1999].

Por otra parte, las aplicaciones de la minería de uso pueden clasificarse en:

- aprendizaje de patrones de navegación.
- aprendizaje de perfiles de usuario para modelizar interfaces adaptativas (personalización).

A continuación presentaremos algunas de las técnicas aplicadas a estos dos tipos de usos.

21.5.1 Aprendizaje de patrones de navegación

Como hemos dicho antes, podemos transformar los datos a una notación tabular (pares atributo-valor) o podemos aplicar representaciones específicas. A continuación vamos a ver una de ellas.

En [Borges & Levene 2000] se presenta un método consistente en representar las sesiones de navegación de los usuarios inferidas desde los archivos log como una gramática probabilística de hipertexto (*hypertext probabilistic grammar*, HPG), tal que las cadenas generadas por la gramática con mayor probabilidad corresponden a los caminos preferidos por los usuarios. La HPG es una gramática regular (una tupla $\langle V, \Sigma, S, P \rangle$) con una relación uno-a-uno entre el conjunto de símbolos no terminales (V) y el conjunto de símbolos terminales (Σ). Cada símbolo no terminal corresponde a una página web y cada regla de producción de la gramática (P) a un enlace entre páginas. Existen dos estados artificiales S y F que representan los estados de inicio y fin de las sesiones de navegación. En la Figura 21.6 se ilustra el diagrama de transición de una HPG.

La probabilidad de una cadena de la gramática es el producto de las probabilidades de las producciones usadas en su derivación. Las producciones que comienzan con S son producciones de inicio, mientras que el resto (enlaces entre páginas) son producciones transitivas.

A partir de la colección de sesiones de navegación (como la que se ilustra en la Tabla 21.2) se obtiene el número de veces que una página fue requerida, el número de veces que fue el primer estado en una sesión y el número de veces que fue el estado final. El número de veces que una subsecuencia $A_i \rightarrow A_j$ de dos páginas aparece en las sesiones da el número de veces que el correspondiente enlace fue atravesado. Se define un parámetro α como el peso de un estado para ser la primera página en una sesión de navegación. Si $\alpha=0$ sólo los estados que fueron primeros en las sesiones actuales tienen probabilidad mayor que cero de ser una producción de inicio. En este caso, la gramática sólo genera cadenas que comienzan por un estado que fue primero en una sesión. Con valores $\alpha>0$ se pueden generar cadenas desde cualquier estado. En el caso extremo de $\alpha=1$, la probabilidad de una producción de inicio es proporcional al número de veces que el correspondiente estado fue visitado. En este caso, el nodo destino de una producción con más alta probabilidad corresponde al estado que fue visitado más a menudo. Variando este parámetro α se proporciona al analista la capacidad de buscar distintos patrones. Finalmente, la probabilidad de una producción transitiva es proporcional a la frecuencia con la cual el correspondiente enlace es atravesado. Un segundo parámetro N ($N \geq 1$) determina la memoria del usuario cuando se navega por la red, es decir, el número de URLs anteriores que pueden influir en la elección del siguiente URL. Si $N=1$ el resultado será lo que se conoce formalmente como una cadena de Markov.

Veamos un ejemplo. Supongamos la siguiente tabla de sesiones de navegación:

ID	Sesión
1	$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$
2	$A_1 \rightarrow A_5 \rightarrow A_3 \rightarrow A_4 \rightarrow A_1$
3	$A_5 \rightarrow A_2 \rightarrow A_4 \rightarrow A_6$
4	$A_5 \rightarrow A_2 \rightarrow A_3$
5	$A_5 \rightarrow A_2 \rightarrow A_3 \rightarrow A_6$
6	$A_4 \rightarrow A_1 \rightarrow A_5 \rightarrow A_3$

Tabla 21.2. Sesiones de navegación.

En este caso, $V=\{S, A_1, A_2, A_3, A_4, A_5, A_6, F\}$ y $\Sigma=\{a_1, a_2, a_3, a_4, a_5, a_6\}$. Tomando $\alpha=0,5$ y $N=1$ podemos calcular la probabilidad de cada producción. Por ejemplo,

$$p(S \rightarrow a_1 A_1) = \frac{\alpha \cdot N_{\text{visitas_} A_1}}{N_{\text{total_visitas}}} + \frac{\alpha \cdot N_{\text{inicios_} A_1}}{N_{\text{total_inicios}}} = \frac{0,5 \cdot 4}{24} + \frac{0,5 \cdot 2}{6} = 0,25$$

Similarmente, según la Tabla 21.2, la página A_4 fue visitada cuatro veces, una de las cuales como última página de una sesión, una vez en el camino hacia la página A_6 y dos veces en el camino hacia A_1 . Por lo tanto,

$$p(A_4 \rightarrow a_1 A_1) = \frac{2}{4}$$

$$p(A_4 \rightarrow a_6 A_6) = \frac{1}{4}$$

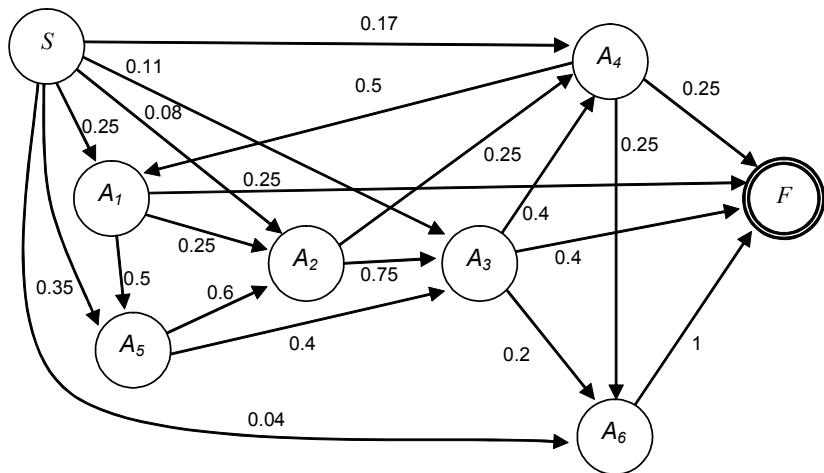
$$p(A_4 \rightarrow F) = \frac{1}{4}$$

La HPG completa se muestra en la Tabla 21.3.

Producción inicio	Producción transitiva	Producción final
$S \rightarrow a_1A_1 : 0,25$	$A_1 \rightarrow a_2A_2 : 0,25$	
$S \rightarrow a_2A_2 : 0,08$	$A_1 \rightarrow a_5A_5 : 0,50$	
$S \rightarrow a_3A_3 : 0,11$	$A_2 \rightarrow a_3A_3 : 0,75$	
$S \rightarrow a_4A_4 : 0,17$	$A_2 \rightarrow a_4A_4 : 0,25$	
$S \rightarrow a_5A_5 : 0,35$	$A_3 \rightarrow a_4A_4 : 0,40$	$A_1 \rightarrow F : 0,25$
$S \rightarrow a_6A_6 : 0,04$	$A_3 \rightarrow a_6A_6 : 0,20$	$A_3 \rightarrow F : 0,40$
	$A_4 \rightarrow a_1A_1 : 0,50$	$A_4 \rightarrow F : 0,25$
	$A_4 \rightarrow a_6A_6 : 0,25$	$A_6 \rightarrow F : 1,00$
	$A_5 \rightarrow a_2A_2 : 0,60$	
	$A_5 \rightarrow a_3A_3 : 0,40$	

Tabla 21.3. HPG correspondiente a las sesiones de la Tabla 21.2 para $\alpha=0,5$ y $N=1$.

La Figura 21.6 muestra la HPG representada como un diagrama de transacciones.

Figura 21.6. Diagrama de transición correspondiente a las sesiones de la Tabla 21.2 para $\alpha=0,5$ y $N=1$.

Las cadenas generadas por la gramática corresponden a las sesiones de navegación de los usuarios, y la intención es identificar el subconjunto de estas cadenas que mejor caracterizan el comportamiento del usuario cuando visita el sitio web. Por lo tanto, pueden servir para probar aplicaciones con las sesiones más comunes, calcular la probabilidad de alcanzar una página si el usuario está en una página dada, diseñar sitios web de acuerdo a las sesiones más habituales o detectar usuarios anómalos. Por ejemplo, cuando buscamos sesiones frecuentes, las seleccionas si la probabilidad de su derivación está por encima de un punto de corte λ . Este punto de corte se compone de dos umbrales, $\lambda=\theta \cdot \delta$, donde $\theta \in (0,1)$ es el umbral de apoyo o cobertura (*support threshold*) y $\delta \in (0,1)$ es el umbral de confianza (*confidence threshold*). El umbral de cobertura es el factor del punto de corte responsable de podar aquellas cadenas cuyo primer paso en la derivación tiene una baja probabilidad, lo que nos permite eliminar aquellas cadenas con alta probabilidad pero que son raramente visitadas ya que la probabilidad de la primera página es baja. El umbral de confianza permite podar aquellas derivaciones que contienen producciones transitivas con baja probabilidad. Asimismo, también se puede limitar la longitud de los patrones a un cierto valor k lo que influye en la eficiencia del algoritmo de búsqueda de los patrones.

Usando una heurística que permite una búsqueda iterativa en profundidad (ID), se puede construir un árbol de exploración desde el estado inicial que considere sólo sesiones

de longitud 1. Una vez construido el árbol con profundidad 1, si no se ha alcanzado el criterio de parada la profundidad del árbol se incrementa en 1 y se exploran todas las sesiones de longitud 2, y así sucesivamente. Cuando una sesión está siendo explorada, cada enlace de salida se evalúa para determinar si es una expansión admisible (si conduce a una sesión con probabilidad por encima del punto de corte) o inadmisible. La Figura 21.7 muestra un árbol de exploración en el que cada rama viene etiquetada con la probabilidad del enlace y la probabilidad de la sesión desde el estado inicial.

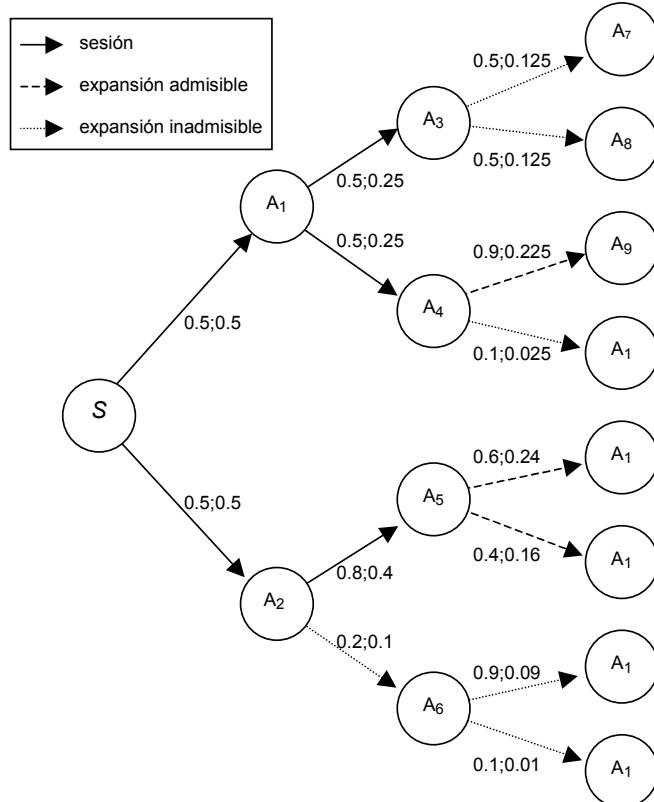


Figura 21.7. Árbol de exploración para $k=2$ y $\lambda=0,15$.

Para $k=2$ y $\lambda=0,15$ las sesiones candidatas son $\{A_1A_3, A_1A_4, A_2A_5\}$. Mirando un paso más a partir de la sesión A_1A_3 , se observa que las sesiones A_3A_7 y A_3A_8 son expansiones inadmisibles. Sin embargo, la expansión A_4A_9 de la sesión A_1A_4 es admisible.

21.5.2 Personalización y sistemas de recomendación

La personalización de la web puede definirse como cualquier acción que adapta la web al gusto del usuario a partir de los patrones de uso pasados. Estas acciones pueden ir desde hacer la presentación más placentera hasta anticiparse a las necesidades del usuario. Es importante establecer el pasado a analizar ya que si se trata de secuencias muy cortas necesitaremos muchas historias, mientras que si son demasiado largas puede que los patrones de uso de los usuarios hayan cambiado.

La personalización no es una cuestión exclusiva de la web. En general, existe un interés por personalizar cualquier sistema *software* que interactúe con los usuarios. Las técnicas

empleadas van desde los métodos colaborativos y los sistemas basados en refuerzo hasta técnicas del aprendizaje automático, principalmente el agrupamiento. Una aplicación muy usual son los sistemas recomendadores (*Recommender Systems*) [Resnick & Varian 1997]. A continuación, describiremos brevemente un ejemplo de cada uno de estos métodos.

Métodos colaborativos

En algunos casos nuestras preferencias se basan en los gustos de otros usuarios cuya opinión valoramos y compartimos (lo que se ha venido en llamar *word of mouth* [Hirsh et al. 2000]). Por ejemplo, muchas veces decidimos ir al cine a ver una película porque nos la han recomendado los amigos. Ésta es la idea que subyace en los métodos colaborativos, en los que los perfiles se hacen en base a la puntuación que los usuarios dan a los ítems y las predicciones se basan en encontrar usuarios que hayan asignado puntuaciones similares en experiencias previas. En [Shardanand & Maes 1995] se presenta el sistema Firefly que hace uso de métodos colaborativos para recomendar música a los usuarios. Para ello, obtiene puntuaciones de una muestra de cantantes y canciones del usuario X . Busca otros usuarios Y s que tengan un patrón de gusto similar a X (es decir, que hayan dado puntuaciones similares para los ítems comunes). Entonces, recomienda a X los artistas y canciones preferidos de los Y s.

En ocasiones, se combinan los métodos colaborativos con métodos basados en contenidos, como en [Basu et al. 1998] para recomendar películas. Algo similar hace "amazon.com" (<http://www.amazon.com/>) para recomendar libros, mostrando sugerencias del tipo "30 por ciento de los usuarios que seleccionaron/compraron este libro también buscaron estos otros tópicos/libros". Asimismo, se han usado métodos colaborativos para recomendar nuevos URLs a un usuario que visita una página a partir de los patrones de navegación de otros usuarios, o bien combinándolos con métodos basados en contenido, como en WebWatcher [Joachims et al. 1997].

Métodos basados en refuerzo

El sistema News Dude [Billsus & Pazzani 1999] es un sistema recomendador o calificador que se basa en refuerzo, es decir, se ajusta a las preferencias de mensajes de *news* del usuario y a las calificaciones realizadas por el usuario (el usuario refuerza o no las calificaciones realizadas por el sistema).

El objetivo del sistema es encontrar *news* no leídas y que puedan interesar al usuario. News Dude descarga noticias sobre política, negocios y deportes. Estas *news* se presentan al usuario de cuatro formas distintas (valores de refuerzo):

- no apropiado por no interesante
- no apropiado por redundante (tiene el mismo contenido que un mensaje anterior)
- apropiado
- muy apropiado

Cada vez que un usuario lee una *new* la califica en alguno de estos valores. A partir de esta información, el sistema mantiene dos perfiles

- un perfil de interés reciente del mensaje basado en los contenidos del mensaje. Si el artículo es muy próximo (con cuidado para que no sea el mismo) se dice que es de interés para el usuario. Si no, se consulta el otro perfil, que vemos a continuación.

- un perfil de interés a largo plazo. Está basado en un clasificador probabilístico que asigna una probabilidad de interés a la noticia basada en la frecuencia de sus palabras en las historias que el usuario ha calificado como interesantes, comparada con la frecuencia de ocurrencias de palabras en las historias que ha calificado como no interesantes.

Aunque el sistema está realizado para mensajes de *news*, es evidente que también puede aplicarse para listas de correo.

Otras técnicas

En [Mobasher et al. 2000] se presenta un sistema para recomendar durante una sesión un conjunto de enlaces a páginas que el usuario podría querer visitar basándose en la similitud de los patrones de uso de la web. Las técnicas empleadas son las reglas de asociación y el agrupamiento. Primero genera grupos de las sesiones basándose en patrones de co-ocurrencia de referencias URL o bien basándose en cuán a menudo las referencias URL ocurren juntas en las sesiones de los usuarios (usando la técnica ARPH, *Association Rule Hypergraph Partitioning* [Han et al. 1997], que funciona eficientemente para conjuntos de datos de alta dimensionalidad). Después, calcula la distancia de una nueva sesión a los grupos para ver cuál es el más cercano y predecir la siguiente página.

21.6 Sistemas de minería de web y textos

Ya hemos visto a lo largo del capítulo que se puede extraer conocimiento de distinto tipo de la web usando herramientas de minería de datos generales. Para ello, es necesario cierto procesamiento previo. Aunque este preprocesamiento se puede realizar manualmente y *ad-hoc* para cada problema, se facilita mucho con herramientas específicas. Además, existen también técnicas específicas que no se encuentran en los sistemas generalistas.

Movidos por las características especiales de los datos en la web, muchas empresas están desarrollando herramientas específicas. Por ejemplo, *SAS*, que ya dispone de la herramienta *Intelligent Miner* para la minería de datos en general, ha desarrollado *SAS Text Miner* (<http://www.sas.com/technologies/analytics/datamining/textminer>), una herramienta para el procesamiento y el análisis de textos escritos en diferentes formatos (pdf, *extended ASCII*, HTML y Microsoft Word), que permite descubrir conceptos contenidos en grandes colecciones de textos, agrupar documentos por tópicos, clasificar documentos en clases predefinidas e integrar datos textuales con datos estructurados.

Algunas otras herramientas comerciales son: el sistema *IBM Intelligent Miner for Text* (<http://www3.ibm.com/software/data/iminer/fortext/index.html>), que permite la extracción de características, el agrupamiento, la categorización, los resúmenes, así como la recuperación de textos completos; *WebDataKit* (<http://www.lotontech.com/wdbc.html>), que permite hacer preguntas SQL a páginas web; *WLS XP* (<http://www.webmining.cl/>) que es una herramienta que descubre patrones de la actividad de los usuarios en un portal web generando reglas de asociación a partir de archivos log; *OK Log* (<http://www.oklog.biz/>), que examina portales web para obtener información de las visitas de los usuarios.

Como herramientas no comerciales citaremos *Analog* (<http://www.analog.cx/>), una herramienta para el análisis de archivos log de servidores web, y *Wum 7.0* (<http://www.iwi>).

hu-berlin.de), un entorno integrado para analizar el comportamiento de navegación de los usuarios y descubrir patrones secuenciales.

Los sistemas referenciados en este apartado son sólo una muestra de las muchas herramientas que para la minería web y textos ya existen en el mercado. Para una lista más completa de sistemas de minería web se puede consultar la dirección <http://www.kdnuggets.com/software/web.html>.

Dado el carácter cambiante de la propia web es de esperar que el área de la minería web evolucione y crezca dando lugar a nuevas técnicas propias así como a nuevos sistemas de minería web.

PARTE VI

IMPLANTACIÓN E IMPACTO

DE LA MINERÍA DE DATOS

En esta parte final se introducen una serie de pasos y recomendaciones para implantar un programa de minería de datos en una organización (recursos humanos y materiales necesarios), pasos, costes, etc., bajo el estándar CRISP-DM, así como unas breves nociones sobre el uso y mal uso de la minería de datos, sus repercusiones y las tendencias futuras de la tecnología de minería de datos.

CAPÍTULOS

- 22. Implantación de un programa de minería de datos**
- 23. Repercusiones y retos de la minería de datos**

Capítulo 22

IMPLANTACIÓN DE UN PROGRAMA DE MINERÍA DE DATOS

La decisión de implantar un programa de minería de datos y el diseño de un plan del mismo deben preceder a cualquiera de las fases que se han visto en capítulos anteriores. De hecho, establecer cuál es el contexto del negocio, los objetivos del mismo y plasmarlos en objetivos de minería de datos, es previo a pararnos a pensar en recopilar y preparar los datos, realizar los modelos, evaluarlos y utilizarlos. Si este libro estuviera dedicado a ser una guía de implantación de minería de datos, empezaríamos justamente sobre cómo diseñar este plan. Sin embargo, es muy difícil realizar un plan de minería de datos (o entender cómo se puede hacer uno) sin conocer la tecnología. Por esta razón, esta primerísima etapa del proceso de extracción de conocimiento la hemos relegado a casi al final de libro.

De alguna manera este capítulo marca la diferencia entre la tecnología y la ingeniería: conocer la tecnología no asegura el éxito si no se sabe cómo aplicarla en un contexto concreto y teniendo en cuenta unas limitaciones de costes, de recursos (tanto materiales como humanos), de plazos y demás aspectos que hacen de la ingeniería un híbrido entre la tecnología y la metodología. Este capítulo desarrolla cierta metodología más general sobre organización, gestión y planificación de proyectos de minería de datos. Quizá, por este motivo, este capítulo es relativamente informal e incluye muchos consejos que cada lector deberá entender en su justa medida y adaptar a sus necesidades.

22.1 Introducción

Llegados a este punto del libro, se puede percibir claramente que los conocimientos acerca de lo que es la extracción de conocimiento comienzan a asentarse. Algún lector incluso puede sentirse tentado a amanecer un lunes por la mañana en el trabajo y decir: “compañeros, hoy empieza una nueva era; desde hoy vamos a empezar un programa de minería de datos en esta organización”, esperando salir aupado en hombros hacia el despacho de dirección por sus compañeros, entre loores, aplausos y aclamaciones.

Evidentemente, no todos tenemos tal ímpetu para empezar una nueva era, y mucho menos los lunes por la mañana. Y si realmente estamos hechos de una pasta especial, es posible que la mayoría de los compañeros sean reacios a la innovación y a dedicar parte de su ya exprimido tiempo a nuevos *planes*, originados e implantados por decreto desde la dirección.

Sea de una manera más impetuosa o más tímida, hoy en día, ninguna empresa u organización de cierto tamaño puede permitirse el lujo de no haber tenido en el orden del día de alguna reunión de dirección el tema: “*Implantación de minería de datos?*”. Es muy posible que la respuesta haya sido que no, o más bien, *de momento* no. Como veremos, lo grave no es responder negativamente a esa pregunta; muchas empresas no necesitan la minería de datos. Lo grave es no haberse hecho todavía la pregunta.

Si realmente se ha hecho dicha pregunta y se ha respondido en alguna ocasión afirmativamente (luego veremos algunos consejos para ayudar a saber responder afirmativa o negativamente a esa pregunta con criterio), debemos empezar algo nuevo que, aunque pueda tener distintos nombres (programa, proyecto, plan o entorno de minería de datos), significa implantar en la organización la tecnología de extracción de conocimiento a partir de datos. Pongámonos en ese caso a la obra.

Un programa de minería de datos es un uso sistemático, prolongado y racional de las técnicas de minería de datos en una determinada organización. En este capítulo nos centramos en concretar aquellos aspectos relevantes para implantar un programa de minería de datos: cuándo es conveniente, con qué grado de autosuficiencia, qué fases y cómo planificarlo, qué integración es necesaria con otros subsistemas de la organización y qué recursos materiales y humanos son necesarios. Lógicamente, nos interesa esclarecer los aspectos anteriores de tal manera que el programa tenga éxito, es decir, se consigan los objetivos (que repercutirán en unos beneficios) con los costes que se habían previsto.

Como aperitivo, pasemos en primer lugar a hacer una breve exposición de las claves del éxito de un programa de minería de datos.

22.1.1 Claves del éxito de un programa de minería de datos

Según numerosos autores, experiencias anteriores y cierta reflexión apoyada en el sentido común, se pueden destacar los siguientes aspectos fundamentales para el éxito de la minería de datos en una organización:

- El negocio⁶⁵ y sus necesidades han de dirigir el desarrollo del programa. Se han de especificar claramente los problemas y objetivos de negocio. Con estos problemas y *objetivos de negocio* podremos averiguar qué datos van a ser necesarios y podrán surgir los objetivos y tareas de minería de datos.
- Una buena especificación de problemas concretos y específicos de minería de datos es otra clave del éxito. Es importante trasladar correctamente los objetivos de negocio a los objetivos concretos de minería de datos. La correspondencia entre ambos determina que estemos realizando tareas de minería de datos que ayuden a resolver necesidades de la organización. La especificación de la calidad de los modelos

⁶⁵ Utilizamos la palabra “negocio” para identificar el ámbito y contexto de la organización. El término se aplica tanto a empresas privadas como a instituciones públicas.

requerida (en términos de precisión, comprensibilidad, fiabilidad...) es también fundamental.

- La integración del resto de programas de la organización y el apoyo incondicional de los altos ejecutivos de la organización es imprescindible. La existencia de programas previos de calidad de datos y de almacenes de datos puede ser muy importante.
- La calidad de datos (ya sea por un programa previo o por limpieza) es primordial. La integración de información externa necesaria para los modelos, como campañas de la competencia, evolución de la economía, calendarios, etc., es, en muchos casos, indispensable.
- El uso de herramientas integradas y de entornos amigables es otro factor destacable. La informatización del proceso a todos los niveles es también muy importante: documentación, comunicación entre el grupo, herramientas de *workflow*, etc.
- La necesidad de un equipo heterogéneo de personal formado no sólo en minería de datos, sino también en estadística, bases de datos y el área de negocio. La continuidad de formación de este personal y de los usuarios es también muy importante. La existencia de un liderazgo efectivo y unificado del equipo es también vital.
- Una evaluación de los modelos más holista (teniendo en cuenta costes, comprensibilidad, relevancia, etc...) y un despliegue de los mismos a todos los niveles (personal de toma de decisiones, resto de usuarios, aplicaciones e incluso la propia base de datos transaccional) permitirá trasladar los resultados de la minería de datos a resultados positivos en el negocio.

Sobre estas claves del éxito vamos a tratar en los siguientes apartados.

Uno de los aspectos a considerar cuando se implantan programas basados en nuevas tecnologías es que, al avanzar en el programa, la tecnología también ha progresado. De hecho, hay que pensar que el programa puede ser revisado por nuevos avances, herramientas, tecnologías o programas de otro cariz, que los engloben o complementen. De hecho, esto es lo que está ocurriendo con muchos programas de almacenes de datos; aún no se han terminado de implantar y ya se está pensando en rediseñarlos para dar cabida a la minería de datos.

22.2 ¿Cuándo empezar? Necesidades y objetivos de negocio

Los consultores, asesores y comerciales *productivos*, así como las revistas y expertos *especializados*, se las ingenian, especialmente en momentos de crisis, de estancamiento o de bonanza, es decir, en cualquier momento, para ofrecernos productos que ya tenemos (con nombres y precios más "modernos") o vendernos productos que no necesitamos, pero que acabamos necesitando. Si recientemente le convencieron de que el futuro está en el *software* de cinco capas de *noodleware*, puede ser que, con tal escarmiento, dude de que realmente la minería de datos no sea más que una moda más, una palabra atractiva que le obligue a actualizarse la versión de su sistema de gestión de base de datos y a comprar los complementos de minería de datos. Quizá tiene este libro entre sus manos para tomar una postura positiva o negativa con conocimiento de causa. Esperamos que mediante este libro

(y posiblemente otras fuentes) haya podido reconocer que dentro de la minería de datos existen no sólo posibilidades muy interesantes sino realidades de implantación y de mejora de procesos y decisiones.

Antes hemos comentado que es imprescindible preguntarse si es rentable para una organización implantar un programa de datos. Lógicamente, si existen beneficios de su implementación, también habrá inconvenientes y costes. Como en la mayoría de decisiones, se trata de sopesar si los beneficios son mayores que los costes. Quizás, en este caso, la pregunta se realiza periódicamente, hasta determinar que se llega a un momento en el que se estima que los beneficios son mayores que los costes.

No obstante, aunque la minería de datos *funciona en general*, no funciona igual de bien en todos los ámbitos y, evidentemente, existirán organizaciones para las que será mayor el esfuerzo que el beneficio obtenido. No hay las mismas posibilidades para una gran empresa de telecomunicaciones, una pequeña empresa de seguros, un equipo de fútbol, un hospital o una pequeña inmobiliaria por Internet, que las que existen para una casa rural no integrada en ninguna red de promoción, una notaría del estado o una gran constructora que subsiste principalmente de los contratos con la administración. En muchos casos no es una cuestión de tamaño o incluso de la rama de negocio, sino más bien dependerá de que se tomen decisiones importantes diariamente sobre un entorno cambiante y que exista una cierta tradición de informatización y de gestión de datos en la organización.

¿Qué necesito saber, por tanto, para tomar la decisión de si debo o no implantar minería de datos? Indudablemente, para tomar una buena decisión debe estudiar el caso en profundidad. Paradójicamente, en muchos casos, esto obliga a diseñar, aunque sea esquemáticamente, el programa de minería de datos, para saber si es factible o no. Es decir, para saber si debo implantar un programa de minería de datos debo, en primer lugar, diseñar y tener entre las manos un programa de minería de datos. Dicho así no suena tan paradójico. Esta situación se muestra en la Figura 22.1.

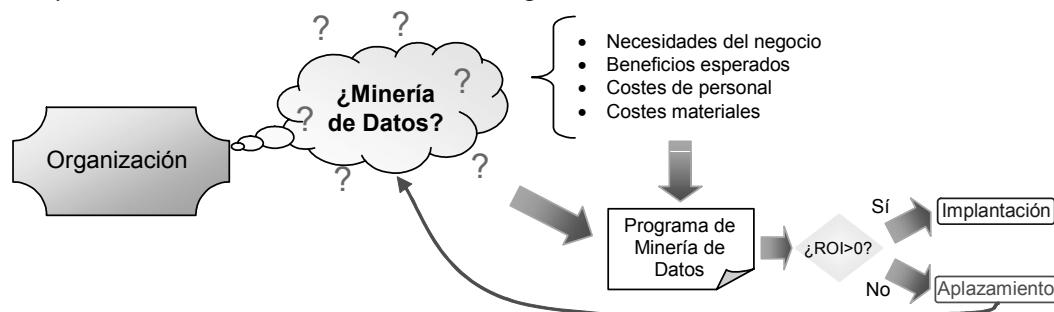


Figura 22.1. La necesidad de un programa de minería de datos sobre la mesa.

Con un (esquema de) programa de minería de datos podemos estimar cuál va a ser el rendimiento o ROI (*Return On Investment*), evaluando los beneficios y evaluando de la manera más ajustada posible los costes, ya que implementar minería de datos puede requerir una inversión considerable en formación, herramientas y personal. Se debe preservar en todo momento la duda sanísima de si, en su caso, la minería de datos le va a proporcionar, en su conjunto, más beneficios que costes.

El concepto de “beneficio” o de “productividad” es muy variable dependiendo de la organización. Por ejemplo, para los Chicago Bulls, la obtención de entre dos y tres puntos

más por partido se considera un éxito y un beneficio sustancial [Dhar & Stein 1997] que justifica el mantenimiento de su programa de minería de datos (que inicialmente se marcó como objetivo). En cambio, para una gran compañía de telecomunicaciones conseguir obtener los mejores destinatarios potenciales para un nuevo producto y aumentar en un 50 por ciento la penetración de su campaña de marketing es también un beneficio significativo. Sea como sea, el beneficio es importante expresarlo en términos económicos, es decir, en euros, dólares o la moneda que se esté utilizando en la organización. Por ejemplo, en el caso de los Chicago Bulls, hay que estimar cuántas victorias más supone una media de dos o tres puntos por partido y cómo se traduce esto económicamente (más títulos, más asistencia a los partidos, más ingresos por publicidad, etc.).

Un problema añadido a la hora de estimar el rendimiento esperado es que el personal que toma las decisiones en la organización es precisamente el que va a tomar el rol central (como promotores y como usuarios) del programa de minería de datos. Van a tener que ser ellos los que determinen si es necesario el programa. Es, muchas veces, una decisión difícil, ya que ellos mismos son sujetos y objetos del programa. En general, este tipo de personal suele tener una visión favorable a todo aquello que facilite su trabajo, pero los directivos suelen ser poco realistas a la hora de evaluar la formación y esfuerzo que ellos mismos van a tener que invertir. Esta subjetividad es similar a la que puede tener un director de recursos humanos cuando se le pregunta si debe autorizar un aumento de sueldo al director de recursos humanos.

Un aspecto a considerar para determinar si es necesario un programa de minería de datos es (como no podría ser de otra manera) la identificación de necesidades que puedan ser cubiertas con la minería de datos. Y es que, en cualquier caso, como bromeábamos con las consultoras, la necesidad debe crear el programa y no el programa debe crear la necesidad. Las necesidades pueden ser clásicas: identificar razones de subidas o bajadas de ventas, adecuar la producción a las ventas futuras, determinar tendencias del mercado o la competencia, realizar una campaña de marketing selectiva de clientes ante cierto producto, reducir las listas de espera de un hospital, reducir el consumo eléctrico de un edificio inteligente, determinar los sectores de más uso de los mensajes de texto, filtrar el correo no solicitado, etc. Estas son las necesidades del negocio y van a marcar los *objetivos de negocio*.

Es decir, los objetivos de negocio de un programa de minería de datos *no* son tareas de minería de datos. Por ejemplo, la asignación de una probabilidad de compra de un producto a un conjunto de clientes es una tarea de minería de datos que, en sí misma, no tiene beneficios. En cambio, la realización de una campaña de marketing selectiva de clientes ante un producto y la obtención de un grado de respuesta de un 5 por ciento (sobre el 2 por ciento que se conseguía por medios tradicionales) es un objetivo de negocio con una medida de evaluación, del cual sí que puede estimarse su beneficio. Dicho de otra manera, el usuario de negocio no quiere crear un patrón predictivo o crear una red neuronal. Si no hay una necesidad de negocio detrás, esto no sirve para nada. El usuario quiere saber a qué segmento puede dirigir cada producto, qué clientes van a ser malos pagadores, conseguir retener a los clientes evitando abandonos o fugas a la competencia (*churn*), etc.

Tener en mente objetivos de negocio y plasmarlos o desglosarlos en tareas de minería de datos puede dar la impresión de que sólo vamos a poder hacer minería de datos *dirigida*, es decir, donde sabemos a priori qué tarea vamos a resolver y qué tipo de modelo vamos a

obtener (por eso se llama también *model-driven*). Si bien, como veremos, este tipo de minería de datos es la ideal para empezar, sí que se puede contemplar también la minería de datos *no dirigida*, donde se parte de los datos (*data-driven*) y se obtengan patrones que, a medida que se van descubriendo, se estime si pueden ayudar a resolver algunas de las necesidades de negocio. Lógicamente, esta manera es más abierta y arriesgada y, en muchos casos, obligará a desechar modelos porque no son relevantes para ningún problema del negocio.

22.2.1 ¿Subcontratar? Grados de autosuficiencia de un programa de minería de datos

Una técnica que es muy útil a la hora de saber si un determinado servicio es necesario para la organización es intentar realizar un contrato virtual, como si realmente se quisiera subcontratar el servicio a otra empresa. Formalizar un contrato tiene muchas ventajas. En un contrato aparecen los objetivos, los servicios prestados de una manera precisa, los costes, los plazos de ejecución, los riesgos, etc. En realidad, la especificación de un programa de minería de datos debería establecerse casi como si fuera un contrato.

En el apartado anterior, hemos comentado algunos aspectos a tener en cuenta para decidir si embarcarse o no en un programa de minería de datos. Hablando de contratos o de subcontratar, resulta que la respuesta a la implantación de un programa de minería de datos no es sólo un sí o un no, o un cuándo, sino, además, un *cómo*. Existen cinco⁶⁶ maneras diferentes de traer la minería de datos a una organización según el nivel de subcontratación del programa:

1. Mediante la compra de las puntuaciones (*scores*) o predicciones: alguna empresa o consultora externa tiene buenos modelos de negocio, obtenidos a partir de grandes bases de datos (de otras organizaciones similares). Le proporcionamos preguntas y ellos nos responden utilizando esos modelos. Por ejemplo, podríamos proporcionarles un listado de clientes y un producto, y la consultora nos puede devolver el listado acompañado de una probabilidad de compra de cada cliente del producto en cuestión (una puntuación, al fin al cabo, de cada cliente). Es decir, todo el análisis es externo; la organización se limita a postular sus preguntas y necesidades y recibir las respuestas por parte de la consultora o la empresa subcontratada, que actúa de experto. Este modelo se utiliza frecuentemente en la detección de fraudes; por ejemplo una tienda envía información a un centro externo que determina si la operación parece correcta o fraudulenta.
2. Mediante la compra de los modelos: muchas organizaciones del mismo ramo llegan a conclusiones parecidas sobre su negocio, ya sea porque las extraen de sus datos y de su experiencia, porque las comentan en foros o porque las leen en las revistas o boletines de divulgación del ramo. Existen modelos obtenidos por algunas organizaciones que van a diferir poco de los modelos que se pueden obtener en otra. Parece lógico que si alguien vende los modelos ya hechos (por ejemplo en un lenguaje estándar de intercambio de modelos, como el PMML, visto en la Sección 19.4) se va a ahorrar tiempo y dinero. Además, parece que va a ser mucho más eficiente tener

⁶⁶ Este número depende de la forma de agruparlo. Por ejemplo, para [Berry & Linoff 2000] existen cuatro maneras.

el modelo y usarlo con los propios datos, que tener que enviar los datos a que se puntúen o categoricen, como en el caso anterior. Y, lo que es más importante, tener el modelo, especialmente si es comprensible, aporta mucho más conocimiento ('know-how' en términos de consultora) que el caso anterior y, en algunos casos, permite cierta adaptabilidad. Por ejemplo, podemos comprar un modelo de fuga de los clientes y utilizarlo cuando deseemos. Los modelos se pueden comprar con mantenimiento, en paquetes y con actualizaciones periódicas, con lo que se evita el problema de que se queden obsoletos. Los sistemas expertos o las bases de conocimiento que se compran a entidades externas siguen esta filosofía.

3. Mediante la compra de *software* específico: se trata de aplicaciones particulares del negocio que automatizan ciertos procesos de minería de datos en ese contexto. Normalmente son herramientas de una parte del negocio específica, por ejemplo CRM (*Customer Relationship Management*), ERP (*Enterprise Resource Planning*) o SCM (*Supply Chain Management*). Estas herramientas requieren que los datos se integren en ellas, pero una vez hecho esto, si incluyen técnicas de minería de datos, pueden ser capaces de generar modelos a partir de los datos de la propia organización. La gran ventaja de esta solución sobre las soluciones anteriores es obvia: el modelo es específico de la organización, permite capturar las regularidades y patrones que se producen por las peculiaridades de la organización y de su contexto y, además, es privado a ella. Esta solución puede ser apropiada para empresas de producción o de distribución, ya que tienen problemáticas y herramientas similares. Existen paquetes clásicos de CRM, ERP o SCM que empiezan a incorporar minería de datos.
4. Mediante la subcontratación de consultores o expertos en minería de datos: se encarga a un equipo externo la realización del programa de minería de datos. Esto permite obtener rápidamente un equipo con experiencia, fijar unos plazos cortos para los primeros resultados y dar la flexibilidad y posibilidades que queramos. Es importante asegurarse de que realmente el equipo externo tiene la experiencia en minería de datos que dicen tener, además de un cierto conocimiento del área de negocio de su negociación. En general, hay que desconfiar de los que le ofrezcan imposibles, como, por ejemplo, más de 30 años de experiencia en minería de datos.
5. Mediante el desarrollo de un programa interno: esta opción es la más ambiciosa, y requiere la formación de personal dentro de la organización y la creación de un grupo de trabajo. En esta situación, la organización tiene control total sobre el proceso y además tiene un conocimiento más completo del negocio (a no ser que el equipo formado esté integrado mayoritariamente por recién contratados).

Desde la primera a la quinta solución se aumenta en coste inicial y complejidad pero también aumentan en flexibilidad, privacidad y potencialidad y, a largo plazo, en rentabilidad. En algunas áreas las primeras tres opciones no son posibles, o bien porque existe mucha variabilidad entre unas empresas y otras del mismo ramo (clientes diferentes, zonas diferentes, etc.), o bien porque el negocio avanza muy rápidamente y no hay manera de establecer reglas generales con cierta perdurabilidad.

Además, existe un criticismo importante acerca de la contratación a una firma externa o a una consultora de aspectos de minería de datos. Si la "inteligencia de negocio" se relega a una empresa externa, podemos tener como resultado que el conocimiento extraído y las ventajas competitivas pueden, de alguna manera, ser "reutilizadas" por la empresa

contratada para la competencia (pasando a suministrar puntuaciones o modelos nuestros a otras empresas sin nosotros saberlo). En teoría esto debería impedirse legalmente en los contratos, pero es muy difícil impedir no usar (o “redescubrir”) una regla de “know-how”, simplemente porque se aprendió haciendo minería de datos para otro cliente.

Una táctica bastante inteligente es subcontratar inicialmente y aprender de los consultores para, posteriormente, ir independizándose formando un equipo propio, es decir, ir avanzando en el nivel de autosuficiencia. Esta solución tiene el riesgo de que normalmente la empresa subcontratada intenta seguir siendo necesaria, un fenómeno que es muy usual con las consultoras: cuando resuelven el problema resulta que ha aparecido otro que sólo ellos saben resolver.

Una solución menos arriesgada es pedir asesoramiento externo para la creación del grupo de minería de datos o para el arranque, pero sin que todo sea realizado externamente. Además, en este caso, se puede recurrir a expertos en universidades u otros centros de investigación que, por regla general, darán un servicio más desinteresado y desearán que la organización sea autosuficiente cuanto antes, al contrario de una consultora profesional, que, en cierto modo, basa su negocio en que la organización no sea autosuficiente.

Pese a todas las alternativas anteriores, a continuación, y para el resto del capítulo, asumiremos el caso en el que el programa de minería de datos se implementa internamente, ya que, al ser la opción más compleja, es la que tiene más aspectos que tratar.

22.3 Formulación del programa: fases e implantación

En las secciones anteriores hemos visto que la elaboración de un programa es crucial para tomar una decisión definitiva sobre si hay que embarcarse o no en él. Pasemos, por tanto, a formular dicho programa. ¿Qué debo incluir en el programa? ¿Qué estructura debo seguir? ¿Qué extensión debe tener el programa?

Lógicamente, el programa, y en especial su extensión, será muy diferente si se trata de una gran empresa o si lo desarrollamos en una pequeña empresa. Es importante comprender que mientras una pequeña empresa tiene menos recursos (materiales y personales) para llevar a cabo un programa de minería de datos, como contrapartida también es evidente que la ejecución de un programa de minería de datos en una gran empresa es mucho más complejo que en una pequeña empresa. Ésta es la razón por la cual hoy en día plantearse un programa de minería de datos es igualmente razonable en pequeñas o grandes empresas. Para ambos casos existen soluciones adecuadas.

Sobre los aspectos a tratar en el programa de minería de datos, podemos, en primer lugar, destacar el contexto y las necesidades del negocio, para poder formular los *objetivos de negocio*. A partir de ellos, podríamos destacar las fases que se vieron en el Capítulo 2: recopilación de datos, integración, limpieza, selección, minería de datos, evaluación y uso. Quizás aquí la importancia radica en la formulación de los objetivos de negocio, a partir de los cuales se va dando forma al resto de fases. Evidentemente, cuanto mejor formulados estén los problemas más sencillo será realizar el resto del proceso.

Para formular el programa es muy conveniente establecer no sólo las fases, sino también sus interconexiones y las salidas (el subproducto) de cada fase. Esto puede realizarse utilizando herramientas clásicas de gestión de proyectos o puede llevarse a cabo bajo las nuevas herramientas de “*workflow management*”.

Empezar de cero a establecer estas fases puede ser complicado la primera vez que se realiza un programa de minería de datos. La mayoría de organizaciones desearían tener un patrón o metodología para llevarlo a cabo. Afortunadamente, este patrón existe: se dispone de una metodología estándar para el proceso de minería de datos.

22.3.1 El modelo y guía de referencia CRISP-DM

CRISP-DM (<http://www.crisp-dm.org>) (*CRoss-Industry Standard Process for Data Mining*) es un consorcio de empresas (inicialmente bajo una subvención inicial de la Comisión Europea), incluyendo SPSS, NCR y DaimlerChrysler. La difusión de este estándar ha sido altísima y, al ser independiente de la plataforma o herramienta, está siendo utilizado por cientos de organizaciones en todo el mundo.

El estándar 1.0 incluye un modelo de referencia y una guía para llevar a cabo un proyecto de minería de datos. La guía puede ser muy útil como referencia a la hora de establecer una formulación o planificación de un programa de minería de datos adaptado a las necesidades de cada organización. El modelo y la guía se estructuran en seis fases principales, como se muestra en la Figura 22.2. Como se puede observar⁶⁷, existe realimentación bidireccional entre algunas fases, es decir, algunas fases pueden obligar a revisar parcial o totalmente a fases anteriores.

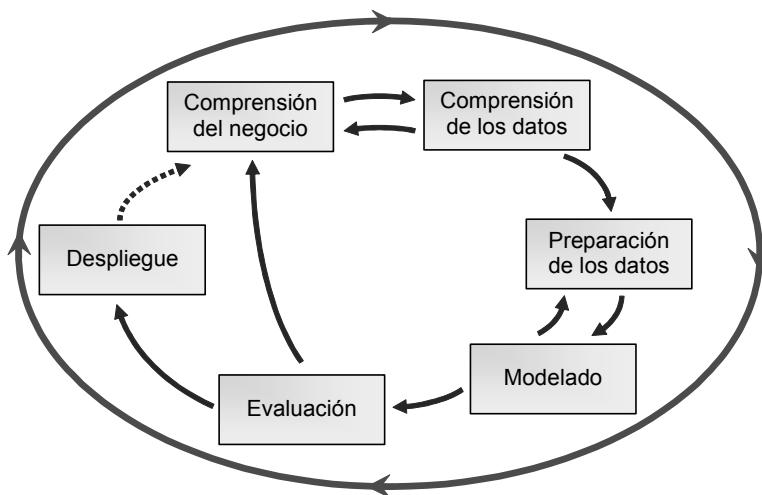


Figura 22.2. Fases del modelo de referencia CRISP-DM.

Pasemos a describir brevemente las fases, las subfases (mostradas en negrita) y los subproductos de cada fase (que se muestran entre paréntesis).

- **Comprensión del negocio:** la primera fase se centra en entender los objetivos y requerimientos del proyecto desde una perspectiva de negocio, plasmando todo esto en una definición del problema de minería de datos y un plan preliminar dise-

⁶⁷ La conexión de la fase de “despliegue” a la “comprensión del negocio” no existe en la propuesta original. No obstante, hemos considerado que en el despliegue de modelos se aprende mucho sobre el negocio y, por tanto, ayuda a la comprensión del mismo. Quizá la representación original quiera mostrar que, aunque el proceso es circular, tal y como se muestra por el círculo que rodea todo el proceso, el inicio de cada ciclo se encuentra en la “comprensión del negocio” y el final del ciclo en el “despliegue”.

ñado a obtener los objetivos. Consta de cuatro subfases: **establecimiento de los objetivos de negocio** (contexto inicial, objetivos y criterios de éxito), **evaluación de la situación** (inventario de recursos, requerimientos, suposiciones y restricciones, riesgos y contingencias, terminología y costes y beneficios), **establecimiento de los objetivos de minería de datos** (objetivos de minería de datos y criterios de éxito) y **generación del plan del proyecto** (plan del proyecto y evaluación inicial de herramientas y técnicas).

- **Comprensión de los datos:** según este estándar, se trata de recopilar y familiarizarse con los datos, identificar los problemas de calidad de datos y ver las primeras potencialidades o subconjuntos de datos que puede ser interesante analizar (para ello es muy importante haber establecido los objetivos de negocio en la fase anterior). La fase de comprensión de los datos consta de cuatro subfases: **recopilación inicial de datos** (informe de recopilación), **descripción de datos** (informe de descripción), **exploración de datos** (informe de exploración) y **verificación de calidad de datos** (información de calidad).
- **Preparación de los datos:** el objetivo de esta fase es obtener la “vista minable”, aunque el estándar no use esta terminología (en realidad se habla de *dataset* y de su descripción). Aquí se incluye la integración, selección, limpieza y transformación. Consta de cinco subfases: **selección de datos** (razones de inclusión / exclusión), **limpieza de datos** (informe de limpieza de datos), **construcción de datos** (atributos derivados, registros generados), **integración de datos** (datos mezclados) y **formateo de datos** (datos reformateados).
- **Modelado:** es la aplicación de técnicas de modelado o de minería de datos propiamente dichas a las vistas minables anteriores. Consta de cuatro subfases: **selección de la técnica de modelado** (técnica de modelado, suposiciones de modelado), **diseño de la evaluación** (diseño del test), **construcción del modelo** (parámetros elegidos, modelos, descripción de los modelos) y **evaluación del modelo** (medidas del modelo, revisión de los parámetros elegidos).
- **Evaluación:** es necesario evaluar los modelos de la fase anterior, pero ya no sólo desde un punto de vista estadístico respecto a los datos, como se realiza en la última subfase de la fase anterior, sino ver si el modelo se ajusta a las necesidades establecidas en la primera fase, es decir, si el modelo nos sirve para responder a algunos de los requerimientos del negocio. Consta de tres subfases: **evaluación de resultados** (evaluación de los resultados de minería de datos, modelos aprobados), **revisar el proceso** (revisión del proceso) y **establecimiento de los siguientes pasos** (lista de posibles acciones, decisión).
- **Despliegue:** se trata de explotar la potencialidad de los modelos, integrarlos en los procesos de toma de decisión de la organización, difundir informes sobre el conocimiento extraído, etc. Consta de cuatro subfases: **planificación del despliegue** (plan del despliegue), **planificación de la monitorización y del mantenimiento** (plan de la monitorización y del despliegue), **generación del informe final** (informe final, presentación final) y **revisión del proyecto** (documentación de la experiencia).

Casi todas estas fases corresponden con fases ya vistas en capítulos anteriores (aunque no exactamente en el mismo orden y agrupadas de la misma manera, en particular las fases de comprensión y preparación de datos). Lo destacable del estándar CRISP-DM es, en primer lugar, el detalle de las fases y la especificación de los subproductos de cada subfase. Muchos de estos subproductos son informes de cada subfase. Este tipo de informes por fases no han sido comentados en capítulos anteriores pero pueden ser muy importantes a la hora de realizar un programa y de realizar su seguimiento.

Pero quizás la fase más novedosa y relevante de las incluidas en el CRISP-DM y que queremos recalcar en este capítulo es la primera, la fase de “comprensión del negocio”. Esta fase no ha sido suficientemente tratada en los capítulos anteriores, así que vamos a detallar sus cuatro subfases aquí:

- **Establecimiento de los objetivos de negocio.** Es fundamental analizar cuáles son los objetivos del negocio, para abordar aquellas cuestiones que son realmente relevantes y beneficiosas para la organización. Si no existe suficiente conocimiento sobre el negocio en el equipo habrá que entrevistar e involucrar a los expertos en el dominio del negocio. Una primera salida de este proceso es un resumen del contexto inicial antes de comenzar el proyecto (que servirá como referencia de partida al finalizar el proyecto). Este contexto debe incluir una descripción de la organización: qué divisiones, departamentos y grupos de trabajo existen (en forma gráfica, si es posible), identificación de las personas clave y de los *instigadores* del proyecto de minería de datos y la motivación del mismo, así como las unidades del negocio que se verán más afectadas por el programa de minería de datos. A partir de este contexto se pueden establecer los objetivos principales. Se generará una lista de objetivos de negocio de la organización (como por ejemplo “reducir el nivel de *stock* de los productos perecederos” o “incrementar las ventas por catálogo de los clientes actuales”) de la manera más precisa posible e intentando establecer objetivos factibles. También es importante esclarecer las interrelaciones entre los objetivos, ya que la satisfacción de algunos objetivos (“incrementar ventas por catálogo” y “reducir costes de marketing”) pueden interactuar (negativamente). Acompañando a estos objetivos de negocio se generará una lista de criterios de éxito. Los criterios pueden ayudar a especificar los objetivos (por ejemplo “reducir la fuga de clientes a un dos por ciento anual”) o pueden ser subjetivos (por ejemplo “proporcionar pautas interesantes sobre los ingresos hospitalarios”), caso en el que hay que especificar quién va a realizar el juicio (por ejemplo “el supervisor de urgencias, el Dr. Martínez, valorará el interés de los modelos sobre pautas de ingresos en una escala de 0 a 10”).
- **Evaluación de la situación.** Esta tarea involucra una búsqueda más detallada de los recursos, restricciones, suposiciones y otros factores al definir el plan del proyecto. La primera salida de esta subfase es un inventario de recursos, incluyendo el personal (expertos en el negocio, expertos en gestión de datos, soporte técnico, expertos en minería de datos), los datos y conocimiento previo existentes (operacionales o en almacén de datos, externos, conocimiento de negocio “know-how” no informatizado, conocimiento de expertos, documentación, etc.), los recursos informáticos (*hardware* y *software*), así como el *software* específico de minería de datos o de análisis. Otra salida de esta subfase es un listado de todos los requerimientos (calidad, seguridad, legalidad), las suposiciones (en especial las no contrastables), las restric-

ciones (disponibilidad de recursos, capacidades de los sistemas, restricciones temporales o económicas), los riesgos o eventos que pueden ocurrir (corte de financiación, reestructuración de productos, adelantamiento sorpresivo de la competencia, mala calidad de datos, etc.) y los planes de contingencia en el caso de que ocurran. Otra salida importante es un glosario de terminología sobre el proyecto, tanto de minería de datos como de la organización. Finalmente, una salida imprescindible es un análisis de costes y beneficios del proyecto, de la manera más específica posible, en términos monetarios (costes de recopilación y preparación de los datos, de personal, de implementación del plan, etc., frente a beneficios del cumplimiento de los objetivos).

- **Establecimiento de los objetivos de minería de datos.** Deben especificarse en términos de minería de datos, y no en términos de negocio. Por ejemplo, un objetivo de negocio podría ser, como hemos visto antes, “reducir el nivel de *stock* de los productos perecederos”, que podría corresponderse con uno o varios objetivos de minería de datos, por ejemplo, “predecir cuántos productos perecederos por categoría va a comprar cada cliente, a partir de las compras de los últimos tres años”. Por tanto, se deberá convertir cada objetivo de negocio en uno o más objetivos de minería de datos. La salida principal de esta subfase es, por tanto, un listado de los objetivos de minería de datos (con su correspondencia con los objetivos de negocio y la manera de utilizar los patrones obtenidos para conseguir los objetivos de negocio). Otra salida son los criterios de éxito para estos objetivos de minería de datos, que ya pueden expresarse en términos de precisión, error cuadrático medio, número de reglas, etc.
- **Generación del plan del proyecto.** Finalmente, una vez establecida la situación, las necesidades, los objetivos de negocio y los objetivos de minería de datos, podemos establecer el plan para obtener los objetivos de minería de datos y, en consecuencia, los objetivos de negocio. El plan de proyecto es la salida principal de esta subfase y debe incluir todas las etapas a desarrollar en el proyecto, junto con su duración, los recursos requeridos, las entradas, las salidas y las dependencias. Como en cualquier proyecto, es importante realizar estimaciones realistas, para lo cual se pueden utilizar técnicas de gestión de proyectos (como por ejemplo las medias entre expectativas pesimistas y optimistas). En cualquier caso, es importante considerar que gran parte del tiempo y esfuerzo (incluso más del 50 por ciento) de los proyectos de minería de datos se suelen dedicar a la parte de preparación de datos. Este plan de proyecto es dinámico y debe actualizarse a medida que avanza el proyecto. En este sentido, el plan de proyecto debe incluir acciones a tomar en caso de situaciones excepcionales y debe recoger puntos de revisión periódicos. Finalmente, en conjunción con este plan, hay que tomar decisiones sobre las herramientas y técnicas a utilizar, cuánto van a costar y cuándo van a estar disponibles.

En cierto modo estas subfases recuerdan las fases de especificación de requerimientos y análisis de cualquier gestión de proyectos. Quizás uno de los aspectos más importantes es la jerarquía de objetivos, es decir, se marcan unos objetivos en términos del negocio y después se plasman o convierten en objetivos, generalmente más concretos, en términos de minería de datos. Por ejemplo, la detección de fraude por teléfonos móviles/celulares es un problema de negocio que debe convertirse en varios problemas de minería de datos tales

como segmentar los clientes y las llamadas, buscar correspondencias entre clientes y llamadas, definir un clasificador de fraude a partir de segmentos de clientes y llamadas anómalas y otros atributos. En algunos casos, los problemas de minería de datos generados son problemas dirigidos (*model-driven*) pero, en otros casos, como hemos comentado anteriormente, pueden ser problemas no dirigidos (*data-driven*), como por ejemplo “estudiar mejor los patrones de las llamadas”.

La especificación del CRISP-DM, en especial la guía, detalla el resto de fases. Como hemos dicho, no es mala opción obtener una copia de la guía (gratuitamente desde la página web del consorcio) y adaptarla a nuestras necesidades. Además de servirnos como una buena referencia, permite que el personal externo se acople más fácilmente (si está familiarizado con las fases y terminología del CRISP-DM) y permite también comparar unos proyectos con otros. Existen otras guías, como por ejemplo la guía SEMMA del SAS Enterprise Miner, pero son menos generales, menos completas o menos independientes del fabricante.

Como hemos visto, el CRISP-DM se basa en las fases de la extracción de conocimiento vistas en el Capítulo 2, englobándolas con una serie de subfases previas de “comprensión del negocio” y unas subfases finales de evaluación del proyecto. No obstante, no trata algunos aspectos importantes, como son la organización y características del personal, el material necesario, los plazos o la ciclicidad del programa. Estos aspectos fundamentales en la implantación de un programa de minería de datos los trataremos en puntos siguientes.

22.3.2 Implantación progresiva. Ciclo de vida en espiral

El primer programa de minería de datos no ha de contemplar todos los aspectos mejorables en la gestión de la organización. En primer lugar, porque sería imposible contemplarlos todos y, en segundo lugar, porque haría inviable el proyecto. La alternativa consiste en elegir aquellos aspectos o necesidades más claros y relevantes y, posteriormente, una vez realizados estos objetivos, plantearse otros.

Por tanto, la primera implantación debe marcarse unos problemas concretos y que tengan unos beneficios manifiestos. Las dificultades de estos primeros pasos sólo serán compensadas si los primeros problemas elegidos son claros, sencillos de determinar y, por tanto, de resolver con una formación inicial y unas herramientas más sencillas. Esta primera “ronda” puede constituir lo que vulgarmente se denomina proyecto “piloto”. A medida que el programa avance se pueden ir recogiendo problemas más ambiciosos. Esta aproximación cíclica es similar al ciclo de vida en espiral de la ingeniería del *software*.

Como se muestra en la Figura 22.3 el primer ciclo completo engloba las fases de identificación de problemas del negocio, la planificación y organización, identificando los problemas de minería de datos, las fases de extracción de conocimiento, de difusión, despliegue y explotación de modelos y, finalmente, la evaluación de resultados, midiendo los costes y beneficios. Este ciclo corresponde con un miniproyecto piloto de minería de datos, donde se recomienda que las tareas de minería de datos sean principalmente dirigidas. Del primer proyecto y, especialmente, de lo aprendido (de la evaluación de costes, beneficios obtenidos, etc.), nos podemos plantear objetivos más ambiciosos y, en sucesivos ciclos, podemos plantearnos minería de datos no dirigida.

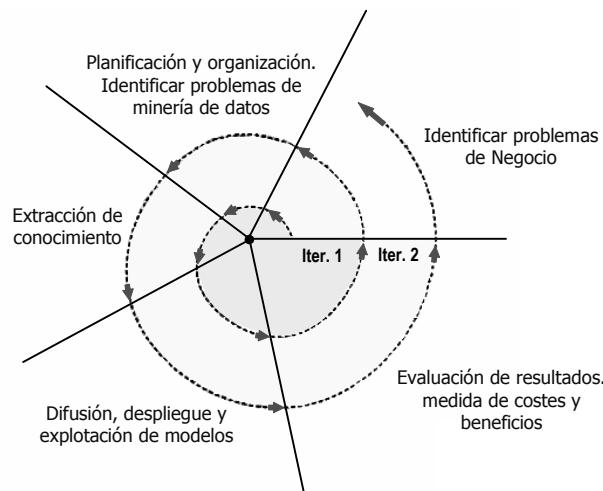


Figura 22.3. Progreso en Espiral de las acciones de minería de datos en una organización.

Con este ciclo en espiral, la pregunta de cuánto tiempo necesito para implantar un programa de minería de datos es baldía. A medida que va avanzando el tiempo se van planteando objetivos más generosos. Aún así, subsiste la pregunta de ¿cuánto debe durar el primer ciclo? La respuesta es difícil de responder, pero debería durar entre unas semanas a unos meses. En general, no deberíamos dejar esperar más de seis meses para tener los primeros resultados, aunque fueran simplemente los más sencillos. Ser capaz de dar una vuelta completa al ciclo anterior es quizás la garantía de que vamos a poder continuar con las siguientes vueltas. También el primer ciclo piloto representa, en cierto modo, un experimento controlado (con beneficios y costes pequeños), a partir del cual se puede tomar la decisión de ampliar el programa de una manera más ambiciosa.

22.4 Integración con las herramientas y proyectos de la organización

Un aspecto crucial para el éxito de un programa de minería de datos, como hemos comentado al principio, es que se integre con el funcionamiento operacional y estratégico (es decir, de toma de decisiones) de la organización, tanto desde el punto de vista de las herramientas, como el de otros proyectos o programas relacionados.

Si nos referimos al organigrama de una organización, el lugar más idóneo para realizar esta integración son los departamentos de logística y prospectiva, de investigación operativa, o similares y en estrecha colaboración con el departamento de informática y con la dirección.

No obstante, si nos referimos al *proceso*, la integración debe realizarse durante todo el proceso de minería de datos pero son las *entradas* (los datos y conocimiento previo) y las *salidas* (los modelos y patrones), los que deben acoplarse más eficazmente.

22.4.1 Integración con las entradas

Respecto a las entradas, las herramientas de minería de datos han de ser capaces de acceder directamente a las fuentes de datos de la organización. La situación ideal es la existencia de

un almacén de datos, como comentamos en el Capítulo 3, y la conexión directa de las herramientas de minería de datos a dicho almacén.

Muchos entornos y herramientas de minería de datos trabajan con archivos planos, es decir, archivos de texto con campos separados por comas, espacios u otros separadores. En realidad, ésta puede ser una opción para experimentar o para un primer programa piloto. No obstante, un programa a largo plazo de minería de datos debe intentar realizar un almacén de datos y permitir que la herramienta de minería de datos se conecte, aunque sea a través de un relativamente lento ODBC, con la base de datos.

A la hora de implantar un programa de minería de datos pueden existir, en relación a los datos, dos programas estrechamente relacionados: el programa de calidad de datos y el programa de almacenes de datos.

Respecto al programa de calidad de datos, si éste ya existe, va a facilitar mucho la tarea, ya que un gran esfuerzo de la minería de datos se centra en la limpieza de datos. No obstante, es importante evaluar si los objetivos de calidad de datos marcados en su momento son adecuados para la minería de datos. Por ejemplo, el programa de calidad de datos puede centrarse en los datos operacionales o sólo en una parte de ellos, por ejemplo en los aspectos más críticos (económicos, facturación, stocks, etc.), pero olvidar otros datos que pueden ser muy importantes para la minería de datos (atributos de los clientes, visitas a la web, etc.). En algunos casos será necesario revisar el programa de calidad de datos y, si existen equipos o grupos de trabajo de los dos programas, hacer reuniones periódicas. Dentro de este programa de calidad de datos se puede también ampliar la captación de información en las bases de datos operacionales (y después en las estratégicas) que sea relevante para la minería de datos (por ejemplo la edad de los clientes, el número de tonos antes de responder una llamada, etc.) que hasta el momento no se registraba porque no se consideraba importante.

En relación con el programa de almacenes de datos, como ya hemos dicho, la minería de datos se verá beneficiada. No obstante, hay que tener en cuenta que el almacén de datos puede estar diseñado con unos objetivos, mientras que, en la nueva situación, se le exigirán otros nuevos. La consecuencia más frecuente es el rediseño del almacén de datos, teniendo en cuenta que también va a ser usado para la minería de datos. La razón de esta inadecuación se debe generalmente a que el almacén de datos no registra la información necesaria, ya sea de fuentes externas o internas, falta de atributos relevantes o demasiado nivel de agregación. Por ejemplo, es muy típica la situación de un almacén de datos que se realiza pensando en un nivel de agregación alto para las consultas y análisis OLAP cotidianos, agregando por ejemplo las compras por horas o por días. En cambio, para realizar las tareas de minería de datos (por ejemplo asociaciones en la cesta de la compra), nos interesan las compras individuales y no agregadas por horas o por días. Esto demuestra que los requerimientos de un programa de almacenes de datos son diferentes si se tiene en mente o no un programa paralelo o conjunto de minería de datos. Esto es especialmente patente en el nivel de agregación, ya que el nivel de agregación de la minería de datos suele ser más fino.

El conocimiento previo de la organización, ya sea en forma de reglas, de procesos, de “know-how”, de estudios previos, estadísticas, etc., debe integrarse y formalizarse, con el objetivo de no volver a descubrir lo que ya se sabe y para asistir en la determinación de objetivos de negocio y objetivos de minería de datos. Intentar informatizar toda esta

información en un banco de reglas puede parecer complejo y hasta utópico, pero quizá sea la mejor opción. Si no es posible esta recopilación de conocimiento previo de manera informática y operativa, al menos en forma de informe o documentación.

22.4.2 Integración con las salidas

La integración de las salidas de la minería de datos es uno de los aspectos en los que se están haciendo más avances recientemente. Como vimos en el Capítulo 19, los modelos extraídos de la minería de datos deben aplicarse, utilizarse para tomar decisiones y difundirse en las áreas de la organización donde puedan ser útiles. Para ello, lógicamente, es necesario integrarse con un conjunto de técnicas y de programas que pueden existir ya en la organización.

Antes de la llegada de la minería de datos (y de los almacenes de datos), en una misma organización pueden combinarse sistemas de informes tradicionales basados en SQL (*reporting tools*), sistemas propios de ayuda a la toma de decisiones (DSS, *Decision Support Systems*), los clásicos MIS (*Management Information Systems*) o los EIS (*Executive Information Systems*), así como los sistemas específicos de partes del negocio: sistemas CRM (*Customer Relationship Management*), sistemas ERP (*Enterprise Resource Planning*) y sistemas SCM (*Supply Chain Management*⁶⁸). Existen muchas otras siglas, pero para hacerse una idea, puede ser suficiente. Todos estos sistemas se llaman informalmente (e impropriamente quizás), “inteligencia de negocio” y, en muchos casos, suelen ir asociados con los sistemas de gestión de bases de datos o con la gestión de la información desde un punto de vista más integral.

Quizá sean el CRM, ERP y el SCM los tres subsistemas más clásicos en una empresa mediana o grande con los que deben integrarse las herramientas de minería de datos. Son generalmente una mezcla de sistemas *operacionales* y sistemas *estratégicos*, ya que continuamente están tomando decisiones (sobre campañas, precios, stocks, pedidos, etc.). Cuando hablamos de las salidas, nos interesa precisamente averiguar cómo la minería de datos puede afectar en estas decisiones. Veamos brevemente estas interrelaciones:

- CRM: es el sistema que estudia e interactúa con los clientes para incrementar los beneficios. Ejemplos de subobjetivos de este sistema son la satisfacción de los clientes, evitar la fuga de clientes (*churn*) o la saturación, diseñar campañas específicas a segmentos de clientes específicos, etc. Para ello hay que hacer las ofertas adecuadas, a los clientes adecuados, en el momento adecuado y a través del canal adecuado. Lógicamente, casi todas estas acciones pueden mejorarse mediante minería de datos. La minería de datos puede actuar en todas las fases de la gestión del cliente: captación, ventas, atención, mantenimiento, seguimiento... para intentar ofrecer lo que el cliente pueda necesitar. Para ello hay que modelar el comportamiento del cliente, en todos los aspectos en los que sea posible. Los modelos de minería de datos han de permitir diseñar campañas de *mailings* (como las vistos en el Capítulo 19), proponer ofertas individuales, perdonar penalizaciones, realizar descuentos, etc. La integración de los modelos extraídos para operar en el propio sistema CRM

⁶⁸ También llamado *Supplier Relationship Management*.

debería estar automatizada, con el objetivo de que estas acciones no tengan que hacerse manualmente o fuera del sistema CRM.

- ERP: es el sistema que asiste en la producción y es fundamental a la hora de gestionar *stocks*, cadenas de montaje, necesidad de piezas, etc. En este sistema se realizan decisiones que se pueden mejorar con herramientas de minería de datos: diseño de procesos, de mezclas, reserva de máquinas, optimización de *stocks*, etc. De modo similar a lo comentado para el CRM, la integración puede ser fundamental para que ciertas acciones no se tomen *fuera* del ERP, sino integradas en él.
- SCM: es el sistema de interacción con los proveedores para maximizar los beneficios, bajando precios y ajustando plazos. Se trata, en este caso, de conseguir mejores precios y servicios de los proveedores. Muchas veces el ERP y el SCM suelen estar integrados, aunque, en el área de distribución, esta diferenciación ayuda a realizar una gestión más agresiva o ajustada de los proveedores.

El CRM es quizás el que más aplicaciones directas tenga, aunque los otros dos sistemas pueden beneficiarse de la minería de datos.

Los otros subsistemas que puede subsistir en una organización son los de más alto nivel: los EIS (Sistemas de Información para la Dirección) y los DSS (Sistemas de Soporte a la Toma de Decisiones). Estos sistemas se centran en el “qué está pasando” y “cómo reaccionar”. Muchos EIS y entornos de soporte a la decisión empiezan a realizar tareas de detección de situaciones anómalas o especiales y avisar mediante alertas a los ejecutivos o los departamentos involucrados. Este tipo de alarmas se desatan generalmente a partir de hechos o de valores agregados relativamente simples. A medida que las consultas que se pueden realizar son más sofisticadas se pueden realizar detectores de situaciones que dependen de condiciones más complejas. Un paso más allá es realizar alarmas que salten cuando ciertos patrones se cumplan o no, utilizando para ello técnicas de minería de datos.

Uno de los problemas de los EIS es que son costosos de mantener y, en muchos casos, no evolucionan al mismo ritmo que lo hace el negocio, ya que no van adaptando las consultas e informes periódicos. Es una de las razones de su cierto declive y su sustitución por sistemas OLAP sobre almacenes de datos.

Es importante destacar que, ya sea para CRM, ERP, SCM, EIS, DSS o cualquier otro sistema, los resultados de la minería de datos son (o deberían ser) novedosos, es decir, la organización no conoce esas reglas de negocio antes de la minería de datos. Por tanto, esas reglas obligan a *revisar* los procesos, adecuándolos a este nuevo conocimiento. Como consecuencia, la integración de la minería de datos en la organización obliga a revisar procesos y, lógicamente, las aplicaciones informáticas de los sistemas anteriores.

Si estas aplicaciones están implementadas de una manera rígida y poco modular, la modificación de código será más compleja y la organización no podrá seguir el ritmo de los descubrimientos obtenidos. En cambio, si estas aplicaciones son capaces de funcionar a partir de un conjunto de reglas (ya estén en una base de conocimiento o en forma de *triggers*) estas reglas se podrán modificar, añadir, borrar o sustituir sin reprogramar las aplicaciones. De hecho, aunque con poca penetración todavía en el mundo empresarial, se considera que lenguajes o sistemas basados en reglas (tradicionalmente llamados sistemas expertos o bases de conocimiento) permiten esta actualización de reglas de negocio de una

manera mucho más sencilla que las aplicaciones tradicionales, y es una opción muy interesante para la codificación del “know-how” de la organización.

Una manera de hacer estos sistemas basados en reglas utilizando tecnología convencional (sin implantar un sistema experto, por ejemplo) es mediante el uso de reglas de actividad, más comúnmente llamadas *triggers* (disparadores). La mayoría de sistemas de gestión de bases de datos disponen de *triggers*, que pueden utilizarse para codificar reglas de negocio provenientes de la minería de datos y que actúen de manera reactiva hacia datos o cambios. Los *triggers*, además, permiten realizar avisos, *mailings* internos y externos, actualizaciones en la base de datos transaccional o en el almacén de datos, con lo que pueden ser una buena vía de difusión y aplicación de los modelos.

Para ello, no obstante, es necesario que los modelos puedan ser integrados en el sistema de gestión de bases de datos. Por ejemplo, el estándar SQL/MM, el “MS OLE DB for data mining” y la aproximación JSR de JDM (*Java Data Mining*) de Oracle (véase el Capítulo 15), permiten que los modelos permanezcan como una especie de “filtros” que pueden aplicarse en consultas SQL. Otra manera de hacerlo es mediante ejecutables externos, ya que la mayoría de sistemas de *triggers* permiten ejecutar código dentro de ellos. Es, por tanto, importante que las herramientas de minería de datos permitan exportar los modelos en forma de código fuente. Esto permitiría invocar este tipo de módulos o componentes de modelos de minería de datos desde los *triggers*.

Otras perspectivas más flexibles sobre el mismo concepto se basan en utilizar estándares de intercambio de datos o de objetos (como DCOM o CORBA). Quizá la tendencia evoluciona a realizarlo con protocolos basados en XML, como el SOAP (*Simple Object Access Protocol*), de carácter general, o más específicos, como la iniciativa PMML, que vimos en el Capítulo 19. Este tipo de estándares permiten no sólo leer e intercambiar los modelos (convirtiéndolos al lenguaje de la aplicación final), sino también invocarlos remotamente o localmente, utilizando un motor de aplicación de modelos XML. De hecho, muchos sistemas de minería de datos están comenzando a incorporar el estándar PMML.

Finalmente, además de todos los sistemas comentados, los modelos de minería de datos pueden integrarse en herramientas de simulación. Este tipo de herramientas son muy interesantes de cara a evaluar los modelos desde el punto de vista de los *beneficios*, teniendo en cuenta costes y realizando pruebas de diferentes decisiones.

22.5 Recursos necesarios

Para llevar a cabo un programa es necesario dedicar unos recursos, tanto materiales como humanos. Estimar bien estos recursos es crucial, en primer lugar, para asegurar el éxito del programa y, en segundo lugar, para estimar bien el coste del programa, ya que son los recursos materiales y humanos las dos partidas más importantes (si no las únicas) del coste de un programa de minería de datos. Veamos estos recursos.

22.5.1 Material necesario

Uno de los errores típicos que se citan en la implantación de un programa de minería de datos es obsesionarse por las herramientas. Para los recién llegados a la minería de datos existe una cierta fascinación por las técnicas incluidas en estos paquetes y los ejemplos

sensacionales que proporcionan dichas herramientas. En las herramientas, existen factores tan o más importantes que la variedad de técnicas: la facilidad de uso, el precio, la integración con los datos, la eficiencia, etc. Pero quizás lo más importante para un programa de minería de datos es que la herramienta dé solución a los problemas de minería de datos establecidos en el análisis y que se derivan de los problemas de negocio.

Los vendedores de herramientas comerciales han sabido reconocer que su cliente es la empresa, no es el estadístico, ni el informático. Por eso, mientras al principio únicamente enumeraban las técnicas incluidas y capacidades técnicas, hoy en día enfocan las capacidades y los ejemplos de la herramienta hacia las necesidades de las organizaciones y las presentan en términos empresariales.

Como vimos en el Capítulo 5, existen varios tipos de herramientas *software* de minería de datos: lenguajes, primitivas y sistemas integrados. Quizás, para iniciar un programa de minería de datos, lo más adecuado sea un sistema integrado que permita experimentar con distintas técnicas, que sea fácil de manejar y que sea económico. Son especialmente interesantes los sistemas que permitan simplemente elegir entre "más precisión / más tiempo" y "menos precisión / menos tiempo" y que no ofrezcan inicialmente (o al menos no estén a la vista) tanta variedad de parámetros a ajustar como otros entornos de minería de datos más sofisticados, que suelen dificultar los comienzos. A medida que avance el programa se puede indagar en estos parámetros o adquirir herramientas más específicas para resolver problemas concretos de gran interés, en los que las técnicas genéricas estén obteniendo resultados más bien limitados.

En el Apéndice A se analizan los sistemas de minería de datos más difundidos, comerciales, como SPSS Clementine, IBM Intelligent Miner, DBMiner, SAS Enterprise Miner, SGI Mineset, Oracle Data Mining Suite (Darwin), o gratuitos, como WEKA, Kepler, Xelopes, MLC++... También se comentan más brevemente algunos sistemas específicos, como por ejemplo C5.0, CART, Autoclass o Neuroshell.

La existencia de sistemas gratuitos es muy sugerente, sobre todo cuando se trata de elegir el primer sistema. Un sistema gratuito puede permitir comenzar con un coste material muy bajo (limitándose al *hardware*) y puede ser ideal para pequeñas o medianas empresas, instituciones públicas comprometidas con el *software* libre, particulares, etc. Algunos sistemas gratuitos, como por ejemplo WEKA, son suficientemente completos (es, de hecho, el que tiene, con diferencia, la mayor variedad de métodos) para poder comenzar y avanzar en el programa de minería de datos, antes de que llegue cierta necesidad de buscar otras herramientas más potentes. Quizás en la facilidad de manejo y en la integración con otras herramientas es donde las herramientas gratuitas todavía sean más limitadas.

En el caso de decidirse por los comerciales, hay que intentar contactar con ellos previamente para ver su disponibilidad (especialmente la implantación que tienen en cada país y cada zona), los cursos que ofrecen y su trato, intentar obtener *demos* (aunque sean de prueba temporal), averiguar si se ha usado el mismo sistema en situaciones similares y si pueden aportar experiencia sobre el negocio en cuestión. Es decir, no sólo hay que realizar una visita a la web de cada vendedor, hay que intentar conocer quién hay detrás, ya que se están pagando unos productos que, por lo general, son bastante costosos. Además, es importante, contactar con varios vendedores, ya que algunos pueden ayudarnos a destacar las limitaciones de los otros.

22.5.2 Personal necesario y formación

Si bien es cierto que, a diferencia de unos años atrás, se puede realizar minería de datos con unos conocimientos más superficiales de estadística y la ausencia de una base matemática fuerte, también es cierto que la aparición de nuevas técnicas del aprendizaje automático, de la integración con las bases de datos y las nuevas aplicaciones, hacen que se requiera un personal bien formado en el campo de la minería de datos.

La minería de datos está llegando a un nivel importante de especialización tecnológica y, por tanto, no hay vuelta de hoja; hacen falta ciertas capacidades para poder afrontar un programa de este estilo. Estas capacidades fundamentales son: conocimiento estadístico y de técnicas de aprendizaje automático, conocimiento de bases de datos relacionales y técnicas OLAP y conocimiento del negocio. Casi en consonancia con estas capacidades aparecen tres perfiles de profesional indicado para la minería de datos:

- **Informático:** proporciona las capacidades técnicas en tecnologías de la información y, posiblemente, en aprendizaje automático. Se centrará fundamentalmente en utilizar las mejores técnicas, las más eficientes y permitirá integrarlo todo de manera fluida con las fuentes de datos. Permitirá la automatización de procesos, podrá extender o particularizar las herramientas fácilmente, y facilitar la integración con otras herramientas y sistemas informáticos de la organización.
- **Estadístico:** proporciona rigor y conocimiento extenso sobre las tareas de limpieza y selección de datos, el análisis exploratorio de los datos y otras técnicas estadísticas, pero, muy especialmente, sobre la validación de los modelos.
- **Empresarial:** proporciona el conocimiento sobre el negocio. Conoce el funcionamiento de la toma de decisiones de dirección y los problemas y planes estratégicos. Es importante que sean expertos en el dominio o en la organización. Por tanto, la contratación *desde fuera* de profesionales de carácter empresarial o industrial para el programa de minería de datos suele tener malos resultados. El personal con este perfil debe provenir del área de la decisión de la propia organización (o al menos de la rama de negocio de organizaciones similares) y debe tener experiencia en la propia organización.

Cada uno de ellos tiene sus ventajas e inconvenientes. Los de perfil técnico (informáticos y estadísticos) generalmente pueden acabar resolviendo problemas que no son interesantes o resolviéndolos de una manera que no es adecuada desde un punto de vista empresarial [Dhar & Stein 1997]. Los informáticos suelen ser menos prudentes, en cuestión de validación y uso de los modelos, que los estadísticos y empresariales. Los de perfil empresarial no suelen ser realistas en las dificultades o costes de informatización de algunos procesos y suelen infravalorar las dificultades de la minería de datos y engrandecer sus posibilidades.

Debido a las limitaciones de cada perfil, si se va a organizar un grupo/equipo de personas de minería de datos, es preferible contratar un grupo variado, que incluya componentes de los tres perfiles y que trabajen en un nivel de integración alto. No obstante, en cualquier caso, los que conocen el negocio deben conocer un mínimo de la tecnología y los técnicos deben conocer un mínimo del negocio. Además, los directivos por encima de este grupo (si no están integrados directamente en el grupo) también deben conocer un

mínimo de la tecnología y estar al corriente, mediante informes o reuniones periódicas, del desarrollo del programa y de lo realizado por el grupo.

Si, en cambio, en una pequeña empresa u organización, el programa de minería de datos lo va a llevar adelante una persona únicamente, la decisión sobre el profesional adecuado es más difícil. Si opta por un informático, debe conocer los sistemas de negocio y tener una cierta base estadística (los planes de estudios informáticos suelen incluir materias sobre estadística). Si es un estadístico, debe también conocer el negocio y tener una base mínima en tecnologías de la información. Quizá cuando se trata de elegir una persona únicamente, la opción del perfil empresarial tiene mayores limitaciones.

Existen algunas situaciones que pueden facilitar la decisión. Si la misma persona se va a encargar de los programas de almacenes de datos y minería de datos, escoja un informático. Si piensa utilizar herramientas gratuitas que hagan necesario *trastear* con ellas, es preferible también un profesional con base informática. También puede ocurrir que ni siquiera pueda dedicar una persona a tiempo completo al programa. Si va a dedicar a tiempo parcial a una sola persona, probablemente un informático será más polivalente, con cierto seguimiento por parte del gerente o directivo de la organización.

En el caso de haber decidido que hace falta más de una persona, es necesario decidir cuántas. El tamaño ideal del grupo es muy difícil de estimar, como lo suele ser en cualquier otro tipo de proyecto. Para hacerse una idea, incluso para grandes multinacionales con larga tradición en minería de datos, puede ser suficiente con una docena. Por ejemplo, el grupo de minería de datos de BT (antes British Telecom), fundando en 1994, contaba cuatro años después [Roberts 1998], con doce integrantes experimentados en áreas del aprendizaje automático, estadística, visualización y bases de datos. No obstante, hoy en día ese equipo es mucho mayor y se encuentra distribuido por divisiones, actuando el núcleo inicial, en cierta forma, como asesores de los equipos que se van creando en cada departamento o división de negocio.

Lógicamente, pocos lectores podrán aspirar a proyectos del calibre de un magnate de las telecomunicaciones y pensarán que sus equipos serán mucho más reducidos. Aunque existen autores que suelen cifrar el número de componentes del equipo entre ocho y 12 personas [Berry & Linoff 2000], el tamaño dependerá de la organización. En muchas de las aplicaciones en pequeñas y medianas empresas (e incluso grandes empresas), este número será mucho menor. Quizá, según nuestra opinión, el tamaño de entre ocho y 12 personas hay que tomarlo como un máximo, ya que el equipo funcionará coherentemente entre una y 12 personas, pero a partir de ese número será muy difícil de coordinar. Un número mayor requerirá una especialización o una jerarquía que puede resultar poco natural.

Finalmente, no hay que escatimar sobre la satisfacción y cuidado de los componentes del grupo. Están tratando con información de alta importancia para la organización y el éxito del programa puede proporcionar grandes beneficios. Mantener parte o la totalidad del equipo en una situación precaria laboralmente (contratos pocos estables, sueldos escasos), puede motivar la marcha de algunos de sus miembros, lo que puede ralentizar o paralizar el proyecto y, lo que es más grave, filtrar a la competencia el conocimiento extraído hasta el momento.

Decidida la formación del equipo, esta nueva constitución involucra, generalmente, formación. No obstante, incluso para grupos consolidados la formación debe ser una tarea

constante, debido a la evolución rápida del campo y de las técnicas. Aunque la autoformación, mediante libros, manuales, artículos, asistencia a conferencias, etc., suele ser más económica y efectiva, en algunos casos es conveniente organizar cursos (ya sea dentro de la organización o fuera). En general puede ser interesante formar al equipo en la herramienta determinada que se va a utilizar, aunque en primer lugar, deberían tener una visión más global del área de extracción de conocimiento a partir de datos.

La formación del equipo en la tecnología de minería de datos debe complementarse con la consulta de ejemplos de programas de minería de datos *similares* (de la misma rama de negocio). Para ello, se pueden consultar libros que contienen numerosos ejemplos: en CRM y marketing [Berry & Linoff 2000], en telecomunicaciones [Mattison 1997], en aplicaciones de ingeniería y científicas [Grossman et al. 2001], medicina [Krzysztof 2001; IBM 2001] o de muchos otros tipos (finanzas, gubernamentales, seguros, etc.) (véase por ejemplo [Klösgen & Zytkow 2002]). En general, las empresas no suelen estar muy dispuestas a ventilar públicamente sus programas, con lo que el uso de referencias indirectas (libros) es la opción más sencilla.

Además del equipo de implantación del programa de minería de datos, existe personal dentro de la organización que también puede requerir formación. En primer lugar, existen usuarios que no están en el equipo de implantación, por ejemplo los usuarios de los modelos o los usuarios que pueden realizar alguna tarea simple de minería de datos sin haber participado en el programa. Aunque los usuarios de minería de datos no llegan al volumen de los usuarios de un almacén de datos, pueden existir del orden de diez veces más usuarios de minería de datos que miembros en el equipo que han desarrollado el programa. Frecuentemente se realizan aplicaciones propias que restringen las tareas o modelos que estos usuarios pueden utilizar (por ejemplo para diseñar una campaña de marketing o un *mailing*). Sea como sea, estos usuarios también hay que formarlos, tanto en las aplicaciones propias como en una base mínima sobre el programa de minería de datos de la organización.

No sólo es importante la formación de los usuarios directamente involucrados, sino la información de éstos y del resto. De hecho, uno de los problemas que dificultan el éxito de la implantación de la minería de datos es que los resultados de la minería no se comunican de una manera eficaz dentro de la organización. Esto incluye los usuarios de la minería de datos, pero también cualquier otro personal de la organización al que el conocimiento extraído pueda ser útil o pertinente. Quizás un poco de publicidad del programa de minería de datos, centrándose en los objetivos más que en la tecnología, puede ser suficiente. Como consecuencia, además, puede ser sorprendente el hecho que se observa frecuentemente al poner en marcha un programa y difundir cierto conocimiento de qué se va a hacer y cuáles son objetivos. Esta conciencia colectiva de que se han de mejorar ciertos aspectos es suficiente, a veces, para obtener mejoras en el funcionamiento de la organización, debido al hecho de sistematizar y documentar ciertos procesos, necesidades y objetivos para la minería de datos. Este “efecto placebo” ha de tenerse en cuenta a la hora de evaluar el éxito de un programa de minería de datos.

Para acabar este capítulo, hemos de destacar que, además de la aplicación a empresas y a instituciones públicas, existe un nicho donde la minería de datos empieza a utilizarse cada día más y que no se ajusta exactamente a los parámetros que hemos tratado en el capítulo. Se trata de la minería de datos a nivel personal. Posiblemente un autónomo que

desea utilizar minería de datos o un particular simplemente quiere aplicarlo a sus datos personales. Lógicamente, lo que hemos discutido en las secciones anteriores puede ser útil pero, en cierta medida, le vendrá grande. En general, para el uso particular, como por ejemplo, el filtrado de correo, o el análisis de sus datos económicos, o de sus viajes, etc., la formación es igualmente importante, en particular porque habrá que aportar más imaginación a la limitación de recursos (y de datos externos).

Capítulo 23

REPERCUSIONES Y RETOS DE LA MINERÍA DE DATOS

Al lo largo de los capítulos precedentes hemos discutido diferentes técnicas de minería de datos y descrito los conceptos asociados a las mismas. En este último capítulo abordamos algunas cuestiones finales relativas al impacto social de la minería de datos y a las cuestiones éticas y legales derivadas de la privacidad de los datos utilizados. También trataremos el problema de la escalabilidad de las técnicas de minería de datos a bases de datos voluminosas. Comentaremos algunas soluciones parciales o algunas más globales, como la minería de datos distribuida.

Para terminar, en este capítulo presentamos algunos de los retos a los que la minería de datos debe enfrentarse y que creemos van a marcar la dirección en la evolución y desarrollo de esta tecnología en los próximos años.

23.1 Impacto social de la minería de datos

No cabe duda que la minería de datos se ha convertido en los últimos años en un término muy popular. Cada vez son más los usuarios, las aplicaciones, las investigaciones y los desarrollos relacionados con ella, y crecen los sistemas *software* que afirman ser productos de minería de datos. La minería de datos ha ido evolucionando desde su aparición, desarrollando nuevos métodos para adaptarse a las necesidades de una amplia variedad de dominios de aplicación.

Por todo ello, esta disciplina se ha convertido en una tecnología ampliamente reconocida por compañías de todo tipo, organizaciones, instituciones públicas e individuos. El uso de información aprendida desde los datos es necesario para mantener la competitividad en todos los entornos empresariales, así como optimizar las decisiones de las instituciones públicas para dar un mejor servicio a los ciudadanos. Los almacenes de datos, que han hecho posible el almacenamiento de grandes volúmenes de datos en un mismo repositorio,

junto con el incremento en potencia de la computación, son las causas de que las empresas de hoy en día busquen herramientas y tecnologías capaces de extraer información útil de los datos.

El nivel real de implantación y arraigo de la minería de datos en la sociedad es subjetivo y muy difícil de determinar, y lógicamente depende de países y zonas geográficas. No obstante, algunos muestran una cierta penetración a diferentes ámbitos. Por ejemplo, las grandes empresas y organizaciones, así como los gobiernos, están cambiando la perspectiva de un análisis de datos más tradicional, más descriptivo y confirmatorio, a una minería de datos más orientada al sistema de información, a la búsqueda de modelos y patrones, y, sobre todo, una visión más en el fin (sacar partido de la información que nos rodea) que en el medio (este medio se encuentra, justamente, en la "minería de datos"). Si bien las grandes organizaciones incorporan o incluso sustituyen terminologías, métodos y metodologías en los departamentos de estadística e investigación operativa, prospectiva y similar, para las pequeñas y medianas empresas la minería de datos representa su primera oportunidad de entrar en el mundo del análisis de la información, algo que parecía imposible hace unos pocos años, por el coste en recursos humanos y materiales que se requería para ello. No sólo existen herramientas de minería de datos cada vez más completas y accesibles (o incluso gratuitas), sino que los sistemas de gestión de bases de datos están incluyendo primitivas, lenguajes e incluso entornos de minería de datos. Hasta las aplicaciones ofimáticas (hojas de cálculo) comienzan a incluir este tipo de herramientas.

Como en cualquier otra faceta tecnológica, la web está contribuyendo mucho a esta difusión. Cada vez más compañías llevan a cabo sus actividades a través de la web, sobre todo la relación con los clientes o entre proveedores. Los datos recogidos (patrones de compra y de navegación) pueden proporcionar mucha más información sobre los clientes tanto individualmente como en grupo, y esta información puede ayudar a las empresas a ofrecer unos servicios más personalizados y adaptados a las características de los clientes. Servir a las necesidades de los clientes puede significar un ahorro económico sustancial para las empresas (por ejemplo, evitando hacer campañas publicitarias generales), y un beneficio para los clientes que comprueban satisfechos cómo les ofrecen productos en los que están interesados y no pierden su tiempo en digerir ofertas por las que no están interesados.

Existen otros ámbitos en los que la minería de datos está teniendo un impacto muy importante. Las extensión de aplicaciones de la minería de datos en medicina y, general, el área que se ha venido a llamar bioinformática, tienen unas consecuencias indirectas en la sociedad, que no son desdeñables. Lo mismo ocurre en otras áreas, como en el ámbito de la seguridad o la lucha antiterrorista, campo en el que se está trabajando profusamente.

Pero la minería de datos no es sólo útil en los ámbitos empresariales, institucionales o científicos, sino también a nivel individual [Han & Kamber 2001]. Por ejemplo, la mayoría de navegadores de última generación incluyen métodos de aprendizaje automático (generalmente bayesianos) para clasificar el correo electrónico y detectar los mensajes *spam*. Esto es sólo el principio, nuestra información personal también será procesada y analizada por herramientas que nos sugerirán patrones, nos filtrarán información y nos harán más llevadera y provechosa la nueva sociedad de la información, convirtiéndola en la sociedad del conocimiento.

23.2 Cuestiones éticas y legales

Tal y como acabamos de comentar y hemos ido viendo a lo largo del libro, la minería de datos puede reportar numerosos beneficios en muy diversas aplicaciones. Pero existe también un lado peligroso en la minería de datos que tiene que ver con dos aspectos fundamentalmente: la privacidad de las personas con cuyos datos se trabaja y el uso descuidado de los modelos obtenidos. Tanto uno, el uso de los datos, como el otro, el uso de la información, tienen implicaciones éticas y legales.

23.2.1 Privacidad

Casi todas las constituciones y leyes supranacionales del mundo afirman taxativamente que deben observarse una serie de medidas para un tratamiento riguroso de la información privada de las personas. Un ejemplo más que ilustrativo es el de los datos médicos personales. A nadie le gustaría que su historial médico apareciese en una página web o que un trabajador del archivo de un hospital pudiera ojear libremente este historial, a la caza de algún famoso con alguna enfermedad singular. En un mundo donde la información fluye tan rápidamente, evitar este tipo de problemas obliga a limitar en gran medida la manera en la que se recogen y almacenan los datos, para que incluso los operarios tengan limitado ese acceso y, fundamentalmente, para que las bases de datos no se puedan ceder de unos a otros libremente, ni usarse para fines diferentes de para aquellos para que los datos fueron recogidos.

Existe ya una tradición de directivas y leyes para esta protección de datos. Por ejemplo, una iniciativa europea para la protección de los datos, conocida como *European Data Protection Directive* (Directiva 95/46/EC, *Official Journal of the European Communities*, 23 de noviembre de 1995, Nº L. 281, p. 31), establece el marco legal que se debe cumplir cuando se comercia con información personal en cualquiera de los estados miembros de la Unión Europea. Estas directrices garantizan unos derechos básicos sobre los datos recogidos de un individuo:

- derecho a acceder a los datos
- derecho a rectificar cualquier error en los datos
- derecho a conocer de dónde se obtuvieron los datos
- derecho a recurrir contra tratamientos ilegales
- derecho a denegar el permiso para usar tus datos en campañas publicitarias

Las leyes nacionales generalmente suelen concretar estas directivas. Por ejemplo, en España existe una Ley de Protección de Datos de Carácter Personal (Ley Orgánica 15/1999, de 13 de diciembre de 1999) que tiene por objetivo “garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar”.

En América la tendencia es similar, aunque se parte generalmente de unas leyes con menor nivel de protección que en Europa. De hecho, en Estados Unidos estos niveles de protección y derechos se relajan mucho si quien realiza la minería de datos es algún organismo o agencia federal. También pueden tener interpretaciones diferentes dependiendo del dominio de los datos, por ejemplo el ámbito médico (“The Health Insurance Portability and Accountability Act, 1996”, “Standards for Privacy and

Individually Identifiable Health Information, 1999”, “Protection of Human Subjects” (“The Common Rule”).

El uso potencial de las técnicas de minería de datos puede significar que los datos se usen para fines distintos para los cuales se recopilaron. Para evitar problemas relacionados con la privacidad de los datos y la intimidad, se debería establecer bajo qué condiciones y para qué propósitos se van a usar los datos antes de que éstos sean recogidos. De hecho, en muchos países se obliga ya a establecer dicha cláusula a la hora de recoger los datos. En ese caso, deberíamos poner, por ejemplo, si se trata de un supermercado, alguna frase del estilo “cuando usted dé de alta la nueva tarjeta *superclub*, podremos analizar sus perfiles de compra cada vez que use la tarjeta, con el objetivo de mejorar nuestro servicio, o para realizar un análisis posterior junto al resto de ventas, siempre de acuerdo con la legislación vigente”.

Dado que, generalmente, los datos no se recogen directamente para minería de datos (desgraciadamente se piensa en hacer minería de datos cuando los datos ya se han recogido), muchas veces no tenemos este permiso. En estos casos, lo más razonable es utilizar una de estas técnicas:

- Eliminar claves e información identificativa: nombre y apellidos, dirección, documento de identidad, números de tarjetas o de cuentas, etc. En la mayoría de los casos esta información no nos interesa, o sólo nos interesa parcialmente (código postal, edad, etc.). Si la minería de datos se va a utilizar para realizar decisiones personalizadas, se debe crear una tabla de correspondencias (entre la antigua tabla, incluyendo todos los datos, y la nueva tabla, que sólo incluye los datos no sensibles).
- Agregar los datos: para algunas aplicaciones de minería de datos se pueden agregar los datos (por zonas geográficas, por períodos, etc.) de tal manera que ya no exista información personalizada. De hecho, la información agregada (por ejemplo las estadísticas) pueden intercambiarse y publicarse sin ningún problema, siempre que de esa información global no pueda inferirse la información particular (por ejemplo, porcentaje de morosos en los clientes de un banco por nacionalidades, y resulta que todos los clientes son españoles menos uno que es mejicano).

En cualquier caso, aunque en muchos casos la ley permite utilizar datos detallados dentro de la misma organización, si el grupo de trabajo de minería de datos está formado por mucha gente, es preferible que el análisis se realice ya sobre la base de datos sin información sensible, habiendo realizado en un primer lugar la eliminación de información identificativa o habiendo agregado convenientemente.

23.2.2 Modelos. Errores y discriminación

El uso de los modelos extraídos, como hipótesis que son, puede conllevar errores. De hecho, es rara la aplicación de la minería de datos en la que no se produzcan errores. En algunos casos estos errores están claros cuando se analiza el problema, pero en otros la atención se puede centrar tanto en los objetivos principales que nos podemos olvidar de los efectos secundarios. Por ejemplo, como se menciona en [Dunham 2003], que un individuo haga compras con su tarjeta de crédito similares a las que se hacen cuando la tarjeta es robada no significa que realmente la tarjeta haya sido robada, y que, por tanto, el individuo sea un

ladrón. Poner bajo sospecha a un cliente honesto puede ser a veces casi peor que detectar a un cliente deshonesto. Este tipo de situaciones se pueden evaluar de múltiples maneras (costes, análisis ROC, etc.), como vimos en el Capítulo 17.

Quizás un problema legal que se plantea cuando se utilizan muchos modelos obtenidos por minería de datos y se combinan, es asumir responsabilidad. Cuando un modelo se equivoca, ¿quién asume las responsabilidades? La respuesta no puede ser más tajante, la responsabilidad es siempre del que se deja asesorar por el modelo, nunca de quién lo generó y mucho menos de quién desarrolló la herramienta con la que se hizo. Esta interpretación es la misma que ocurre con los errores del *software*.

Un problema más sutil es cuando los modelos se utilizan para discriminar, y los ejemplos que estamos discriminando o clasificando son personas [Witten & Frank 2000]. Por ejemplo, basándose en datos personales (como por ejemplo, el sexo) una compañía de seguros podría rechazar el suscribir una póliza para el coche. De hecho, esto ha pasado en algunos casos y ha sido denunciado por discriminación, pese a que el modelo y los datos digan tozudamente que las mujeres generan menos accidentes graves que los hombres. Sin embargo, la situación es compleja y todo depende del tipo de aplicación. Por ejemplo, usar información referente al sexo o a la raza puede ser ético para el diagnóstico médico pero no para determinar si se debe o no conceder un crédito bancario.

En definitiva, tanto en el uso de los datos como en el de los modelos, la legislación contempla muchos casos comunes, y el equipo que realiza la minería de datos debe conocer perfectamente la ley al respecto en su país y sobre los datos que está trabajando, pero, en cualquier caso, la dimensión ética debe estar presente en la mente de los que realizan y utilizan la minería de datos, no sólo en cuanto a los datos usados sino también en cuanto al uso de los patrones minados.

23.3 Escalabilidad. Minería de datos distribuida

Una de las dificultades a las que se ha enfrentado la minería de datos desde sus inicios y que, al mismo tiempo, ha justificado su desarrollo, es la necesidad de tratar con grandes volúmenes de datos. El término “grande” es, lógicamente, subjetivo; lo que hace unos años eran grandes volúmenes de datos hoy son pequeños volúmenes. No obstante, las exigencias siempre van a la par que las capacidades, y las cantidades de datos que se quieren minar hoy en día son también mucho mayores que las que se aspiraba a minar hace unos años. Por tanto, pese a que los computadores cada día son más potentes, también tenemos mayores volúmenes de datos y exigimos modelos más precisos y comprensibles.

Esto hace que la escalabilidad de las técnicas de minería de datos sea fundamental. El término escalabilidad, aplicado a una técnica o algoritmo, significa que si para un tamaño x una técnica requiere un tiempo y memoria t , para un tamaño, digamos, $10x$, la técnica requerirá no mucho más que $10t$. Dicho de una manera más precisa, queremos un comportamiento lineal respecto al tiempo (y también a la memoria necesaria) para ejecutar una técnica de minería de datos según van creciendo los datos. A la medida que crecen los volúmenes de datos, si no tenemos escalabilidad, en unos años no podremos aplicar algunas técnicas al conjunto de bases de datos.

Otra cuestión que se está evidenciando en los últimos años es el crecimiento en talla y número de las bases de datos y de los almacenes de datos. Muchos de los algoritmos de

aprendizaje son computacionalmente complejos y requieren que todos los datos residan en memoria, lo cual es inmanejable para muchas aplicaciones reales.

Se ha investigado intensamente en algoritmos que escalan bien con grandes volúmenes de datos [Cohen 1995a; Han et al. 1996], incluyendo técnicas para realizar muestreos eficientes, selección de características, restricción del espacio de búsqueda y aplicación del conocimiento del dominio (un buen resumen se puede encontrar en [Provost & Kolluri 1997]). Muchas de estas aproximaciones mejoran las prestaciones de los algoritmos de aprendizaje vistos en los capítulos de la Parte III.

De los dos problemas, tiempo de ejecución y memoria necesaria, el problema del espacio puede resolverse si el algoritmo trabaja de forma incremental: se procesan las instancias de una en una (o en pequeños grupos) actualizando cada vez el modelo. De esta forma sólo las instancias que constituyen el conjunto de entrenamiento deben residir en memoria principal. La incrementalidad (de la que ya hemos hablado en la Sección 19.6) también facilita la minería cuando se incorporan nuevos datos a la base de datos, ya que permite actualizar el modelo sin tener que minar de nuevo el conjunto de datos entero. Algunos ejemplos de algoritmos que trabajan incrementalmente son el método Naive Bayes así como algunas versiones incrementales de algoritmos de inducción de árboles de decisión. No obstante, para otros muchos métodos de los vistos en el libro todavía no se han desarrollado versiones incrementales.

Pese a todas estas mejoras, en la práctica, algunas de estas soluciones no son viables. El problema es que asumen que los datos residen en memoria, mientras que en la mayoría de los casos los datos residen en disco. Incluso en algunos casos la vista minable es tan grande que, simplemente, no cabe en memoria principal. En algunos apartados de este libro (véase por ejemplo Capítulo 11) se han tratado adaptaciones de algoritmos para realizar una buena paginación e intercambio de datos entre el disco y la memoria principal, evitando múltiples pasadas sobre las mismas tablas. En otros apartados hemos tratado aproximaciones paralelas (véase por ejemplo el Capítulo 9, Sección 9.2.1, o el Capítulo 15, página 395).

La paralelización es una forma de reducir la complejidad temporal del aprendizaje. La idea es partir el problema en pequeñas partes, resolver cada una de ellas en un procesador y luego combinar los resultados. Para hacer esto se deben crear versiones paralelas de los algoritmos de aprendizaje. Algunos algoritmos son paralelizables de forma natural. Por ejemplo, el método del vecino más próximo puede distribuirse entre varios procesadores partiendo los datos en varias muestras y permitiendo que cada procesador encuentre el vecino más próximo en su parte del conjunto de entrenamiento. Los algoritmos de aprendizaje de árboles de decisión también se pueden paralelizar dejando que cada procesador construya un subárbol del árbol completo. *Bagging* y *stacking* son también algoritmos paralelizables. Sin embargo, como se comenta en [Witten & Frank 2000], la paralelización es sólo un remedio parcial, ya que con un número fijo de procesadores la complejidad temporal asintótica del algoritmo no puede mejorarse.

En relación con esta última aproximación para la escalabilidad de la minería de datos, y debido a la existencia de fuentes de datos heterogéneas, de múltiples fuentes o almacenes de datos, de interconectividad con la web, y, en general, una visión más abierta de la minería, ha cobrado renombre recientemente una nueva aproximación: la minería de datos distribuida.

23.3.1 Minería de datos distribuida

Un sistema de información distribuido consta de un sistema de gestión de bases de datos distribuidas, una base de datos distribuida (donde los datos se distribuyen entre varias bases de datos) y una red para la interconexión. Una de las aproximaciones para operar con estas bases de datos es la arquitectura cliente-servidor, donde el objetivo es comunicar múltiples usuarios con múltiples servidores de forma transparente. Si atendemos a los datos que hay que una bases de datos distribuidas, podemos distinguir entre bases de datos homogéneas, cuando el mismo esquema está repetido en cada servidor y se tienen, por tanto, los objetos (las tuplas), repartidos, o bases de datos distribuidas heterogéneas, donde cada parte recoge algunas tablas o incluso atributos diferentes de la misma tabla. No vamos a entrar aquí en detalle sobre las bases de datos distribuidas; para más información, se recomienda [Ozs & Valduriez 1999].

La minería de datos sobre bases de datos distribuidas se denomina, sencillamente, minería de datos distribuida (*Distributed Data Mining*, DDM). A pesar de resultar muy interesantes para muchas aplicaciones, la minería de bases de datos distribuida ha recibido atención desde hace muy poco tiempo. En esta sección vamos a revisar algunas propuestas para la minería de datos distribuida.

Una posibilidad para minar estas bases de datos distribuidas [Thuraisingham 1999] es que cada procesador o nodo distribuido disponga de un componente de minería encargado de minar los datos en la base de datos local y luego se combinen todos los resultados, como se ve en la parte izquierda de la Figura 23.1. Otra posibilidad es empotrar la herramienta de minería de datos en el sistema de consulta distribuido, como se ve en la parte central de la Figura 23.1. Una aproximación alternativa es implementar una única herramienta de minería de datos en la parte superior del sistema distribuido que actúa sobre una vista integrada de las distintas bases de datos, como se ilustra en la parte derecha de la Figura 23.1.

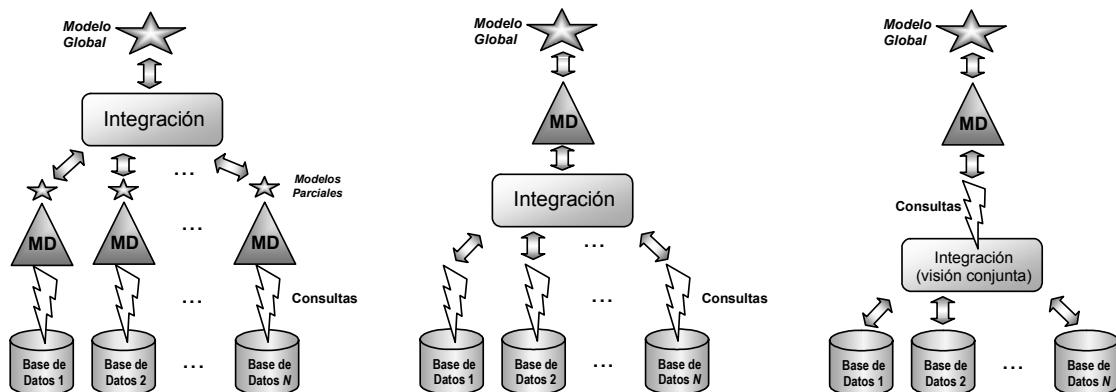


Figura 23.1. Tres arquitecturas diferentes para minería de datos distribuida.

En cualquiera de las tres arquitecturas, muchas de las técnicas de minería de datos que se han definido son extensiones de técnicas clásicas de minería de datos para bases de datos relacionales al caso de fuentes distribuidas. A continuación, resumimos aquí algunas de las propuestas que se relatan en [Kargupta et al. 2000].

El análisis de datos distribuido desde datos homogéneos, que es el caso más sencillo, conlleva la combinación de diferentes modelos de datos extraídos desde cada repositorio, como si fueran muestras diferentes (poblaciones diferentes) de los mismos individuos. Ya hemos visto algunas técnicas de combinación de modelos en el Capítulo 18, algunas de las cuales pueden extenderse para la agregación de modelos múltiples en DDM, como la técnica de *bagging* [Breiman 1999] o el *stacking* [Ting & Low 1997]. El meta-aprendizaje [Chan & Stolfo 1993] ofrece otra clase de técnicas para minar datos distribuidos homogéneos. Esta aproximación consiste en usar primero técnicas de aprendizaje supervisado para detectar conceptos en las bases de datos locales, y después aprender meta-conceptos desde un conjunto de datos generados usando los conceptos localmente aprendidos dando lugar a un meta-clasificador. El sistema JAM [Stolfo et al. 1997] presenta una técnica que es similar al meta-aprendizaje pero que está especialmente diseñada para inducir modelos descriptivos desde los clasificadores aprendidos en el entorno distribuido. También se han desarrollado técnicas basadas en el aprendizaje bayesiano [Yamanishi 1997]. La idea se basa en considerar agentes bayesianos que estiman los parámetros de la distribución objetivo y una población de sistemas de aprendizaje que combinan las salidas de los modelos bayesianos producidos. En [Cho & Wüthrich 1998] se presenta una aproximación fragmentada en la que por cada fuente de datos distribuida se genera una regla (la más simple y mejor), se ordenan las reglas de acuerdo a cierto criterio y se seleccionan las mejores para formar el conjunto de reglas final. Otras aproximaciones son el sistema PADMA [Kargupta et al. 1997b] que implementa un algoritmo de agrupamiento distribuido y el algoritmo FDM (*Fast Distributed Mining*) [Cheung et al. 1996] usado para minar reglas de asociación desde fuentes distribuidas.

También se han propuesto algunos métodos para analizar datos desde fuentes heterogéneas, aunque, debido a su dificultad, esta cuestión ha sido menos tratada en la literatura. Cuando se trabaja con datos heterogéneos las diferentes características de un mismo objeto (atributos) están distribuidas en diferentes localizaciones. Uno de los problemas, por lo tanto, es establecer algún método para definir la correspondencia entre los diferentes componentes del mismo objeto, es decir entre las diferentes filas almacenadas en sitios distintos. En [Kargupta et al. 2000] se presenta un marco colectivo para generar modelos predictivos que denominan minería de datos colectiva (*Collective Data Modelling, CDM*). El marco CDM consiste en crear modelos localmente correctos a lo que sigue la generación de un modelo de datos global a través de la agregación de los resultados locales. El marco CDM se ha empleado para aprender árboles de decisión y para la regresión multi-variante. La minería de datos distribuida sobre bases de datos heterogéneas es especialmente útil cuando tenemos una dimensionalidad muy alta, y entonces tenemos un subconjunto de las características de cada objeto en cada una de las bases de datos. Una técnica utilizada en esta configuración es el análisis de componentes principales (PCA), vista en el Capítulo 4, para extraer características independientes y significativas desde cada una de las bases de datos. El PCA puede usarse dentro de un marco colaborativo denotado como CPCA [Kargupta et al. 2001].

23.4 Tendencias futuras

En la breve historia de la minería de datos, se han cumplido algunas expectativas y se han dejado abiertas otras muchas. En particular, se espera una minería de datos más automática, más sencilla, con más fiabilidad, con patrones más novedosos y más eficiente. De hecho, según autores, se pueden destacar todavía más retos. Por ejemplo, Han y Kamber [Han & Kamber 2001] afirman que para que la minería de datos sea completamente aceptada como una tecnología, se deben resolver algunos problemas principalmente relacionados con la eficiencia y la escalabilidad, la interacción con el usuario, la incorporación de conocimiento de base, las técnicas de visualización, la evolución de lenguajes de consultas de minería de datos estandarizados y mejorar el tratamiento de datos complejos, entre otros.

Como ya mencionamos en el primer capítulo y hemos ido mostrando a lo largo de los demás, la minería de datos es el resultado de la integración de múltiples técnicas. Por tanto, los retos que se plantean han de resolverse por avances en estas disciplinas pero, fundamentalmente, por la combinación de estas disciplinas.

Si comenzamos por la materia prima de la minería de datos, es evidente que disponer de buenos datos es clave para esta disciplina ya que la calidad del conocimiento extraído depende tanto o más de los datos usados que de la técnica empleada. Muchos de los datos que se recopilan son imprecisos, incompletos o inciertos. En algunos casos, los datos no se encuentran disponibles en un formato apto para su tratamiento informático y puede ni siquiera saberse dónde encontrarlos. Éste es uno de los grandes retos de la minería de datos: identificar los datos, almacenarlos en repositorios para que puedan ser computacionalmente procesados, limpiarlos y darles el formato apropiado para ser minados. Aunque ya se han aportado algunas soluciones parciales de las que hemos dado cuenta en los capítulos 3, 4 y 5, en particular la tecnología de almacenes de datos, todavía no hay una buena aproximación general que determine qué hacer para tener datos con una calidad óptima [Thuraisingham 1999]. Por tanto, todo esfuerzo encaminado a mejorar la fase de preparación de datos: técnicas de recopilación, almacenes de datos, limpieza de datos, técnicas de muestreo, transformación, lenguajes de consultas, etc., tendrá un efecto importante de cara a la minería de datos.

Uno de los principios de la minería de datos es que tiene que trabajar de forma eficiente y efectiva con grandes bases de datos. Como hemos dicho, los conjuntos de datos masivos y con una alta dimensionalidad crean espacios de búsqueda combinatoriamente explosivos e incrementan la probabilidad de que el algoritmo de minería de datos requiera un tiempo excesivo y además encuentre patrones no válidos [Fayyad et al. 1996b]. Por lo tanto, existe la necesidad de adaptar las técnicas existentes y de inventar nuevos métodos para manejar la alta dimensionalidad y el alto número de observaciones presentes en los conjuntos de datos masivos. La escalabilidad de las técnicas requiere un trabajo considerable tanto en los fundamentos teóricos como en las pruebas con conjuntos de datos cada vez mayores. Una buena gestión del procesamiento entre memoria y disco, el uso de índices específicos para la minería de datos y de compactación, puede ser crucial para obtener esta eficiencia.

En lo que a las herramientas respecta, existen herramientas multi-estratégicas que son capaces de manejar múltiples técnicas, como se puede ver en el apartado de *suites* del Apéndice A. Por otra parte, muchas de las herramientas de minería de datos requieren que los usuarios sean en cierto grado expertos [Zhang & Zhang 2002]. Para que los productos y

herramientas sean aceptados por una amplia mayoría es necesario que se diseñen mejores interfaces de usuario, más amigables y que permitan a un usuario final poco técnico alcanzar buenos resultados. Esto acercará este tipo de herramientas a las empresas pequeñas y medianas, permitiéndoles incorporar la minería de datos como una actividad más. Ambas cuestiones podemos englobarlas bajo la idea de construir entornos interactivos e integrados que proporcionen una rápida respuesta, unas altas prestaciones y que asistan a los usuarios en la selección de la herramienta y técnica adecuada para alcanzar sus objetivos. En este sentido, en [Fayyad et al. 1996b] se apunta que es necesario hacer más énfasis en la interacción hombre-máquina y menos énfasis en la automatización total, con el ánimo de soportar tanto a usuarios expertos como novatos. Además, las herramientas interactivas facilitan la incorporación de conocimiento previo sobre el problema, algo importante en la minería de datos.

Siguiendo con las herramientas, una de las direcciones prometedoras en investigación y desarrollo en la minería de datos es la construcción de sistemas capaces de proporcionar soluciones específicas para cada tipo de negocios (distribución, medicina, marketing...), lo que en [Zhang & Zhang 2002] se denomina sistemas que proporcionan una solución vertical. Estas soluciones integran la lógica de un negocio de dominio específico con el sistema de minería de datos. Esto contrasta con la situación actual en la que los sistemas de minería, aunque puedan integrar muchas técnicas, son horizontales, es decir, no están especialmente diseñados para ser aplicados a un determinado dominio.

La comprensibilidad de los patrones, así como la capacidad para podar los resultados de la minería, es otra de las cuestiones cruciales. En muchas aplicaciones es importante hacer que la información descubierta sea más comprensible por los humanos (por ejemplo, usando representaciones gráficas y visualización de datos o generando lenguaje natural). No debemos olvidar que, generalmente, el usuario final no es un experto en aprendizaje automático ni en estadística. Los avances que faciliten la integración en los entornos de decisión, la simulación de los modelos para entender sus consecuencias, las herramientas que integren diferentes modelos y los conviertan en modelos globales, y que ayuden en su monitorización y revisión, serán cruciales en facilitar este uso del conocimiento extraído.

Pese a la cantidad y dificultad de todos estos retos, la minería de datos alza el vuelo desde una situación de partida aventajada para lograrlos: es una disciplina muy joven aunque basada en otras muchas con experiencia, goza de un impetuoso interés en el mundo empresarial y una dilatada vocación desde el mundo académico. Sea lo que sea lo que nos depare el futuro de la minería de datos, éste pinta fascinante.

APÉNDICES

Se incluyen dos apéndices en los que se recoge una descripción de los sistemas comerciales y gratuitos de minería de datos, y los conjuntos de datos utilizados a lo largo del libro. A estos apéndices siguen las referencias bibliográficas y un índice de términos.

APÉNDICES

- A. Sistemas y herramientas de minería de datos**
- B. Datos de ejemplo**

APÉNDICE [A]

SISTEMAS Y HERRAMIENTAS

DE MINERÍA DE DATOS

Vicent Estruch Gregori

Se pueden encontrar tanto en ámbitos comerciales como académicos una serie de entornos *software* diseñados para dar soporte al ejercicio de minería de datos. En este anexo revisamos algunos de los entornos de minería de datos más populares que actualmente se encuentran disponibles para el usuario. Para resaltar de forma más clara las principales diferencias entre ellos, hemos clasificado los entornos en tres grupos: librerías, *suites* y herramientas específicas. Las características estudiadas en todos ellos hacen referencia a aspectos tales como la facilidad y comodidad de uso, la portabilidad, la accesibilidad de la información, los modelos disponibles, la información estadística asociada a un modelo, etc.

Librerías

Las librerías de minería de datos comprenden un conjunto de métodos que implementan las funcionalidades y utilidades básicas propias de la minería de datos: acceso a datos, inferencia de modelos (árboles de decisión, redes neuronales, métodos bayesianos, etc.), exportación y comprobación de resultados, etc. Las librerías facilitan el desarrollo de tareas de minería que son estructuralmente más complejas que las anteriormente mencionadas, como pueden ser el diseño de experimentos, el contraste de modelos, la creación de modelos combinados, así como la integración de diversas técnicas de minería de datos. Las librerías constituyen una interfaz para el desarrollador, por lo que para su manejabilidad se precisa de conocimientos de programación. A continuación esbozamos las características y particularidades de dos de las librerías de minería de datos disponibles⁶⁹.

⁶⁹ WEKA, que en su inicio era sólo una librería, hoy en día es un paquete integrado (*suite*) y, por tanto, lo veremos en el apartado siguiente.

Xelopes

Xelopes (*eXtended Library fOr Prudsys Embedded Solutions*) es una librería con licencia pública GNU para el desarrollo de aplicaciones de minería de datos, implementada por *Prudsys AG* en colaboración con *Russian MDA specialist ZSoft Ltd*. La librería implementa de una manera eficiente la mayoría de los algoritmos de aprendizaje, pudiendo el usuario hacer uso de ellas para desarrollar aplicaciones particulares de minería de datos. Además Xelopes es extensible, es decir, el usuario puede incorporar sus propios métodos a la propia librería.

Esquemáticamente, Xelopes se caracteriza por:

- Acceso a datos: existe una clase especial que permite dar uniformidad a todos los modos de accesos de datos permitidos. Así, el usuario puede acceder a archivos .log, archivos de base datos o confeccionarse su propio formato de datos.
- Modelos: entre otros, se contemplan los siguientes:
 - Árboles de decisión lineales y no-lineales (particiones no rectangulares).
 - Máquinas de vectores soporte.
 - Redes neuronales.
 - Métodos de agrupamiento (K medias, jerárquico, etc.).
 - Métodos de reglas de asociación (métodos para el problema clásico de la cesta de la compra, cesta de la compra con taxonomías, etc.).
- Exportación de datos: existen métodos para exportar los modelos y sus resultados a otros entornos de minería de datos, soportando el estándar PMML.

La librería está desarrollada bajo el estándar MDA (*model drive architecture*), está disponible para C++ y Java, y también existe una interfaz para CORBA. Para más información, se puede consultar <http://www.prudsys.com/Produkte/Algorithmen/Xelopes>.

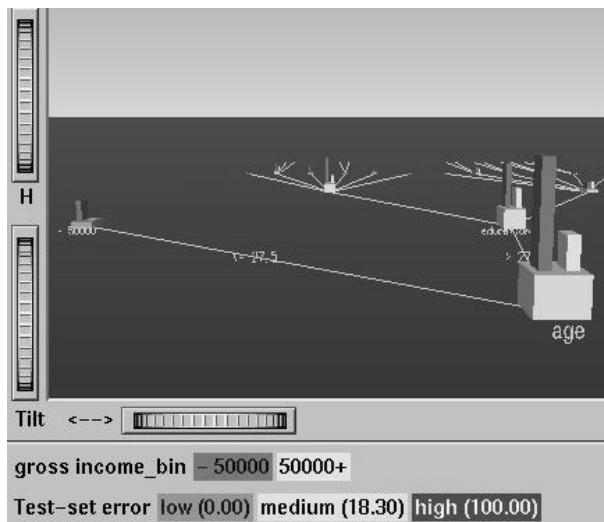
MLC++

MLC++ (*Machine Learning library in C++*) es un conjunto de librerías y utilidades para facilitar las tareas de testear y comparar la eficiencia y resultados proporcionados por diversos algoritmos sobre un mismo problema. Fueron desarrolladas originariamente, hasta la versión 1.3.x y de dominio público, por la Universidad de Stanford. Las versiones posteriores de esta librería se distribuyen por *Silicon Graphics* bajo dominio de investigación a excepción de las versiones 1.3.x que aún son distribuidas bajo licencia de dominio público y por la misma empresa.

La potencia de MLC++ se incrementa por su capacidad de integración con MineSet (*Silicon Graphics*), herramienta ésta última que ha dejado de desarrollarse desde septiembre de 2001 directamente por Silicon Graphics y en la actualidad es mantenida y actualizada por uno de los socios de la compañía (*Purple Insight*) especializado en el desarrollo de utilidades para la visualización de información en el campo de la minería de datos. MineSet proporciona extensiones a MLC++ para acceso a bases de datos, tratamiento de la información (filtrado, construcción de atributos derivados, etc.) y un potente soporte visual.

A nivel general, MLC++ se caracteriza por:

- Acceso a datos: archivos con formato plano siguiendo el estilo de los archivos del repositorio UCI.
- Transformaciones de datos: se dispone de un número de transformaciones sobre datos muy reducido (eliminación de atributos, *binnid*, y métodos simples para la construcción de nuevos atributos).
- Modelos de aprendizaje: éstos se hallan encapsulados mediante objetos. Encontramos implementadas técnicas sobre tablas de decisión, ID3, aprendizaje de árboles de decisión perezosos, árboles de decisión con opciones, métodos bayesianos y el Perceptrón. Además ofrece interfaces en el mismo estándar para invocar a C4.5, C5.0, CART, CN2, PEBLS, etc.



Visualización del árbol de decisión en MLC++ mediante la herramienta MineSet.

Para obtener más información, se puede consultar <http://www.sgi.com/tech/mlc> o bien <http://www.purpleinsight.com>.

Suites

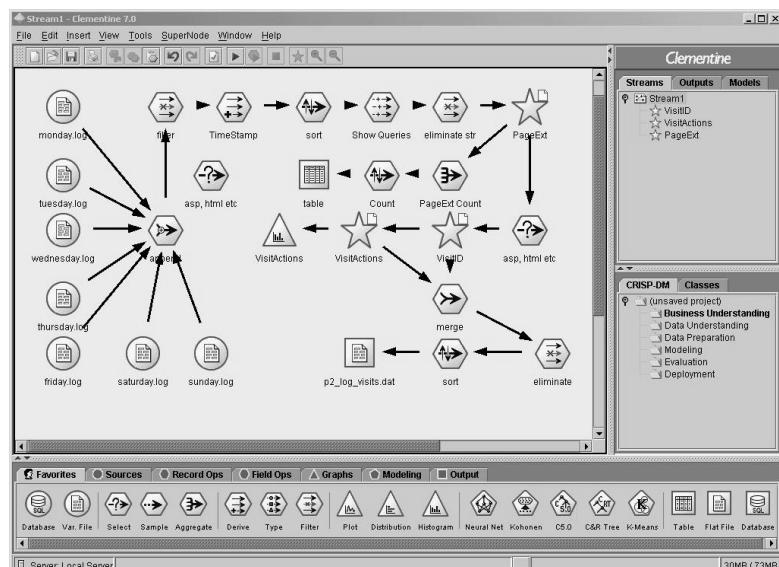
Una *suite* integra en un mismo entorno capacidades para el preprocesado de datos, diferentes modelos de análisis, facilidades para el diseño de experimentos y soporte gráfico para la visualización de resultados. A diferencia de las librerías, su manejabilidad no se halla condicionada a que el usuario posea conocimientos de programación, ya que existe una interfaz (en la mayoría de los casos gráfica) que facilita la interacción entre el usuario y la herramienta.

SPSS Clementine

La empresa SPSS, además de su conocido paquete estadístico, distribuye Clementine, uno de los sistemas de minería de datos más populares del mercado. Se trata de una herramienta visual inicialmente desarrollada por ISL (*Integral Solutions Limited*). En la actualidad esta herramienta, comercializada por SPSS, posee una arquitectura distribuida (cliente/servidor).

Este sistema se caracteriza por:

- Acceso a datos: fuentes de datos ODBC, tablas Excel, archivos planos ASCII y archivos SPSS.
- Preprocesado de datos: *pick & mix*, muestreo, particiones, reordenación de campos, nuevas estrategias para la fusión de tablas, nuevas técnicas para recodificar intervalos numéricos, etc.
- Técnicas de aprendizaje: árboles de decisión (C5.0 y C&RT), redes neuronales (redes de Kohonen, perceptrón multi capa y RBF), agrupamiento (K medias), reglas de asociación (GRI, Apriori, etc.), regresión lineal y logística, combinación de modelos (*boosting* con C5.0).
- Técnicas para la evaluación de modelos guiadas por las condiciones especificadas por el experto.
- Visualización de resultados: Clementine ofrece un potente soporte gráfico que permite al usuario tener una visión global de todo el proceso, que comprende desde el análisis del problema hasta la imagen final del modelo aprendido. Se dispone pues de:
 - Gráficos estadísticos: histogramas, diagramas de dispersión, etc.
 - Gráficos 3-D y gráficos animados.
 - Visualizadores interactivos de las diferentes tareas que realiza el experto.
 - Navegadores para árboles de decisión, reglas de asociación, redes neuronales de Kohonen, agrupamientos, etc.
- Exportación: generación automática de informes (HTML y texto), volcado de los resultados del ejercicio de minería en bases de datos, exportación de los modelos a distintos lenguajes (C, SPSS, HTML, estándar PMML, SQL para árboles de decisión y reglas).



Muestra del entorno gráfico del Clementine.

Clementine es un sistema multiplataforma. La aplicación está disponible para sistemas Windows, Sun Solaris, HP-UX AIX y OS/400.

Para más información, consúltese <http://www.spss.com/spssbi/clementine>.

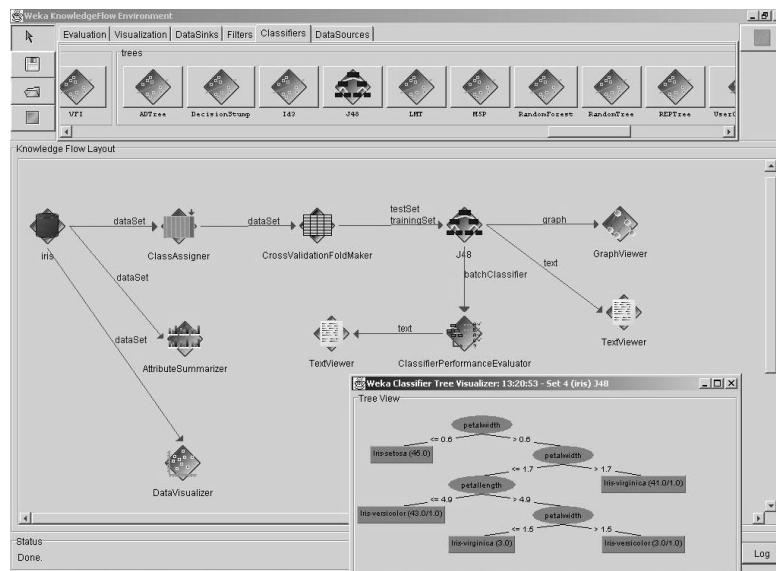
WEKA

WEKA (*Waikato Environment for Knowledge Analysis*) es una herramienta visual de libre distribución (licencia GNU) desarrollada por un equipo de investigadores de la universidad de Waikato (Nueva Zelanda). Como entorno de minería de datos conviene destacar:

- Acceso a datos: los datos son cargados desde un archivo en formato ARFF (archivo plano organizado en filas y columnas). El usuario puede observar en los diferentes componentes gráficos, información de interés sobre el conjunto de muestras (talla del conjunto, número de atributos, tipo de datos, medias y varianzas de los atributos numéricos, distribución de frecuencias en los atributos nominales, etc.)
- Preprocesado de datos (destacar la gran cantidad de filtros disponibles):
 - Selección de atributos.
 - Discretización.
 - Tratamiento de valores desconocidos.
 - Transformación de atributos numéricos.
- Modelos de aprendizaje:
 - Árboles de decisión (J4.8, versión propia del método C4.5).
 - Tablas de decisión.
 - Vecinos más próximos.
 - Máquinas de vectores soporte (método *sequential minimal optimization*).
 - Reglas de asociación (método Apriori).
 - Métodos de agrupamiento (K medias, EM y Cobweb).
 - Modelos combinados (*bagging, boosting, stacking*, etc.).
- Visualización (la interfaz gráfica se compone de diversos entornos):
 - El entorno *Explorer* permite controlar todas las operaciones anteriores (filtrado, selección y especificación del modelo, diseño de experimentos, etc.).
 - El entorno consola (CLI) posibilita la invocación textual de las operaciones anteriores. (También es posible acceder directamente a los métodos que implementan dichas tareas e incorporarlos en el código fuente de la aplicación de minería de datos que se esté programando.)
 - El entorno *Experimenter* facilita el diseño y la realización de experimentos complejos.
 - El proceso global de minería de datos en WEKA se acelera considerablemente gracias al entorno *KnowledgeFlow* que, de una forma gráfica y a modo de flujos de operaciones, permite definir la totalidad del proceso (carga de datos, preproceso, obtención de modelos, comprobación y visualización de resultados).

La herramienta está implementada en Java, luego no presenta problemas de portabilidad, siempre y cuando el sistema disponga de la máquina virtual apropiada. Es interesante

remarcar, que dado que se trata de una herramienta bajo licencia GNU, es posible actualizar su código fuente para incorporar nuevas utilidades o modificar las ya existentes, de ahí que podamos encontrar toda una serie de proyectos asociados a WEKA (*Spectral clustering*, *Kea*, *WEKAMetal*, etc.) que permiten garantizar la continua evolución y adaptación de dicha herramienta.



Detalle gráfico del entorno KnowledgeFlow de WEKA.

Para más información, visítese <http://www.cs.waikato.ac.nz/~ml/WEKA>.

Kepler

Kepler es un sistema desarrollado por un grupo de aprendizaje automático de la *GMD (German General Research Center for Information Technologie)* y transformado en una herramienta comercial distribuida por Dialogis. Kepler está dotado de múltiples modelos de análisis y en su diseño se contemplan aspectos tales como la flexibilidad y la extensibilidad. Entre las herramientas de aprendizaje que proporciona Kepler cabe citar:

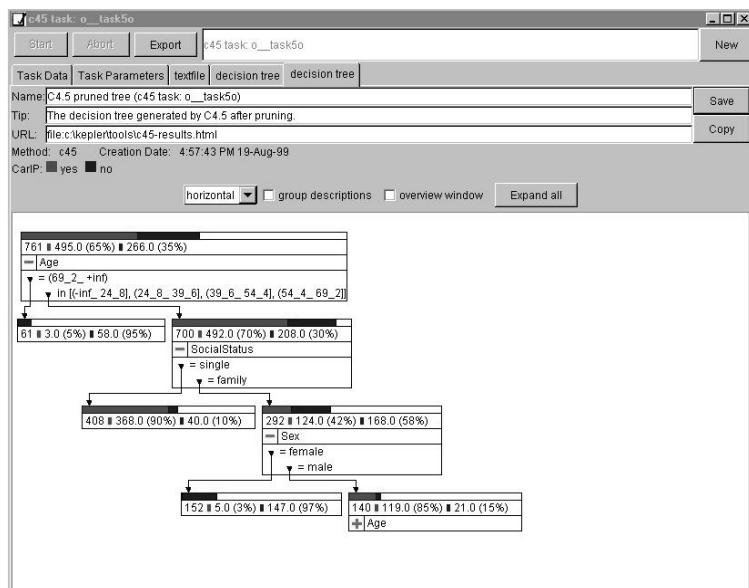
- Árboles de decisión.
- Redes neuronales.
- Regresión no lineal.
- Vecinos más próximos.
- Algoritmos multirelacionales.
- Utilidades estadísticas.

Se cubre, así, un amplio rango de problemas propios de la minería, que van desde la búsqueda de patrones en un conjunto de datos y agrupación de objetos, hasta la clasificación y predicción. Todo el sistema se maneja mediante una interfaz gráfica de usuario programada en Java, la cual permite:

- Preprocesado de datos.

- Especificación de un experimento: elección del modelo, instancia de sus parámetros, etc.
- Manipulación de la representación gráfica de los modelos obtenidos: aunque los resultados obtenidos pueden exportarse a otros entornos o herramientas de análisis o de visualización más específicas. Por ejemplo, los datos de naturaleza cartográfica pueden visualizarse con Descartes (del que se ve un ejemplo en la página 527) que es una herramienta complementaria de Kepler, especializada en el tratamiento gráfico de este tipo de información.

También se dispone de un lenguaje de *script* que permite agilizar el diseño de los experimentos. Los datos de entrada pueden ser importados desde diversos formatos, entre los que se incluye el acceso a la información estructurada en base de datos.



Visualización en Kepler de un árbol de decisión aprendido.

La herramienta se encuentra disponible para plataformas Sun/Solaris y Windows, y se prevé el desarrollo de futuras versiones para plataformas HP y AIX. Para más información, puede verse <http://www.dialogis.de>.

ODMS: Oracle Data Mining Suite (Darwin)

Sistema de minería de datos desarrollado por *Thinking Machines* bajo el nombre de Darwin, y adquirido y comercializado después por *Oracle*. ODMS está diseñado sobre una arquitectura cliente-servidor y ofrece una elevada versatilidad para el acceso a grandes volúmenes de información. Actualmente se distribuye de manera separada y es independiente al *Java Data Mining* (JDM), la librería Java que viene con Oracle desde la versión 9i. La interfaz gráfica potencia la interactividad entre el usuario y el sistema. Técnicamente, el sistema se caracteriza por:

El acceso a datos en diversos formatos:

- Almacenes de datos.

- Bases de datos relacionales (*Oracle, SQL Server, Informix y Sybase*).
- Archivos planos.
- Conjuntos de datos SAS.
- Preprocesado de datos:
 - Muestreo de datos.
 - Obtención de información derivada apoyándose sobre una librería de funciones de tipo matemático, estadístico, lógico, manipulación de caracteres, etc.
 - Particiones de los conjunto de datos.
- Modelos de aprendizaje:
 - Redes neuronales para clasificación y regresión.
 - Regresión lineal (como caso particular de las redes neuronales).
 - Inferencia de árboles de decisión usando el criterio CART.
 - Vecinos más próximos.
 - Aprendizaje bayesiano.
 - Técnicas de agrupamiento (K medias y O-agrupamiento).
- Herramientas de visualización:
 - Representabilidad de los modelos inferidos: destaca su visualizador interactivo de árboles de decisión.
 - Resultados estadísticos: resultados relacionados con el propio modelo o estudios comparativos entre modelos diferentes aplicados sobre un mismo problema (histogramas, gráficos de líneas, sectores, etc., en dos y tres dimensiones).
 - Importación de gráficos desde herramientas comerciales tales como *Microsoft Excel, Microsoft Word y Microsoft PowerPoint*.

La aplicación cliente de ODMS está disponible para entornos Windows, mientras que el servidor puede ejecutarse sobre sistemas Windows, Sun Solaris y HP-UX 11.0. Para más información consúltese <http://otn.oracle.com/products/datamining>.

DBMiner

DBMiner es un sistema interactivo inicialmente desarrollado por *Data Base Systems Research Laboratory* de la *Universidad Simon Fraser* (Canadá) bajo licencia pública, aunque la versión empresarial es comercializada por *DBMiner Technology Inc.* Se trata de un sistema concebido para la extracción de conocimiento en grandes bases de datos relacionales, almacenes de datos y *web*. Para este propósito cuenta con una amplia gama de funciones que ofrecen soporte a tareas de clasificación, agrupamiento, asociación, etc.

En la arquitectura de diseño de este sistema cabe destacar los módulos:

- OLAP (*online analytic processing*): concentra la funcionalidad de manejo multidimensional del almacén de datos.
- OLAM (*online analytic mining*): concentra la funcionalidad específica de minería de datos (clasificación, agrupamiento, etc.).

Los módulos anteriores se encuentran interconectados, debido a que las operaciones OLAP pueden ejecutarse sobre un subconjunto de datos procedentes de las técnicas de minería

aplicadas sobre el cubo de datos, o por el contrario se puede aplicar técnicas de minería sobre una parte concreta del cubo (vista minable) delimitada por una serie de operaciones OLAP.

DBMiner contempla dos modos de trabajo:

- Vía interfaz gráfica: permite al usuario solicitar cualquier operación OLAP y/o OLAM a través de una interfaz estándar.
- Vía lenguaje de *script*: es un lenguaje de consulta similar al SQL denominado DMQL para especificar las diferentes tareas de minería.

Por lo que a la potencia de visualización gráfica se refiere, podemos encontrar:

- Gráficos estadísticos: gráficos de barras, sectores, gráficos de líneas, tablas, etc.
- Visores de árboles de decisión y tablas de reglas.
- Utilidades gráficas para visualizar y gestionar el cubo de datos.

La herramienta está disponible para plataformas Windows. Para más información, véase <http://www.dbminer.com>.

YALE

YALE es un entorno para la realización de experimentos de aprendizaje automático implementado en Java por la *Universidad de Dortmund*. En el momento de escribir este libro, la herramienta es de disponibilidad pública. Las operaciones elementales se encapsulan en los llamados operadores, los cuales son configurables mediante archivos XML. Estos archivos XML, a su vez, pueden ser especificados gráficamente.

El sistema incluye operadores para:

- Importación y preprocesamiento de datos.
- Aprendizaje automático: máquinas de vectores soporte, árboles de decisión, agrupamiento y algoritmos genéticos.
- Validación de los modelos.

El sistema es ejecutable tanto en plataformas UNIX como Windows. Para más información, consultese <http://yale.cs.uni-dortmund.de>.

DB2 Intelligent Miner

DB2 Intelligent Miner es una herramienta comercial distribuida (cliente/servidor) desarrollada por *IBM* y pensada para explotar los masivos sistemas de información de las grandes corporaciones. *DB2 Intelligent Miner* engloba una serie de paquetes destinados a diferentes aspectos de la minería de datos. Entre ellos cabe mencionar:

- *DB2 Intelligent Miner for Data*: es una suite que proporciona un grupo de herramientas para aplicar tareas de minería en bases de datos (*DB2*, archivos planos, variedades de datos accesibles mediante *DataJoiner*, como por ejemplo *Oracle* o *TerraData*. Además se proporciona un módulo que permite importar datos desde archivos *Oracle* y *Sybase* a archivos *DB2*). Las tareas soportadas son:
 - Agrupamiento.
 - Asociaciones.
 - Patrones.

- Clasificación.
- Predicción.
- Análisis de series temporales.

Además conviene resaltar también de esta herramienta que es especialmente útil para trabajar con grandes volúmenes de datos puesto que proporciona herramientas escalables. También posee un lenguaje de programación que permite al usuario especificar los experimentos a realizar y agilizar así el proceso de exploración de los datos.

- *DB2 Intelligent Miner Scoring*: extiende la funcionalidad de la base de datos para poder aplicar técnicas de minería de datos en tiempo real, y cuyo objetivo es reducir el intervalo de tiempo comprendido entre el inicio del análisis del problema y la decisión tomada por el experto.
- *DB2 Intelligent Miner Modeling*: es un paquete orientado al descubrimiento de relaciones específicas entre los datos, más orientado a CRM.
 - Descubrimiento de asociaciones, por ejemplo relaciones entre los productos de la cesta de la compra.
 - Agrupamiento demográfico útil para el estudio de perfiles de venta, hábitos de compra, etc.
 - Aprendizaje de árboles de clasificación para la inferencia de perfiles de clientes.
- *DB2 Intelligent Miner Visualization*: permite visualizar y evaluar gráficamente los resultados de los diferentes modelos de análisis.

Las aplicaciones servidores de las herramientas están disponibles para plataformas Windows (2000, NT, XP), Solaris, AIX, OS (390, 400) y z/OS. Mientras que las aplicaciones clientes lo están para Windows y AIX.

Para más información, véase <http://www-3.ibm.com/software/data/iminer>).

SAS Enterprise Miner

Aunque *SAS Institute* proporciona el famoso *SAS System* para tareas estadísticas y de visualización, *SAS Enterprise Miner* es una de las dos *suites* comerciales que proporciona *SAS Institute* para tareas de minería. Esta primera se centra en la minería de datos tradicional, mientras que la herramienta *SAS Text Miner* amplía su funcionalidad para trabajar con información procedente de archivos de texto, como se ha comentado en el Capítulo 21. A continuación, exponemos la primera de las herramientas.

SAS Enterprise Miner posee una arquitectura distribuida, en donde toda la funcionalidad del sistema es accesible mediante una potente interfaz gráfica de usuario. Su diseño está inspirado en la metodología SEMMA (*Sample, Explore, Modify, Model and Assess*), acuñada por el propio instituto. Tal como se deduce de dicha metodología, las tareas soportadas por el sistema son:

- Acceso a datos:
 - Formato de archivo propio de SAS.
 - Archivos de sistemas de bases de datos comerciales: *Oracle, DB2, Sybase, etc.*
 - Soporte para almacenes de datos (integración de las tareas de minería con los almacenes de datos).

- Preprocesado de datos:
 - Transformaciones: variables nuevas que se definen en base a las ya existentes aplicando una serie de operaciones de tipo matemático o estadístico.
 - Tratamiento estadístico para los valores desconocidos.
 - Filtros para la eliminación de valores extremos.
 - Tareas de muestreo: particionar el conjunto de datos para entrenamiento, validación y comprobación del modelo.
- Modelos:
 - Árboles de decisión: aproximaciones de los métodos CHAID, C&RT y C4.5.
 - Regresión lineal y logística.
 - Redes neuronales: MLP y RBF, con sus variantes.
 - Construcción de modelos múltiples utilizando métodos *ensemble* como *boosting*, *bagging*, etc.
- Evaluación: este módulo permite comparar la eficacia y el rendimiento entre diferentes modelos de aprendizaje.
- Visualización y presentación de resultados: es un potente módulo del sistema que consta de:
 - Visualizador de los resultados obtenidos: gráficos estadísticos en dos y tres dimensiones, visores de árboles de decisión, diagramas ROC, etc.
 - Generador automático de informes: resume la información en un informe con formato HTML para ser visualizado en cualquier explorador.
 - Presentación de la información en lenguaje natural: por ejemplo, las reglas subyacentes en un árbol de decisión pueden ser expresadas por el sistema en lenguaje natural, para que le sean al usuario de una más fácil comprensión.

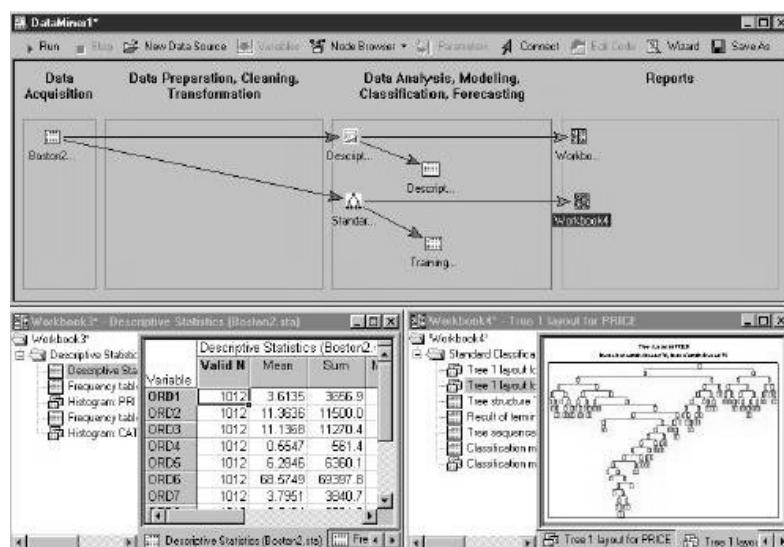
Tanto el programa cliente como servidor de *SAS Enterprise Manager*, puede ser trasladado a diferentes plataformas: Windows, Linux, Solaris, HP-UX, Digital Unix, etc. Para más información, véase <http://www.sas.com>.

STATISTICA Data Miner

Además del conocido paquete estadístico, STATISTICA Data Miner es un sistema visual desarrollado y comercializado por *StatSoft Ltd*. Como puntos de interés de la herramienta conviene comentar:

- Acceso a datos: el sistema está especialmente optimizado para trabajar con grandes volúmenes de datos de entrada. Resaltamos:
 - Importación de datos en diversos formatos: *Microsoft Excel*, tablas de *Dbase*, archivos planos de texto, *Lotus*, bases de datos de *Oracle*, *Microsoft SQL Server* y *Sybase*. También dispone de un formato de archivo propio.
 - Explorador *drill-down* interactivo: el cual posibilita la gestión de tablas multidimensionales. Este módulo es funcionalmente parecido a las herramientas OLAP.
- Preprocesado de datos:
 - Selección de características.

- Muestreo.
- Operaciones estándar de filtrado, transformaciones de variables, tratamiento de valores desconocidos, etc.
- Modelos de análisis: entre los cuales mencionamos,
 - Reglas de asociación.
 - Árboles de decisión: denominados en el entorno como GTREES (*General Classification and Regression Trees*), incorporan nuevas extensiones con respecto a las implementaciones clásicas del algoritmo CART.
 - Agrupamiento: proporciona una serie de implementaciones de gran capacidad de los métodos K medias y EM.
 - Redes Neuronales: incorpora una selección de los métodos de redes neuronales más importantes en el mercado.
 - Utilidades estadísticas para la regresión de modelos lineales, no lineales, regresión múltiple, etc.
- Visualización: el sistema se controla mediante una potente interfaz gráfica, facilitando de esta manera cualquier tarea que desee llevar a cabo el usuario. En cuanto a la representación de resultados, el sistema ofrece:
 - Representación gráfica de sus modelos: navegador de árboles de decisión, visualizadores de la topología de la red neuronal, visualizadores de reglas de asociación, etc.
 - Gráficos estadísticos (en dos y/o tres dimensiones): gráficos de barra, sectores, diagramas de líneas, diagramas de puntos, curvas de nivel, etc.
 - El usuario dispone de la opción de especificar sus propias representaciones gráficas de los datos.



Muestra del entorno gráfico de STATISTICA Data Miner.

El sistema está disponible en plataformas Windows.

Para más información, véase <http://www.statsoft.com/dataminer.html>.

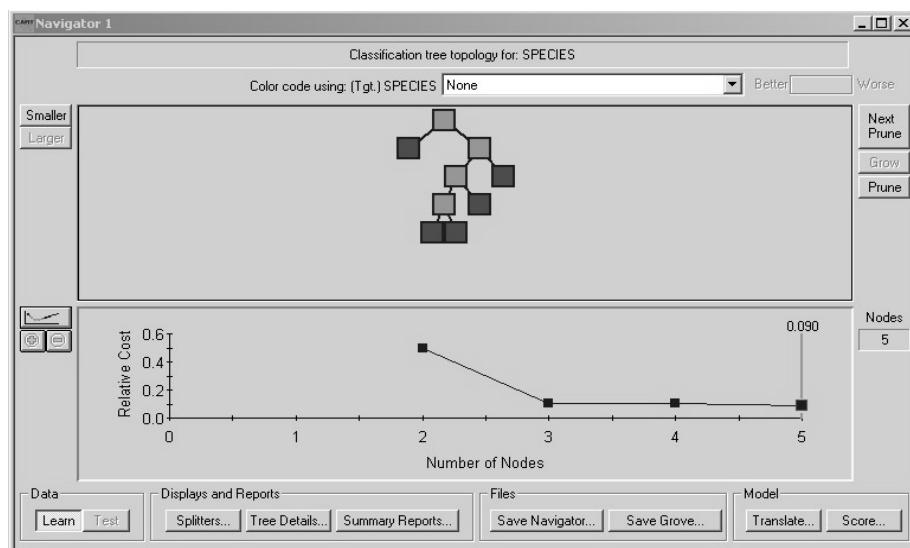
Herramientas específicas

A diferencia de la generalidad propia de las *suites*, este tipo de entornos se caracterizan por centrarse en un determinado modelo (redes neuronales, árboles de decisión, modelos estadísticos, etc.) o en una determinada tarea de minería de datos (clasificación, agrupamiento, etc.). No obstante, pese a incorporar sólo un tipo de técnicas sí que son un entorno que permite realizar todo el proceso de minería de datos⁷⁰. Tampoco se requieren conocimientos de programación para poder ser utilizadas. A continuación exponemos, brevemente, las características de algunas de estas herramientas.

CART

CART es una herramienta gráfica desarrollada y comercializada por Salford Systems. Esta herramienta contiene utilidades para el análisis estadístico y la minería de datos orientada hacia la inferencia de árboles de decisión para tareas de clasificación o regresión. Como entorno de minería de datos destacamos:

- Accesibilidad: CART tiene acceso a más de 70 formatos de archivos diferentes.
- Capacidad de visualización: dispone de herramientas de visualización interactivas, pudiendo el usuario solicitar información detallada del modelo (criterios de partición, distribución de las muestras en cada nodo, etc.)
- Información estadística relativa al modelo: errores de clasificación, influencia de un atributo en la clasificación, etc.



Árbol de decisión aprendido por la herramienta CART.

CART está disponible para plataformas Windows, Linux, Unix (Solaris, IBM AIX, Digital Unix, SGI Irix y HP-UX).

Para más información, véase <http://www.salford-systems.com/index.html>.

⁷⁰ Al final de cada capítulo de la Parte III se han visto técnicas específicas.

AutoClass

Bajo el nombre de *AutoClass* [Cheeseman & Stutz 1996] se reúnen tres distribuciones de este sistema (AutoClass III, AutoClass X y AutoClass C, esta última es de libre distribución) desarrolladas bajo el auspicio de la *NASA*. El propósito es el mismo en todas ellas, tratan problemas de agrupamiento haciendo uso de métodos bayesianos.

Las características a destacar son:

- Dispone de un formato de archivo de entrada propio organizado en filas (muestras) y columnas (características de las muestras).
- Generación de informes que describen las diferentes clases encontradas.
- Extracción de tests predictivos para la clasificación de futuras muestras.

AutoClass C no dispone de utilidades gráficas para la representación de la información. El sistema es multiplataforma. Por ejemplo, *AutoClass C* puede ser ejecutado bajo entornos Windows, Unix (Solaris, SunOs), Linux (Red Hat) e incluso existen versiones anteriores de esta distribución ejecutables para MS-DOS.

Para más información, véase <http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/>.

Neural Planner, NeuroDiet y Easy NN-plus

Neural Planner, EuroDiet y Easy NN-plus son tres herramientas gráficas comerciales ejecutables en plataformas Windows, diseñadas por Stephen Wolstenholme para trabajar con modelos de aprendizaje basados en redes neuronales.

- Neural Planner contempla dos modos de trabajo. Por un lado el usuario puede gráficamente especificar la topología de la red neuronal y por otro lado, ésta puede ser aprendida a partir de un archivo.
- NeuroDiet es un entorno que sigue la línea de funcionamiento del producto anterior pero especializado hacia el tratamiento de problemas dietéticos.
- Easy NN-plus extiende la funcionalidad de Neural Planner. El usuario dispone de más utilidades gráficas para poder controlar el seguimiento y la evaluación del modelo aprendido.

Para más información <http://www.tropheus.demon.co.uk>.

NeuroShell

Bajo el nombre de *NeuroShell* encontramos un conjunto de herramientas gráficas independientes (de las que destacamos *NeuroShell 2*, *NeuroShell Predictor*, *NeuroShell Classifier* y *NeuroShell Trader*) desarrolladas y comercializadas por Ward Systems Group para trabajar fundamentalmente con modelos de aprendizaje basados en redes neuronales. La primera de éstas, *Neuroshell 2*, es una herramienta de uso muy intuitivo pero cuya aplicación se restringe al ámbito académico. El resto de aplicaciones son más robustas y se utilizan para problemas más reales.

- *NeuroShell Predictor* se utiliza para la predicción de variables numéricas tales como índices de ventas, precios de mercado, costes, etc. Dicha herramienta se basa en el desarrollo de unas técnicas propias de redes neuronales, *TurboProp2* (TM2) y una variante del *General Regression Neural Net* (GRNN), y estimadores estadísticos guia-

dos por algoritmos genéticos. El entorno dispone de capacidades de representación gráfica de la información estadística asociada al modelo.

- *NeuroShell Classifier* optimiza las técnicas anteriores (TM2 y la variante de GRNN) con el fin de utilizar dichos modelos para tareas de clasificación. Por lo que se refiere a sus capacidades de representación gráfica y manejabilidad es prácticamente idéntico al anterior.
- *NeuroShell Trader*, añade respecto a los dos anteriores técnicas de lógica difusa e indicadores de agrupamiento enfocadas ambas hacia el reconocimiento de patrones.

Los sistemas están disponibles para plataformas Windows. Para más información, véase <http://www.wardsystems.com/products.asp>.

See5 / C5.0

See 5 / C5.0 es una herramienta de fácil manejo desarrollada y comercializada por la empresa *RuleQuest Research Pty Ltd* dirigida por Ross Quinlan. El entorno See5 / C5.0 se centra en la construcción de modelos de clasificación basados en árboles de decisión y conjuntos de reglas (es un heredero del popular método C4.5 de Quinlan) haciendo uso de la técnica de aprendizaje C5.0. Además permite la combinación de modelos mediante *boosting*.

La herramienta ha sido diseñada para operar sobre grandes volúmenes de datos. Aunque el entorno trabaja con un formato de archivos predefinido (*.data), se puede utilizar una herramienta complementaria (*ODBCHook*), desarrollada también por la misma compañía, que permite traducir fuentes de datos accesibles vía ODBC en archivos *.data. Los modelos aprendidos pueden ser exportados a código en C, con lo que pueden ser incorporados como parte de otros sistemas de aprendizaje más complejos o específicos.

El sistema está disponible para plataformas Windows, Solaris, Irix y Linux. Para más información consultese <http://www.rulequest.com/download.html>.

APÉNDICE [B]

DATOS DE EJEMPLO

A lo largo de todo el libro, se han ido introduciendo numerosos ejemplos con conjuntos de datos de distintos ámbitos y que se utilizaban, en cada momento, para ilustrar una u otra técnica. Ya que muchos de estos conjuntos de datos se utilizaban varias veces y varios capítulos, la descripción de los datos se ha realizado de una manera sencilla, sin pararse a puntualizar el origen y significado de los datos. En este apéndice vamos a enumerar los conjuntos de datos utilizados a lo largo del libro.

Gran parte de los conjuntos de datos (*datasets*) utilizados provienen del repositorio de la Universidad de California en Irvine (UCI) [Blake & Merz 1998], que se encuentra disponible en la siguiente dirección <http://www.ics.uci.edu/~mlearn/MLRepository.html>. Este repositorio se utiliza para ilustrar ejemplos en muchos textos y herramientas de aprendizaje automático y minería de datos y es, por tanto, un referente importante no sólo para los investigadores en minería de datos sino también para los usuarios, ya que pueden experimentar ellos mismos con los datos con los que han visto ilustrada una técnica o una herramienta. Éste es el mismo caso que ocurre con este libro. Animamos al lector a repetir (o mejorar) los experimentos incluidos a lo largo de este libro, utilizando para ello los siguientes conjuntos de datos:

Adult (Census Income)

Éste es un conjunto de datos extraído de la información sobre el censo de los Estados Unidos en 1994. Base de datos original del UCI (*Adult Database*, también conocida como *Census Income*). Cada ejemplo contiene 14 atributos con información sobre una persona: edad, nivel de estudios, sexo, país de nacimiento, etc. Cada ejemplo contiene además una clase que indica el nivel de ingresos anuales del individuo (mayor a 50.000 dólares, o

menor o igual a 50.000 dólares). Existen 48.842 ejemplos, de los cuales seis son contradictorios o duplicados.

Boston Housing

Este conjunto de datos proviene también del repositorio UCI (*Housing Database*). Contiene 14 atributos numéricos (en realidad uno de ellos es binario) incluyendo la clase y 506 instancias sobre las condiciones de viviendas en los suburbios de la ciudad de Boston. Los atributos contienen información sobre la zona en la cual se ubica la vivienda, entre los que se encuentran indicadores como por ejemplo la ratio de crímenes por habitantes (CRIM), la concentración de óxidos nítricos (NOX), el número medio de habitaciones por vivienda (RM), etc. El valor dependiente (a priori) es un atributo numérico que tiene relación con el precio de la vivienda. Este conjunto de datos es apropiado para tareas de regresión. El origen de este conjunto de datos es el trabajo [Harrison & Rubinfeld 1978].

German Credit

Este conjunto de datos contiene 1.000 ejemplos que representan clientes de una entidad bancaria que demandaron un crédito. Existen siete atributos numéricos y 13 nominales. Los atributos de cada objeto indican información sobre el cliente en cuestión: estado civil, edad, parado, etc., así como información del crédito: propósito del crédito, cantidad solicitada, etc. La clase es binaria e indica si el cliente puede ser considerado como fiable para concederle el crédito o no. Este conjunto de datos incluye matriz de costes, por lo que es especialmente indicado ilustrar aprendizaje sensible al coste y diseño de campañas de “marketing”. Se ha extraído del repositorio UCI del directorio “Statlog Databases” cedido por Ross D. King y Hans Hofmann.

Iris

El conjunto de datos “iris” (del repositorio UCI, *Iris Plant Database*) consiste en 150 ejemplos de lirios de tres tipos, con cuatro atributos numéricos (*sepallength*, *sepalwidth*, *petallength*, *petalwidth*) y un quinto nominal, que representa la clase o tipo de lirio (*setosa*, *versicolour*, *virginica*). Los atributos contienen información sobre la forma de la flor. Concretamente, la longitud y anchura del sépalo, y la longitud y anchura del pétalo. Este conjunto de datos es probablemente uno de los más utilizados en el área de reconocimiento de formas.

Isolet

Esta base de datos está compuesta por la pronunciación de letras por 150 personas diferentes. Para la recopilación de los datos, 150 sujetos pronunciaron el nombre de 26 letras del alfabeto dos veces. Por lo tanto, hay 52 ejemplos pertenecientes a cada hablante, un total de 7.800 ejemplos, de los cuales 203 han sido descartados por problemas en la grabación, con lo que el conjunto de datos está formado por 7.597 ejemplos. Cada ejemplo contiene parámetros numéricos con valores entre -1 y 1, conteniendo información sobre la letra pronunciada tales como: características sonoras, coeficientes espectrales, etc. Proviene del repositorio UCI (*Isolet Spoken Letter Recognition Database*).

Labor

El conjunto de datos “labor” (originario del UCI, *Labor relations Database*) recoge información (16 atributos: ocho nominales y ocho numéricos) sobre acuerdos sindicales, indicando si éstos fueron buenos o malos (la clase). El conjunto consta de 57 instancias. Los atributos indican condiciones del acuerdo: duración (dur), horas de trabajo (hours.hrs), días de vacaciones (*holidays*), etc. Proviene de las estadísticas sobre tratados laborales en Canadá durante los años 1987 y 1988.

Lenses

El *lenses dataset*, también originario del UCI, recoge ejemplos de adaptación de lentes de contacto. Es un conjunto muy sencillo: sólo contiene 24 ejemplos, con cuatro atributos, todos nominales: edad (joven, presbíptico, pre-presbíptico), prescripción (miopía, hipermetropía), astigmatismo (sí, no), producción-lágrimas (reducida, normal). La clase es el tipo correcto de lente (dura o blanda) o bien el uso de gafas (no).

SERVO

Conjunto de datos proveniente de un servomecanismo. Es también original del UCI (Servo data) donado por J.R. Quinlan. Los 167 datos sobre el servomecanismo incluyen dos atributos numéricos referentes al amplificador del mecanismo, un atributo nominal sobre el motor y otro atributo nominal con información acerca de la posición de una tuerca. La clase es numérica e indica el “tiempo de subida” del servo mecanismo o el tiempo requerido por el mecanismo para responder a un cambio de posición.

Wisconsin Diagnostic Breast Cancer (wdbc)

Este conjunto recoge información sobre 569 exploraciones de mamografías para determinar la existencia de cáncer de mama. El conjunto está formado por datos que contienen una serie de diez características medidas en los núcleos de células: textura, perímetro, área, simetría, etc. Estas características se han capturado desde imágenes digitalizadas de muestras del tejido orgánico. Para cada característica se ha anotado su valor medio en las células de la muestra, su desviación tipo y el caso más desfavorable, configurando en total 30 variables numéricas disponibles para la predicción. Cada ejemplo contiene una clase que indica el carácter benigno o maligno del tumor. Además, existe un atributo identificador del ejemplo. Este conjunto de datos proviene del UCI (*Wisconsin Breast Cancer Databases*, concretamente el conjunto wdbc).

Tabla resumen

A continuación, mostramos una tabla resumen de todos estos conjuntos de datos, con el nombre del conjunto de datos, el número de instancias, el número de atributos numéricos y nominales, el tipo del atributo salida (en el caso de que haya un atributo dependiente especial), si existen atributos faltantes y si se proporciona una matriz de coste con el ejemplo.

Nombre	Número de Instancias	Atributos numéricos	Atributos nominales	Tipo de la salida (clases)	Atributos faltantes	Matriz de coste
Adult	48.842	6	8	Nominal (2)	Sí	No
Boston Housing	506	13	0	Numérico	No	No
German Credit	1.000	7	17	Nominal (2)	No	Sí
Iris	105	4	0	Nominal (3)	No	No
Isolet	7597	0	617	Nominal (26)	No	No
Labor	57	8	8	Nominal (2)	Sí	No
Lenses	24	0	4	Nominal (3)	No	No
Servo	167	2	2	Numérico	No	No
wdbc	506	30	0	Nominal (2)	No	No

REFERENCIAS BIBLIOGRÁFICAS

- [Aamodt & Plaza 1994] Aamodt, A.; Plaza, E. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". *AI Communications*, 7 (1), 39-52, 1994.
- [Abiteboul et al. 1997] Abiteboul, S.; Quass, D.; McHugh, J.; Widom, J.; Wiener, J.L. "The Lore query language for semistructured data", *International Journal on Digital Libraries*, 1(1): 68-88, 1997.
- [ACM 2003] Special Issue on Multi-Relational Data Mining, SIGKDD Explorations, 5(1), ACM, 2003.
- [Adano 2001] Adano, J.M. *Data Mining for Association Rules and Sequential Patterns*, Springer-Verlag, 2001.
- [Agrawal & Srikant 1995] Agrawal, R.; Srikant "Mining Sequential Patterns" in *Proceedings of the Eleventh International Conference on Data Engineering*, pp. 3-14, 1995.
- [Agrawal & Srikant 1996] Agrawal, R.; Srikant "Mining Sequential Patterns: Generalizations and Performance Improvements" in *Proceedings of the Eleventh International Conference on Extending Database Technology*, Lecture Notes in Computer Science, Springer Verlag, Vol 1057, pp. 3-17, 1996.
- [Aguilera et al. 2003] Aguilera, J.J.; del Jesus, M.J.; González, P.; Herrera, F.; Navío, M.; Sáinz, J. "Extracción Evolutiva de Reglas de Asociación en un Servicio de Urgencias Psiquiátricas". Actas del *II Congreso español sobre Metaheurísticas, Algoritmos evolutivos y bioinspirados*: 548-555, 2003.
- [Aha et al. 1991] Aha, D.; Kibler, D.; Albert, M. "Instance-based learning algorithms". *Machine Learning*, 6, 37-66, 1991.
- [Aitkin et al. 1989] Aitkin, M.; Anderson, D.; Francis, B.; Hinde, J. *Statistical Modelling in GLIM*, Oxford University Press, 1989.
- [Allwein et al. 2000] Allwein, E.; Schapire, R.; Singer, Y. "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers". *Journal of Machine Learning Research*, 1(Dec):113-141, 2000.
- [Allwein et al. 2000] Allwein, E.L.; Schapire, R.E.; Singer, Y. "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers" *Journal of Machine Learning Research*, 1:113-141, 2000.
- [Aluja & Morineau 1999] Aluja Banet, Tomàs; Morineau, Alain; *Aprender de los datos: el análisis de componentes principales*, Barcelona: EUB, 1999.
- [Andrienko & Andrienko 1999] Andrienko, G.L.; Andrienko, N.V. "Interactive Maps for Visual Data Exploration" *International Journal of Geographical Information Science*, vol. 13 (4), pp.355-374. 1999.
- [Angluin 1988] Angluin, D. "Queries and concept learning" *Machine Learning*, 2(4): 319-342, 1988.
- [Apt 1997] Apt, K.R. *From Logic Programming to Prolog*, Prentice Hall, 1997.
- [Araujo et al. 2000] Araujo, D.L.A.; Lopes H.S.; Freitas, A.A."Rule discovery with a parallel genetic algorithm" in Freitas A.A, Hart, W.; Krashogor, N.; Smith, J. (Eds.) *Data Mining with Evolutionary Algorithms*: 89-94, 2000.
- [Au & Chan 1998] Au, W.H.; Chan, K.C.C. "An effective algorithm for discovering fuzzy rules in relational databases". In *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ IEEE'98)*: 1314-1319, 1998.
- [Au & Chan 1999] Au, W.; Chan, K.C.C. "FARM: A Data Mining System for Discovering Fuzzy Association Rules" In *Proceedings of the 8th IEEE International Conference on Fuzzy Systems*: 1217-1222, 1999.
- [Bäck et al. 1997] Bäck, T.; Fogel, D.; Michalewicz, Z. *Handbook of Evolutionary Computation*, Oxford University Press, 1997.

- [Baeza-Yates & Ribeiro-Neto 1999] Baeza-Yates, R.; Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [Bakiri & Dietterich 2002] Bakiri, G.; Dietterich T.G. "Achieving high-accuracy text-to-speech with machine learning" in *Data Mining Techniques in Speech Synthesis*. Chapman and Hall. 2002.
- [Bala et al. 1996] Bala, J.; De Jong, K.; Huang, J.; Vafaie, H.; Wechsler, H. "Using Learning to Facilitate the Evolution of Features for Recognizing Visual Concepts", *Evolutionary Computation* 4(3): 297-311, 1996.
- [Barnett & Lewis 1994] Barnett, V.; Lewis, T.; *Outliers in Statistical Data*, John Wiley and Sons, 1994.
- [Basu et al. 1998] Basu, C.; Haym, H.; Cohen, W.W. "Recommendation as Classification: Using social and content-based information in recommendation", in Proceedings of the Fifteenth National Conference on Artificial Intelligence, pp. 714-720, 1998.
- [Battacharyya et al. 1998] Battacharyya, S.; Zumbach, G.; Olivier, P. "Representational semantics for genetic programming based learning in high-frequency financial data" in *Proc. 3rd Annual Conf on Genetic Programming 1998*: 11- 16, 1998.
- [Bauer & Kohavi 1999] Bauer, E.; Kohavi R. "An Empirical Comparison of Voting Classification Algorithms: {B}agging, Boosting, and Variants" *Machine Learning*, Vol 36(1), pp 105-139, 1999.
- [Beasly et al. 1993] Beasly, D.; Bull, D.R.; Martin R.R. "A sequential niche technique for multimodal function optimization", *Evolutionary Computation* 1: 101-125, 1993.
- [Becher et al. 2000] Becher, J.D.; Berkhin, P.; Freeman, E. "Automating exploratory data analysis for efficient data mining" *Knowledge Discovery and Data Mining*, 424-429, 2000.
- [Beckmann et al. 1990] Beckmann, N.; Kriegel, H.P.; Schneider, R.; Seeger, B. "The R*-tree: An efficient and robust access method for points and rectangles". In *Proc. ACM SIGMOD Conf. on Management Data*, 322-331, 1990.
- [Ben-Hur et al. 2001] Ben-Hur, A.; Horn, D.; Siegelmann, H. T.; Vapnik, V. "Support Vector Clustering". *Journal of Machine Learning Research*, 2 (Dec):125-137, 2001.
- [Berberian 1961] Berberian, S. K. *Introduction to Hilbert Space*, Oxford University Press, 1961.
- [Bergadano & Gunetti 1993] Bergadano, F.; Gunetti, D. "An Interactive System to Learn Functional Logic Programs", in *Proceedings of the 13th International Conference on Artificial Intelligence*, pp. 1044-1049, 1993.
- [Bergadano & Gunetti 1995] Bergadano, F.; Gunetti, D. *Inductive Logic Programming: from Machine Learning to Software Engineering*, The MIT Press, 1995.
- [Berry & Linoff 2000] Berry, M. J. A.; Linoff, G. S. *Mastering Data Mining: The Art and Science of Customer Relationship Management*, Wiley Computer Publishing, 2000
- [Berthold & Hand 2003] Berthold, M.; Hand, D.J. (eds.) *Intelligent Data Analysis. An Introduction*, Springer, 2nd Edition, 2003.
- [Billsus & Pazzani 1999] Billsus, D.; Pazzani, M. "A hybrid user model for news story classification" in *Proceedings of the 7th International Conference on User Modelling*, pp.99-108, 1999.
- [Blake & Merz 1998] Blake, C. L.; Merz, C.J. "UCI Repository of machine learning databases" *University of California, Irvine, Dept. of Information and Computer Sciences*, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, 1998.
- [Blockeel & De Raedt 1996] Blockeel, H.; De Raedt, L. "Inductive Database Design", in *Proceedings of the International Syposium on Methodologies for Intelligent Systems*, pp: 376-385, 1996.
- [Blockeel & De Raedt 1998] Blockeel, H.; De Raedt, L. "Top-down induction of first order logical decision trees", *Artificial Intelligence*, 101: 285-297, 1998.
- [Blum & Mitchell 1998] Blum, A.; Mitchell, T. M. "Combining Labeled and Unlabeled Sata with Co-Training". *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 92-100, 1998.

- [Bojarczuk et al. 2000] Bojarczuk, C.C.; Lopes, H.S. Freitas, A.A. "Genetic programming for knowledge discovery in chest pain diagnosis", *IEEE Engineering in Medicine and Biology Magazine- special issue on data mining and knowledge discovery* 19(4): 38-44, 2000.
- [Borges & Levene 2000] Borges, J.; Levene, M. "A Fine Grained Heuristic to Capture Web Navigation Patterns", *SIGKDD Explorations*, 2(1): 1-11, 2000.
- [Bosc et al. 1999] Bosc, P. Pivert, O.; Ughetto, L. "Database mining for the discovery of extended functional dependencies", In *Proceedings of the NAFIPS'99*: 580-584, 1999.
- [Boser et al. 1992] Boser, B.; Guyon, I.; Vapnik, V. "A Training Algorithm for Optimal Margin Classifiers". In *Proceedings of the 5th Annual ACM W4orkshop on Computational Learning Theory, COLT*, 1992.
- [Bowman & Azzalini 1997] Bowman, A. W.; Azzalini, A. *Applied Smoothing Techniques for Data Analysis*, Oxford University Press, 1997.
- [Box et al. 1997] Box, G. E. P.; Jenkins, G. M.; Reinsel, G. C. *Time Series Analysis, Forecasting and Control* 3rd ed. Prentice Hall, Englewood Cliffs, NJ.1997
- [Bradford et al. 1998] Bradford, J.; Kunz, C.; Kohavi, R.; Brunk, C.; Brodley, C. "Pruning decision trees with misclassification costs", in *H.Prade (ed.) Proceedings of the European Conference on Machine Learning*, pp. 131-136, 1998.
- [Brandt et al. 2001] Brandt, N.; Brockhausen, P.; de Haas, M.; Kietz, J.-U.; Knobbe, A.; Rem, O.; Zücker, R. *Mining Multi-relational Data*, Deliverable D15, 2001.
- [Breiman 1996] Breiman, L. "Bagging Predictors" *Machine Learning*, Vol 24(2), pp 123-146, 1996.
- [Breiman 1999] Breiman, L. "Combining predictors", in A. J. C. Sharkey, editor, *Combining Artificial Neural Nets*, pages 31-50. Springer-Verlag, 1999.
- [Breiman et al. 1984] Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*, Wadsworth and Brooks/Cole, Monterey, 1984.
- [Brodley & Utgoff 1995] Brodley, C. E.; Utgoff, P. E. "Multivariate decision trees" *Machine Learning*, Vol 19, pp 45-77, 1995.
- [Brown et al. 1994] Brown, C.M.; Danzig, P.B.; Hardy, D.; Manber, U.; Schwartz, F. "The Harvest information discovery and access system", in *Proceedings of the Second International World Wide Web Conference*, pp. 763-771, 1994.
- [Buntine 1992] Buntine, W. "Learning Classification Trees" *Statistics and Computing*, 2:63-73, 1992.
- [Buntine 1994] Buntine, W. "Operations for learning with graphical models". *Journal of Artificial Intelligence Research*. 2:159–225. 1994.
- [Burges 1998] Burges, C. "A Tutorial on Support Vector Machines for Pattern Recognition". *Knowledge Discovery and Data Mining*, 2:2, 1998.
- [Burl et al. 1998] Burl, M.C.; Asker, L.; Smyth, P.; Fayyad, U.M.; Perona, P.; Crumpler, L.; Aubele, J.: "Learning to Recognize Volcanoes on Venus". *Machine Learning* 30(2-3): 165-194,1998.
- [Cano et al. 2003] Cano, J.R.; Herrera, F.; Lozano, M. "Using evolutionary algorithms as instante selection for data reduction in KDD: An experimental study", to appear in: *IEEE Transactions on Evolutionary Computation*, 2003.
- [Cardie 1997] Cardie, C. "Empirical methods in information extraction", *AI Magazine*, 18(4): 65-79, 1997.
- [Castillo et al. 1998] Castillo, E.; Gutierrez, J.M; Hadi, A. *Sistemas Expertos y modelos de Redes Probabilísticas*. Monografías de la Academia de Ingeniería. 1998.
- [Cauwenberghs & Poggio 2001] Cauwenberghs, G.; Poggio, T. "Incremental and Decremental Support Vector Machine Learning" in *Advances in Neural Information Processing Systems (NIPS*2000)*, volume 13, Vancouver, Canada, 2001.
- [Celma et al. 2003] Celma, M.; Casamayor, J.C.; Mota, L. *Bases de Datos Relacionales*, Pearson Educación, 2003.
- [Cestnik et al. 1987] Cestnik, G.; Kononenko, I.; Bratko, I. "Assistant 86: A Knowledge Acquisition Tool for Sophisticated Users" in Bratko, I.; Lavrac, N. (Eds.) *Progress in Machine Learning*, Sigma Press, 1987.

- [Chakrabarti 2000] Chakrabarti, S. "Data mining for hypertext: A tutorial survey", *SIGKDD Explorations*, 1(2): 1-11, 2000.
- [Chakrabarti 2003] Chakrabarti, S. *Mining the Web*, Morgan Kaufmann, 2003.
- [Chakrabarti et al. 1998] Chakrabarti, S.; Dom, B.; Gibson, D.; Kleinberg, J.; Raghavan, P.; Rajagopalan, S. "Automatic resource compilation by analyzing hyperlink structure and associated text", in *Proceedings of the 7th international World Wide Web Conference*, Computer Networks vol. 30, pp. 65-74, 1998.
- [Chambers & Hastie 1993] Chambers, J.M.; Hastie, T.J. (eds.) *Statistical Models in S*, Chapman & Hall, 1993.
- [Chan & Stolfo 1993] Chan, P.; Stolfo, S. "Experiments on multistrategy learning by meta-learning", in *Proceedings of the Second International Conference on Information Knowledge Management*, pp. 314-323, 1993.
- [Charles & Fyfe 1998] Charles D.; Fyfe C. "Modelling Multiple Cause Structure using Rectification Constraints". *Network: Computation in Neural Systems*, 9:167-182, May 1998.
- [Cheeseman & Stutz 1996] Cheeseman, P.; Stutz, J. "Bayesian classification (AUTOCLASS): Theory and results" In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. 1995.
- [Chen & Wei 2002] Chen, G.; Wei, Q. "Fuzzy association rules and the extended mining algorithms", *Information Sciences* 147: 201-228, 2002.
- [Chen 1969] Chen, C.H. A theory of Bayesian learning systems. *IEEE Trans. Sys. Sci. Cyb*, SSC-5, 30-37, January, 1969.
- [Chen et al. 1998] Chen, M.-S.; Park, J.S.; Yu, P.S. "Efficient data mining for traversal patterns", *IEEE Transactions on Knowledge and Data Engineering*, 10(2): 209-221, 1998.
- [Cheung et al. 1996] Cheung, D. W.; Ng, V.T.; Fu, A.W.; Fu, Y.J. "Efficient mining of association rules in distributed databases", *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, p. 911-922, 1996.
- [Chickering et al. 1996] Chickering, M.; Geiger, D.; Heckerman, D. "Learning bayesian networks in NP-Complete". In *Learning from data: Artificial and Statistics V* (Fischer, D. and Lenk, H. Eds). Springer-Verlag, pp 121—130. 1996.
- [Cho & Wüthrich 1998] Cho, V.; Wüthrich, B. "Toward real time discovery from distributed information sources", in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 376-397, 1998.
- [Chow & Liu 1968] Chow, C.K.; Liu, C.N. "Approximating discrete probability distributions with dependence trees". *IEEE Transactions on Information Theory*. 14:462-467. 1968.
- [Clark & Boswell 1991] Clark, P.; Boswell, R. "Rule induction with CN2: Some recent improvements" in *Proceedings of the Fifth European Working Session on Learning*, pp. 151-163, Springer, Berlin 1991.
- [Clark & Niblett 1989] Clark, P.; Niblett, T. "The CN2 induction algorithm" *Machine Learning*, 3(4): 261-283, 1989.
- [Cleary & Trigg 1995] Cleary, J.G.; Trigg, L.E. "K*: An Instance- based Learner Using an Entropic Distance Measure", *Proceedings of the 12th International Conference on Machine learning*, pp. 108-114, 1995.
- [Cleveland 1993] Cleveland, W. *Visualizing Data*, Hobart Press, 1993.
- [Cohen 1993] Cohen, W. "Efficient pruning methods for separate-and-conquer rule learning systems" in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 988-994, Morgan Kaufmann, 1993.
- [Cohen 1995a] Cohen, W.W. "Fast effective rule induction" Proc. of the Twelfth International Conference on Machine Learning, pages 115-123, 1995.
- [Cohen 1995b] Cohen, W.W. "Text Categorization and Relational Learning", in *Proceedings of the International Conference on Machine Learning*, pp. 124-132 , 1995.
- [Collins & Duffy 2002] Collins, M.; Duffy, N. "New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron" in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL-2002*, Philadelphia, PA, 2002.

- [Cook & Holder 2000] Cook, D.J.; Holder, L.B. "Graph-based data mining", *IEEE Intelligence Systems*, 15(2): 32-41, 2000.
- [Cook & Yin 2001] Cook, R. D.; Yin, X. "Dimension reduction and visualization in discriminant analysis (with discussion)". *Aust. N. Z. J. Stat.* 43(2): 147-199, 2001.
- [Cooper et al. 1992] Cooper, G.; Herskovits, E. "A bayesian method for the induction of probabilistic networks from data". *Machine Learning*. 9(4): 309–348. 1992.
- [Corchado & Fyfe 2002a] Corchado, E.; Fyfe, C. "The Scale Invariant Map and Maximum Likelihood Hebbian Learning". *Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies*. KES 2002. Edited by E. Damianai et al. IOS Press. 2002.
- [Corchado & Fyfe 2002b] Corchado, E.; Fyfe, C. "Optimal Projections of High Dimensional Data". ICDM '02. The 2002 IEEE International Conference on Data Mining. 2002
- [Corchado et al. 2002] Corchado, E.; Koetsier, J.; MacDonald, D.; Fyfe, C. "Classification and ICA using Maximum Likelihood Hebbian Learning". IEEE International Workshop on Neural Networks for Signal Processing, Switzerland, 2002.
- [Corchado et al. 2003] Corchado, E.; MacDonald, D. Fyfe, C. "Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit". Data Mining and Knowledge Discovery. 2003.
- [Cordón et al. 1998] Cordón, O.; del Jesus, M.J.; Herrera, F. "Genetic Learning of Fuzzy Rule-based Classification Systems Co-operating with Fuzzy Reasoning Methods". *International Journal of Intelligent Systems* 13 (10/11): 1025-1053. 1998.
- [Cordón et al. 2001] Cordón, O.; Herrera, F.; Hoffmann, F.; Magdalena, L. *Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases*. World Scientific, 2001.
- [Cortes & Vapnik 1995] Cortes, C.; Vapnik, V. "Support Vector Networks". *Machine Learning*, 20(3):273-297, 1995.
- [Cost & Salzberg 1993] Cost, S.; Salzberg, S. "A weighted nearest neighbor algorithm for learning with symbolic features", *Machine Learning*, 10, 57-78, 1993.
- [Cover & Hart 1967] Cover, T. M.; Hart, P. E. "Nearest neighbor pattern classification" *IEEE Trans. Info. Theory*, IT-13, 21-27, January, 1967.
- [Craven 1996] Craven M. W. "Extracting Comprehensible Models from Trained Neural networks" PhD Thesis, Dept. of Computer Science, University of Wisconsin-Madison, 1996.
- [Craven et al. 1998] Craven, M.; Slattery, S.; Nigam, K. "First-order Learning for Web Mining", in *Proceedings of the 10th European Conference on Machine Learning*, pp. 250-255, 1998.
- [Craven et al. 2000] Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T. M.; Nigam, K.; Slattery, S. "Learning to construct knowledge bases from the World Wide Web" *Artificial Intelligence* 118(1-2): 69-113, 2000.
- [Cristianini & Shawe-Taylor 2000] Cristianini, N.; Shawe-Taylor, J. *An introduction to Support Vector Machines and Other Kernel-Based Methods*. Cambridge University Press, 2000.
- [Croft 1993] Croft W.B. "Knowledge-Based and statistical approaches to text retrieval". *IEEE Expert*. 8-11, april 1993.
- [De Jong et al. 1993] De Jong, K.A.; Spears, W.M.; Gordon, D.F. "Using genetic algorithms for concept learning", *Machine Learning*: 161-188, 1993.
- [De Raedt & Bruynooghe 1993] De Raedt, L.; Bruynooghe, M. "A theory of clausal discovery", in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1058-1063, 1993.
- [De Raedt & Kersting 2003] De Raedt, L.; Kramer, S. "Probabilistic Logic Learning", in [ACM 2003], pp: 31-48, 2003.
- [De Raedt & Kramer 2001] De Raedt, L.; Kramer, S. "The levelwise version space algorithm and its application to molecular fragment finding", in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, vol. 2, pp. 853-859, 2001.

- [De Raedt 1998a] De Raedt, L. "An inductive logic programming query language for database mining", Proceedings of the 4th International Conference on Artificial Intelligence and Symbolic Computation (J. Calmet and J. Plaza, eds.), LNAI, vol. 1476, Springer, 1998, pp. 1-13.
- [De Raedt 1998b] De Raedt, L. "Attribute-value learning versus inductive logic programming: the missing links (extended abstract)", in *Proceedings of the Eighth International Conference on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence 1446, pp. 1-8, 1998.
- [De Raedt 2000] De Raedt, L. "A Logical Database Mining Query Language", in Proceedings of the 10th International Conference on Inductive Logic Programming, Lecture Notes in Artificial Intelligence, vol. 1866, pp: 78-92, 2000.
- [Deb 2001] Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.
- [Deerwester et al. 1990] Deerwester, S.; Dumais, S.; Furnas, G.; Landauer, T.; Harshman, R. "Indexing by latent semantic analysis", *Journal of the American Society for Information Science*, 41(6): 391-407, 1990.
- [Dehaspe & Toivonen 1999] Dehaspe, L.; Toivonen, H. "Discovery of frequent datalog patterns", *Data Mining and Knowledge Discovery*, 3(1): 7-36, 1999.
- [Dehaspe & Toivonen 2001] Dehaspe, L.; Toivonen, H. "Discovery of Relational Association Rules", en [Džeroski & Lavrač 2001], pp: 189-212, 2001.
- [Dehaspe et al. 1994] Dehaspe, L.; Van Laer, W.; De Raedt, L. "Applications of a Logical Discovery Engine", in *Proceedings of the Fourth International Workshop on Inductive Logic Programming*, GMD-Studien, vol. 237, pp. 291-304, 1994.
- [DeJong & Mooney 1986] DeJong, G. F.; Mooney, R. "Explanation-based learning: An alternative view". *Machine Learning*, 1(2):145-176, 1986.
- [Dempster et al. 1977] Dempster, A.; Laird, N.; Rubin, D. "Maximun Likelihood from incomplete data via EM algorithm". *Journal of the Royal Statistical Society B*. 39:1–38. 1977.
- [Dhar & Stein 1997] Dhar, V.; Stein, R.M. *Seven Methods for Transforming Corporate Data Into Business Intelligence*, Prentice-Hall 1997.
- [Diaconis & Freedman 1984] Diaconis, P.; Freedman, D. "Asymptotics of Graphical Projections". *The Annals of Statistics*, 12(3):793-815, 1984.
- [Dietterich & Bakiri 1995] Dietterich, T.G.; Bakiri G. "Solving multiclass learning problems via error-correcting output codes" *Journal of Artificial Intelligence Research*, Vol 2, pp 263-286, 1995.
- [Dietterich 1998] Dietterich T. G. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms", *Neural Computation*, 10 (7) 1895-1924. 1998.
- [Dietterich 2000a] Dietterich, T.G., "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization" *Machine Learning*, Vol 40(2), pp 139-157, 2000.
- [Dietterich 2000b] Dietterich, T.G. "Ensemble Methods in Machine Learning" in *Proceedings of the First International Workshop on Multiple Classifier Systems*. pp. 1-15. Lecture Notes in Computer Science, 2000.
- [Dietterich 2002] Dietterich, T.G. "Machine Learning for Sequential Data: A Review" In *Proc. of Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops SSPR SSPR/SPR 2002*: 15-30, 2002.
- [Diligenti et al. 2003] Diligenti, M.; Frasconi, P.; Gori, M. "Hidden Tree Markov Models for Document Image Classification" *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(4): 519-523. 2003.
- [Dobson 1991] Dobson, A.J. *An introduction to generalized linear models*. Chapman & Hall, 1991.
- [Domingos 1997] Domingos, P. "Knowledge Acquisition from Examples Via Multiple Models" En *Proceedings of the 1997 International Conference on Machine Learning*, IOS Press, Morgan Kaufmann Publishers, pp. 98-106, 1997.

- [Domingos 1999] Domingos, P. "MetaCost: A General Method for Making Classifiers Cost-Sensitive" En *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, 1999.
- [Domingos 2000] Domingos, P. "Bayesian Averaging of Classifiers and the Overfitting Problem", in *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 223-230), Morgan Kaufmann. 2000.
- [Drummond & Holte 2000] Drummond, C.; Holte, R.C. "Explicetly Representing Expected Cost: An Alternative to ROC representation", *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (KDD'00), pp. 198-207, 2000.
- [Duda & Hart 1973] Duda, R.O.; Hart, P.E. *Pattern classification and scene analysis*. John Wiley and Sons. 1973.
- [Duda et al. 2001] Duda, R.; Hart, P.E.; Stork D.G. *Pattern Classification* (2nd edition), Wiley, New York, 2001.
- [Dumais et al. 1998] Dumais, S.T.; Platt, J.; Hecherman, D.; Sahami, M. "Inductive Learning Algorithms and Representations for Text Categorization", in *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management*, pp: 148-155, 1998.
- [Dunham 2003] Dunham, M. H. *Data Mining. Introductory and Advanced Topics*, Prentice Hall, 2003.
- [Džeroski & Lavrač 2001] Džeroski, S.; Lavrač, N. *Relational Data Mining*, Springer Verlag, 2001.
- [Džeroski 1996] Džeroski, S. "Inductive Logic Programming and Knowledge Discovery in Databases", in *Advances in Knowledge Discovery and Data Mining*, pp: 117-152, AAAI Press/The MIT Press, 1996.
- [Džeroski 2003] Džeroski, S. "Multi-Relational Data Mining: An introduction", in [ACM 2003], pp: 1-16, 2003.
- [Eggermont et al. 1999] Eggermont, J.; Eiben, A.E.; van Hemert, J.I. "A comparison of genetic programming variants for data classification". In *Proceedings of Intelligent Data Analysis (IDA-99)*. 1999.
- [Elvira Consortium 2002] Elvira Consortium. "ELVIRA: an environment for creating and using probabilistic graphical models". In *Proceedings of the First European Workshop on Probabilistic Graphical Models*. Pp. 222-230, 2002.
- [Emde & Wettschereck 1996] Emde, W.; Wettschereck, D. "Relational Instance Based Learning", in *Proceedings 13th International Conference on Machine Learning*, pp: 22 - 130, 1996.
- [Esposito et al. 1997] Esposito, F.; Malerba, D.; Semeraro, G. "A Comparative Analysis of Methods for Pruning Decision Trees". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 5, pp. 476-491, 1997.
- [Ester et al. 1996] Ester M.; Kriegel H.-P.; Sander J. and Xu X. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, 226-231, 1996.
- [Ester et al. 1998] Ester, M.; Frommelt, A.; Kriegel, H.P.; Sander J. "Algorithms for Characterization and Trend Detection in Spatial Databases". *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 1998: 44-50*, 1998.
- [Ester et al. 2000] Ester, M.; Frommelt, A.; Kriegel, H.P.; Sander J. "Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support" *In Data Mining and Knowledge Discovery 4(2/3): 193-216*, 2000.
- [Estruch et al. 2003a] Estruch, V; Ferri, C.; Hernández-Orallo, J.; Ramírez-Quintana, M.J. "Simple Mimetic Classifiers" *Machine Learning and Data Mining in Pattern Recognition, Third International Conference, MLDM 156-171*. 2003.
- [Estruch et al. 2003b] Estruch, V; Ferri, C.; Hernández-Orallo, J.; Ramírez-Quintana, M.J. "Beam Search Extraction and Forgetting Strategies on Shared Ensembles" in *Proceedings of the Fourth International Workshop on Multiple Classifier Systems*. pp. 1-15. Lecture Notes in Computer Science, 2003.

- [Etzioni 1996] Etzioni, O. "The World Wide Web: quagmire or gold mine?" *Communications of the ACM*, 39(11): 65-68, 1996.
- [Eubank 1999] Eubank, R. L. *Nonparametric Regression and Spline Smoothing*, Marcel-Decker, 1999.
- [Fan & Gijbels 1996] Fan, J.; Gijbels, I. *Local polynomial modeling and its applications*, Chapman & Hall, London, 1996.
- [Fawcett & Provost 1997] Fawcett, T; Provost, F.J. "Adaptive Fraud Detection" *Data Mining and Knowledge Discovery* 1(3): 291-316 1997.
- [Fayyad & Irani 1993] Fayyad, U.M.; Irani, K.B. "Multi-interval discretization of continuous-valued attributes for classification learning". In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.
- [Fayyad 1994] Fayyad, U.M. "Branching on Attribute Values in Decision Tree Generation". In Proc. of the Twelfth National Conference on Artificial Intelligence AAAI-94, pages 601--606, Cambridge, MA, MIT Press. 1994.
- [Fayyad 1995] Fayyad, U.M. "SKICAT: Sky Image Cataloging and Analysis Tool" In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, IJCAI95, 2067-2068, 1995.
- [Fayyad et al. 1996b] Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P. "The KDD Process for Extracting Useful Knowledge from Volumes of Data", *Communications of the ACM*, 39(11): 27-34, 1996.
- [Fayyad et al. 1996a] Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P. "From Data Mining to Knowledge Discovery: An Overview" *Advances in Knowledge Discovery and Data Mining*, pp:1-34, AAAI/MIT Press 1996.
- [Fayyad et al. 2002] Fayyad, U.M.; Grinstein, G.G.; Wierse, A. (eds.) "Information Visualization in Data Mining and Knowledge Discovery" Morgan Kaufmann 2002.
- [Ferri et al. 2000] Ferri, C.; Hernández-Orallo, J.; Ramírez-Quintana, M.J. "Learning Functional Logic Classification Concepts from Databases", in *Proceedings of the Ninth International Workshop on Functional and Logic Programming*, pp. 296-308, 2000.
- [Ferri et al. 2001] Ferri, C.; Hernández-Orallo, J.; Ramírez-Quintana, M.J. "Learning MDL-guided Decision Trees for Constructor-Based Languages" in 11th International Conference on Inductive Logic Programming (ILP'2001), Work in Progress Track, Research Report 2001/12, LSIIT, 64 Bld Brant, 67400 Illkirch, France, 2001.
- [Ferri et al. 2002] Ferri, C.; Flach, P.; Hernández-Orallo, J. "Learning Decision Trees Using the Area Under the ROC Curve" in C. Sammut; A. Hoffman (eds.) The 2002 International Conference on Machine Learning, IOS Press, Morgan Kaufmann Publishers, ISBN: 1-55860-873-7, pp. 139-146, 2002.
- [Ferri et al. 2003a] Ferri, C.; Hernández-Orallo, J.; Salido, M. "Volume Under the ROC Surface for Multi-class Problems" in *The 2003 European Conference on Machine Learning, ECML'03, LNAI, Springer Verlag*, 2003.
- [Ferri et al. 2003b] Ferri, C.; Flach, P.; Hernández-Orallo, J. "Improving the AUC of Probabilistic Estimation Trees" in *The 2003 European Conference on Machine Learning, ECML'03, LNAI, Springer Verlag*, 2003.
- [Fisher 1936] Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7, Part II, 179-188, 1936 / *Contributions to Mathematical Statistics*. John Wiley: New York, 1959.
- [Fisher 1987] Fisher, D. "Knowledge Acquisition Via Incremental Conceptual Clustering" *Machine Learning*, 2, 139-172. 1987, reimpresso en Readings in Machine Learning, J. Shavlik ; T. Dietterich (Eds.), 267-283, Morgan Kaufmann, 1990.
- [Fisher et al. 1995] Fischer, S.; Lienhart, R.; Effelsberg, W. "Automatic Recognition of Film Genres" *Proc. ACM Multimedia 95*, pp. 295-304. 1995.
- [Flach & Lachiche 2001] Flach, P.; Lachiche, N. "Confirmation-guided discovery of first-order rules with Tertius", *Machine Learning*, 42 (1/2): 61-95, 2001.
- [Flener 2002] Flener, P. "Achievements and Prospects of Program Synthesis". Computational Logic: Logic Programming and Beyond, 310-346, 2002.

- [Flockart & Radcliffe 1995] Flockart, I.W. Y Radcliffe, N.J. "GA-Miner: Parallel data mining with hierarchical genetic algorithms". Final Report, EPCC-AIKMS-GA-Miner-Report 1.0, University of Edinburgh, November 1995.
- [Fogel 1988] Fogel, D.B. "An evolutionary approach to the travelling salesman problem" *Biological Cybernetics* 60(2): 139-144, 1988.
- [Fox 1991] Fox, J. *Regression Diagnostics*, Sage University Paper, Newbury Park, 1991.
- [Frank et al. 1999] Frank, E.; Paynter, G.W.; Witten, I.H. "Domain-Specific Keyphrase Extraction", in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 668-673, 1999.
- [Freitas & Lavington 1998] Freitas, A.A.; Lavington, S.H. *Mining very large databases with parallel processing*. Kluwer 1998.
- [Freitas 1999] Freitas, A.A. "On Rule Interestingness Measures" *Knowledge-Based Systems*, 12(5-6):309-315, 1999.
- [Freitas 2000] Freitas, A.A. "Understanding the Crucial Differences Between Classification and Discovery of Association Rules-A Position Paper", *ACM SIGKDD Explorations* 2 (1): 65-69, 2000.
- [Freitas 2002] Freitas, A. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Natural Computing Series, Springer-Verlag, Berlin 2002.
- [Freund & Schapire 1996] Freund, Y.; Schapire R.E. "Experiments with a new Boosting algorithm" in *Proceedings Proc. 13th International Conference on Machine Learning*, ICML'96, pp 148-156, 1996.
- [Friedman 1987] Friedman, J. H. "Exploratory projection pursuit". *Journal of the American Statistical Association*, 82(397):249-266, March 1987.
- [Friedman 1998] Friedman, N. "The bayesian structural EM algorithm". In *Proceedings of the 14th International Conference on Uncertainty in Artificial Intelligence*. pp 129—138. 1998.
- [Friedman et al. 1997] Friedman, N.; Geiger, D.; Goldszmidt, M. "bayesian networks classifiers". *Machine Learning*, 29:131-163. 1997.
- [Friedman et al. 1999] Friedman, N.; Getoor, L.; Koller, D.; Pfeffer, A. "Learning probabilistic relational models", in *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence*, pp. 1300-1309, 1999.
- [Fu et al. 1998] Fu, A.W.-C.; Wong, M.H.; Sze, S.C.; Wong W.C.; Wong W.L.; Yu, W.K. "Finding fuzzy sets for the mining of fuzzy association rules for numerical attributes". In *Proceedings of the First International Symposium on Intelligent Data Engineering and Learning* (IDEAL'98): 263-268, 1998.
- [Furnas et al. 1987] Furnas, G.W.; Landauer, T.K.; Gomez, L.M.; Dumais, S.T. "The vocabulary problem in human system communication", *Communications of the ACM*, 30(11): 964-972, 1987.
- [Furnival & Wilson 1974] Furnival, G.M.; Wilson, R.W.M. "Regression by leaps and bounds" *Technometrics*, 16: 499-511, 1974.
- [Fyfe & Baddeley 1995] Fyfe, C.; Baddeley, R. "Non-linear Data Structure Extraction Using Simple Hebbian Networks". *Biological Cybernetics*, 72(6):533-541, 1995.
- [Fyfe & Charles 1999] Fyfe, C.; Charles, D. "Using Noise to Form a Minimal Overcomplete Basis". In *Seventh International Conference on Artificial Neural Networks*, ICANN99, pages 708-713, 1999.
- [Fyfe 1993] Fyfe, C. "PCA Properties of Interneurons". In *From Neurobiology to Real World Computing*, ICANN 93, pages 183-188, 1993.
- [Fyfe 1995a] Fyfe, C. "Introducing Asymmetry into Interneuron Learning". *Neural Computation*, 7(6):1167-1181, 1995.
- [Fyfe 1995b] Fyfe, C. "Radial feature mapping". In *International Conference on Artificial Neural Networks*. ICANN95, Oct. 1995.
- [Fyfe 1997] Fyfe, C. "A Comparative Study of Two Neural Methods of Exploratory Projection Pursuit". *Neural Networks*, 10(2):257-262, 1997.
- [Gail & Grossberg 1987] Gail, A. C.; Grossberg S. "Art 2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns". *Applied Optics*, 26:4919-4930, 1987.

- [Gail 1990] Gail, A. C.; Grossberg S. "Art 3: hierarchical search using chemical transmitters in self-organizing pattern recognition architectures". *Neural Networks*, 3:129-152, 1990.
- [Gama & Bradzil 2000] Gama, J.; Brazdil P. "Cascade Generalization," *Machine Learning*, vol. 41, pp. 315–343, 2000.
- [Gibbons 1985] Gibbons, J. D. *Nonparametric statistical inference* (2nd ed.). New York: Marcel Dekker, 1985.
- [Giordana & Neri 1995] Giordana, A.; Neri, F. "Search-intensive concept induction", *Evolutionary Computation*, 3 (4): 375-416, 1995.
- [Girolami & Fyfe 1996] Girolami, M.; Fyfe, C. "Negentropy and Kurtosis as Projection Pursuit Indices Provide Generalised ICA Algorithms". In A. Cichocki and A. Back, editors, NIPS96 Blind Signal Separation Workshop, 1996. (Invited Contribution.)
- [Girolami & Fyfe 1996] Girolami, M.; Fyfe, C. "Blind Separation of Signals using Exploratory Projection Pursuit". In Speech and Signal Processing, International Conference on the Engineering Applications of Neural Networks, 1996.
- [Glantz & Slinker 2000] Glantz, S.A.; Slinker, B.K.; *Primer of Applied Regression & Analysis of Variance*, McGraw-Hill/Appleton & Lange; 2nd edition, 2000.
- [Goldberg 1989] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
- [Goldman & Widom 1999] Goldman, R.; Widom, J. "Approximate dataguides", in *Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, 1999.
- [Golfarelli et al. 1998] Golfarelli, M.; Maio, D.; Rizzi, S. "Dimensional fact model" *International Journal of Human-Computer Studies*, 43(5,6), pp. 865-889. 1998.
- [Green & Silverman 1994] Green, P.J.; Silverman, B.W. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*, Chapman & Hall, London, 1994.
- [Greenacre 1984] Greenacre, M.J. *Theory and Applications of Correspondence Analysis*, Academic Press, London, 1984.
- [Greene & Smith 1993] Greene, D.P.; Smith, S.F. "Competition-based induction of decision models from examples", *Machine Learning*, 13: 229-257, 1993.
- [Grossman et al. 2001] Grossman, R.L.; Kamath, C.; Kegelmeyer, P.; Kumar, V.; Namburu, R.R. "Data Mining for Scientific and Engineering Applications" Kluwer Academic Publishers, 2001.
- [Gunsfield 1997] Gunsfield, D. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge Press, 1997.
- [Guralnik & Karypis 2001] Guralnik, V.; Karypis, G. "A Scalable Algorithm for Clustering Sequential Data" *Proceedings of the 2001 IEEE International Conference on Data Mining*, 179-186. 2001.
- [Guttmann 1984] Guttmann, R. "R-trees: A dynamic index structure for spatial searching". In *Proc. ACM SIGMOD Conf. on Management Data*, 47-57, 1984.
- [Hale & Shenoi 1996] Hale, J.; Shenoi, S. "Analyzing FD inference in relational databases", *Data Knowledge Engineering*, 18: 167-183, 1996.
- [Hammond et al. 1995] Hammond, K.; Burke, R.; Martin, C.; Lytinen, S. "FAQ finder: A case-based approach to knowledge navigation", in *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, pp. 69-73, Stanford University, 1995. AAAI Press.
- [Han & Kamber 2001] Han, J.; Kamber, M. *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2000.
- [Han et al. 1996] Han, J.; Fu, Y.; Wang, W.; Chiang, J.; Gong, W.; Koperski, K.; Li, D.; Lu, Y. "DBMiner: a system for mining knowledge in large relational databases" in E. Simoudis, J. Han and U. Fayyad, (Eds.), *Proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 250-255. AAAI Press, 1996.

- [Han et al. 1997] Han, E.; Karypis, G.; Kumar, V.; Mobasher, B. "Clustering based on association rule hypergraphs", in *Proceedings of SIGMOD'97 Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 9-13, 1997.
- [Han et al. 2000] Han, L.; Pei, J.; Yin, Y. "Mining frequent patterns without candidate generation", in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1-12, 2000.
- [Hand & Till 2001] Hand, D.J.; Till, R.J. "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems" *Machine Learning*, 45, 171-186, 2001.
- [Hansen & Salamon 1990] Hansen L.K.; Salamon P. "Neural Network Ensembles" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12(10), pp 993-1001, 1990.
- [Harrison & Rubinfeld 1978] Harrison, D.; Rubinfeld, D.L. "Hedonic prices and the demand for clean air", *Journal of Environment. Economics & Management*, vol.5, 81-102, 1978.
- [Hastie & Tibshirani 1990] Hastie, T.; Tibshirani, R. *Generalized Additive Models*, Chapman & Hall, London, 1990.
- [Hastie et al. 2001] Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer Verlag, 2001.
- [Haykin 1998] Haykin, S. *Neural Networks- A Comprehensive Foundation*. Macmillan 1st Edition, 1994, Prentice Hall, 2nd Edition, 1998.
- [Hebb 1949] Hebb, D. O. "The Organisation of Behaviour". Wiley, 1949.
- [Heckerman et al. 1995] Heckerman, D.; Chickering, M.; Geiger, D. "Learning bayesian networks: the combination of knowledge and statistical data". *Machine Learning*. 20:197–244. 1995.
- [Herbrich 2002] Herbrich, R. *Learning Kernel Classifiers*, MIT Press, 2002.
- [Hernández & Velilla 2001] Hernández, A.; Velilla, S. "Dimension reduction in nonparametric discriminant analysis". *Working Paper 01-39, Statistics and Econometric Series 25, Universidad Carlos III de Madrid*. 2001.
- [Hernández 1999] Hernández-Orallo, José. "Computational Measures of Information Gain and Reinforcement in Inference Processes" Dpto. Lógica y Filosofía de la Ciencia, Universidad de Valencia, Tesis Doctoral, 1999. Resumen en *AI Communications*, 13, 49-50, 2000. (<http://www.dsic.upv.es/~jorallo/tesi/index.html>).
- [Hirsh et al. 2000] Hirsh, H.; Basu, C.; Davison, B.D. "Learning to Personalize", *Communications of the ACM*, 43(8): 102-106, 2000.
- [Ho 1998] Ho T.K. "C4.5 Decision Forests" in *Proceedings of the International Conference on Pattern Recognition*, pp 545-549, 1998.
- [Hoeting et al. 1999] Hoeting, J.; Madigan, D; Raftery A.E.; Volinsky, C.T. "Bayesian Model Averaging: A Tutorial" *Statistical Science*, Vol 14, 1999.
- [Holder & Cook 2003] Holder, L.B.; Cook, D.J. "Graph-based relational learning: current and future directions", *SIGKDD Explorations*, 5(1):90-93, 2003.
- [Holland 1975] Holland, J.H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [Hsu & Dung 1998] Hsu, C.N.; Dung, M.T. "Generating finite-state transducers for semi-structured data extraction from the web", *Information Systems*, 23(8): 521-538, 1998.
- [Hu 1998] Hu, Y.-T. "A genetic programming approach to constructive induction". In *Proceedings of Third International Conference on Genetic Programming*: 146-151, 1998.
- [IBM 2001] IBM Redbooks "Mining Your Own Business in Health Care Using DB2 Intelligent Miner for Data", IBM Corp, 2001.
- [Imielinski & Manilla 1996] Imielinski, T.; Mannila, H. "A database perspective on knowledge discovery" *Communications of the ACM*, 39(11):58-64, 1996.
- [Imielinski et al. 1996] Imielinski, T.; Virmani, A.; and Abdulghani, A. "Discovery board application programming interface and query language for database mining" in *Proceedings of Knowledge Discovery in Databases Conference (KDD'96)*, Portland, Ore, Aug, 1996, pp. 20-26.

- [Inmon 1992] Inmon, W.H. "EIS and the data warehouse: A simple approach to building an effective foundation for EIS" *Database Programming and Design*, 5(11):70-73, 1992.
- [Inmon 2002] Inmon, W.H. *Building the Data Warehouse*, 3rd Edition, Wiley, Chichester, 2002.
- [Inokuchi et al. 2003] Inokuchi, A.; Washio, T.; Motoda, H. "Complete mining of frequent patterns from graphs : Mining graph data", *Machine Learning*, 50 : 321-354, 2003.
- [Inselberg & Dimsdale 1990] Inselberg, A.; Dimsdale, B. "Parallel co-ordinates. a tool for visualising multi-dimensional geometry" Proc. of Visualization'90, pp. 361-370, San Francisco, CA, USA, 1990.
- [Isasi & Galván 2003] Isasi, P.; Galván, I. *Las redes neuronales artificiales y sus aplicaciones prácticas*, Prentice may 2003.
- [Janikow 1993] Janikow, C.Z. "A knowledge-intensive genetic algorithm for supervised learning", *Machine Learning*, 13: 189-228, 1993.
- [Janikow 1998] Janikow, C.Z. "Fuzzy decision trees: issues and methods" *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 28(1): 1-14, 1998.
- [Jensen 2001] Jensen, F.V. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.
- [Joachims 1996] Joachims, T. "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization", Computer Science Technical Report CMU-CS-96-118, Carnegie Mellon University, 1996.
- [Joachims 1998] Joachims, T. "Text Categorization with support vector machines: Learning with many relevant features", in *Proceedings of the 10th European Conference on Machine Learning*, pp. 137-142, 1998.
- [Joachims 2002] Joachims, T. *Learning to Classify Text Using Support Vector Machines -- Methods, Theory and Algorithms*. Kluwer Academic Publishers, 2002.
- [Joachims et al. 1997] Joachims, T.; Freitag, D.; Mitchell, T.M. "Web Watcher: A Tour Guide for the World Wide Web", in Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence IJCAI 97, pp: 770-777, 1997.
- [Johnson 1999] Johnson, D.E. *Métodos multivariados aplicados al análisis de datos*, Paraninfo 1999.
- [Kargupta et al. 1997a] Kargupta, H.; Hamzaoglu, I.; Stafford, B. "Scalable, Distributed Data Mining Using An Agent Based Architecture", in *Proceedings of the Knowledge Discovery and Data Mining*, pp. 211-214, AAAI Press, 1997.
- [Kargupta et al. 1997b] Kargupta, H.; Hamzaoglu, I.; Stafford, B.; Hanagandi, V.; Buescher, K. "PADMA: Parallel data mining agent for scalable text classification", in *Proceedings of the Conference on High Performance Computing*, pp. 290-295, 1997.
- [Kargupta et al. 2000] Kargupta, H.; Park, B.; Hershbereger, D.; Johnson, E. "Collective Data Mining: A new perspective toward distributed data analysis", *Advances in Distributed Data Mining*, AAAI/MIT Press, p. 133-184, 1999.
- [Kargupta et al. 2001] Kargupta, H.; Sivakumar, K.; Huang, W.; Ayyagari, R.; Chen, R.; Park, B.; Johnson, E. "Towards ubiquitous mining of distributed data", in R. Grossman and C. Kamath and Philip Kegelmeyer and V. Kumar and R. Namburu *Data Mining for scientific and engineering applications*, Kluwer Academic Publishers, p. 281-306, 2001.
- [Kashima & Inokuchi 2002] Kashima, H.; Inokuchi, A. "Kernels for graph classification", in *Proceedings of International Workshop on Active Mining*, pp. 71-79, 2002.
- [Kashyap & Blaydon 1968] Kashyap, R.L.; Blaydon, C.C. "Estimation of probability density and distribution functions". *IEEE Trans. Info. Theory*, IT-14, 549-556, July 1968.
- [Kass 1975] Kass, G. V. "Significance testing in automatic interaction detection" *Applied Statistics* 24(2):178-189. 1975.
- [Kass 1980] Kass, G.V. "An Exploratory Technique for Investigating Large Quantities of Categorical Data", *Applied Statistics*, 29, 119-127, 1980.
- [Kaufman & Rousseeuw 1990] Kaufman, L.; Rousseeuw, P. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, 1990.

- [Kearns & Mansour 1996] Kearns, M.; Mansour, Y. "On the boosting ability of top-down decision tree learning algorithms". *Journal of Computer and Systems Sciences*, 58(1), 1999, pp 109-128. Also in Proceedings ACM Symposium on the Theory of Computing, ACM Press, pp.459-468, 1996.
- [Kendall 1975] Kendall, M. G. *Rank correlation methods*, 4th ed. London: Griffin 1975.
- [Kimball 1996] Kimball, R. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Datawarehouses*, Wiley: New York, 1996.
- [Kiritchenko & Matwin 2001] Kiritchenko, S.; Matwin, S. "Email Classification with Co-training", *Technical Report School of Information Technology and Engineering University of Ottawa* (<http://www.site.uottawa.ca/~stan/papers/2001/cascon2002.pdf>), 2001.
- [Kirkpatrick et al. 1983] Kirkpatrick, C.G.Jr.; Vecchi, M. "Optimization by simulated annealing" *Science*, 220:671-680, 1983.
- [Kleinberg 1999] Kleinberg, J.M. "Authoritative sources in a hyperlinked environment", *Journal of the ACM*, 46(5): 604-632, 1999.
- [Klir & Yuan 1995] Klir, G.; Yuan, B. *Fuzzy Sets and Fuzzy Logic. Theory and Applications*. Prentice-Hall, 1995.
- [Klösgen & Zytkow 2002] Klösgen, W.; Zytkow, J.M. *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, 2002.
- [Knorr & Ng 1998] Knorr, E.; Ng, R. "Algorithms for mining distance-based outliers in large datasets" in *Proc. 1998 Intl. Conference on Very Large Data Bases (VLDB'98)*, páginas 392-403, 1998.
- [Kohavi & John 1997] Kohavi, R.; John, G. "Wrappers for feature subset selection", *Artificial Intelligence*, Vol. 97, pp . 273-324, 1997.
- [Kohonen 1982] Kohonen T. "A simple paradigm for the self-organized formation of structured feature maps". In M. Arbib: S. Amari (Eds.). *Lecture notes in biomathematics 45: Competition and cooperation in neural nets*, (pp. 248-266). Berlin. Springer-Verlag. 1982.
- [Kohonen 1984] Kohonen T. *Self-organization and associative memory*, Springer-Verlag, Berlin, 1984.
- [Kohonen 1986] Kohonen T. "Learning vector quantization for pattern recognition". Helsinki university of technology, department of technical physics. Technical report, TKK-F-A601. 1986.
- [Kohonen 1995] Kohonen, T." Self-Organising Maps". Springer, 1995.
- [Koller & Pfeffer 1997] Koller, D.; Pfeffer, A. "Object-oriented Bayesian networks", in *Proc. of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 302-313, 1997.
- [Koller & Pfeffer 1998] Koller, D.; Pfeffer, A. "Probabilistic frame-based systems", in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 508-587, 1998.
- [Kolodner 1993] Kolodner, J.L. *Case-Based Reasoning*. San Francisco: Morgan Kaufmann, 1993.
- [Koperski & Hand 1995] Koperski, K.; Han, J. "Discovery of Spatial Association Rules in Geographic Information Databases", *Proc. 4th Int'l Symp. on Large Spatial Databases (SSD95)* pp. 47-66. 1995.
- [Koperski et al. 1997] Koperski, K.; Han, J.; Adhikary J. "Spatial Data Mining: Progress and Challenges", *Technical Report, Simon Fraser University, Burnaby, B.C.; Canada*, <http://db.cs.sfu.ca/GeoMiner/survey/>, 1997.
- [Koperski et al. 1998] Koperski, K.; Han, J.; Stefanovic, N, "An Efficient Two-Step Method for Classification of Spatial Data", In *Proc. 1998 International Symposium on Spatial Data Handling SDH'98*, , Vancouver, BC, Canada, 1998.
- [Kosala & Blockeel 2000] Kosala, R.; Blockeel, H. "Web Mining Research: A Survey", *ACM SIGKDD Explorations*, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining , vol. 2, pp: 1-15, 2000.
- [Koza 1992] Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
- [Koza 1994] Koza, J.R. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press 1994.
- [Kramer & Widmer 2001] Kramer, S.; Widmer, G. "Inducing Classification and Regression trees in First Order Logic", in [Džeroski & Lavrač 2001], pp: 140-159, 2001.

- [Krzanowski 1988] Krzanowski, W.J. *Principles of Multivariate Analysis: A User's Perspective*, Number 3 in Oxford Statistical Science Series, Oxford University Press, Oxford, 1988.
- [Krzanowski 1996] Krzanowski, W.J. *Principles of Multivariate Analysis*. Oxford University Press, 1996.
- [Krzysztof 2001] Krzysztof J. Cios (Ed.) *Medical Data Mining and Knowledge Discovery*, Physica Verlag, Springer, 2001.
- [Kudo & Skalansky 2000] Kudo, M.; Skalansky, J. "Comparison of algorithms that select features for pattern classifiers", *Pattern Recognition* 33 (1): 25-41, 2000.
- [Kuhn & Tucker 1951] Kuhn, H.; Tucker, A. "Nonlinear Programming". In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probabilistics*, pp. 481-492, University of California Press, 1951.
- [Kuncheva 2002] Kuncheva, L.I. "A Theoretical Study on Six Classifier Fusion Strategies", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 24(2), pp 281-286, 2002.
- [Kushmerick et al. 1997] Kushmerick, N.; Weld, D.; Doorenbos, R. "Wrapper induction for information extraction", in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 729-737, 1997.
- [Lagus et al. 1999] Lagus, K.; Honkela, T.; Kaski, S.; Kohonen, T. "Websom for textual data mining". *Artificial Intelligence Review*, vol. 13, issue 5/6, pp. 345-364, Kluwer Academic Publisher, 1999.
- [Lang 1995] Lang, K. "NewsWeeder: Learning to filter netnews", in *Proceedings of the 12th International Conference on Machine Learning*, pp: 331-339, 1995.
- [Langley et al. 1992] Langley, P.; Iba, W.; Thompson. K. "An Analysis of bayesian Classifiers". In *Proceedings of the 10th National Conference on Artificial Intelligence*. 223-223. 1992.
- [Larrañaga & Lozano 2002] Larrañaga, P.; Lozano, J.A. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.
- [Larrañaga 1998] Larrañaga, P. "Aprendizaje automático de modelos gráficos II: Aplicaciones a la clasificación supervisada". En *Sistemas Expertos Probabilísticos* (Gámez & Puerta, editores). Ediciones de la Universidad de Castilla-La Mancha. pp 141-162. 1998.
- [Lavrač & Džeroski 1994] Lavrač, N.; Džeroski, S. *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, 1994.
- [Lavrač et al. 1991] Lavrač, N.; Džeroski, S.; Grobelnik, M. "Learning nonrecursive definitions of relations with LINUS", in *Proceedings of the Fifth European Working Session on Learning*, pp. 265-281, 1991.
- [Lebart et al. 1984] Lebart, L.; Morineau, A.; Fenelon, J-P. *Tratamiento Estadístico de Datos*. Marcombo-Boixareu, Barcelona, 1984.
- [Lee & Kim 1997] Lee, D.H.; Kim, M.H. "Database summarization using fuzzy ISA hierarchies", *IEEE Transactions on Systems, Man and Cybernetics B*, 27: 68-78, 1997.
- [Lewis & Ringuette 1994] Lewis, D.D.; Ringuette, M. "A comparison of two learning algorithms for text categorization", in *Third Annual Symposium on Document Analysis and Information Retrieval*, pp: 81-93, 1994.
- [Li & Vitányi 1997] Li, M.; Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications*, Springer Verlag; 2nd edition, 1997.
- [Liu & Motoda 1998a] Liu, H.; Motoda, H. (eds.) *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Boston, Kluwer Academic Publishers, 1978.
- [Liu & Motoda 1998b] Liu, H.; Motoda, H. (eds.) *Feature Selection for Knowledge Discovery and Data Mining*, Boston, Kluwer Academic Publishers, 1998.
- [Liu & Motoda 2001] Liu, H.; Motoda, H. *Instance selection and construction for Data Mining* Kluwer Academic Publishers, 2001.
- [Liu & Setiono 1998] Liu, H.; Setiono, R. "Incremental feature selection" *Applied Intelligence*, Vol. 9, pp. 217-230, 1998.

- [Liu et al. 2000] Liu, B.; Xia, Y.; Yu, P.S. "Clustering Through Decision Tree Construction" in Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 6-11, pp. 20-20, ACM 2000.
- [Liu et al. 2002] Liu, H, Hussain, F, Tan, C.L.; Dash, M. "Discretization: An Enabling Technique". *Data Mining and Knowledge Discovery*, 6: 393-423, 2002.
- [Lloyd 1987] Lloyd, J. *Foundations of Logic Programming*, Springer Verlag, 1987.
- [Lodhi et al. 2002] Lodhi, H.; Saunders, C.; Shawe-Taylor, J.; Cristianini, N.; Watkins, C. "Text Classification using String Kernels". *Journal of Machine Learning Research*, 2:419-444, 2002.
- [Loh et al. 2000] Loh, S.; Wives, L.K.; Palazzo, J. "Concept-based Knowledge Discovery in Texts Extracted from the Web." *SIGKDD Explorations*, 2(1): 29-39, 2000.
- [Lui et al. 2002] Liu, H, Hussain, F, Tan, C.L.; Dash, M. "Discretization: An Enabling Technique" *Data Mining and Knowledge Discovery*, 6: 393-423, 2002.
- [MacDonald 2001] MacDonald, D. "Unsupervised Neural Networks for Visualisation of Data". PhD thesis, University of Paisley, 2001.
- [Maclin & Optiz 1997] Maclin, R.; Optiz, D. "An Empirical Evaluation of Bagging and Boosting" in *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pp 546-541, 1997.
- [MacQueen 1967] MacQueen, J. "Some methods for classification and analysis of multivariate observations". *Proceedings of the fifth Berkley symposium on mathematical statistics and probability*, 1, 281-297, 1967.
- [Madsen et al. 2003] Madsen, A.; Lang, M.; Kjaerulff, U.; Jensen, F. "The Hugin Tool for learning bayesian Networks". In *Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. Pp. 594-605. 2003.
- [Manning & Schütze 1999] Manning, C.D.; Schütze, H. *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.
- [Marcus & Minc 1988] Marcus, M.; Minc, H. *Introduction to Linear Algebra*, New York: Dover, p. 145, 1988.
- [Mattison 1997] Mattison, R. *Data Warehousing and Data Mining for Telecommunications*, Artec House Publishers, 1997.
- [McClelland et al. 1986] McClelland, J.; Rumelhart D. E.; The PDP Research Group. *Parallel Distributed Processing*, Volume 1 and 2. MIT Press, 1986.
- [McCullagh & Nelder 1989] McCullagh, P.; Nelder, J.A. *Generalized Linear Models*, number 37 in *Monographs on Statistics and Applied Probability*, Chapman and Hall, London, 2nd edition, 1989.
- [McHugh et al. 1997] McHugh, J.; Abiteboul, S.; Goldman, R.; Quass, D.; Widom, J. Lore: "A database management system for semistructured data". *SIGMOD Record*, 26(3): 54-66, 1997.
- [McLachan & Krishnan, 1996] McLachan, G.J.; Krishnan, T. *The EM Algorithm and Extensions*, Wiley, 1996.
- [McLachlan 1992] McLachlan, G.J. *Discriminant Analysis and Statistical Pattern Recognition*, John Wiley, 1992.
- [Mehta et al. 1996] Mehta, M.; Agrawal, R.; Rissanen, J. "SLIQ: A Fast Scalable Classifier for Data Mining" in Peter M. G. Apers, Mokrane Bouzeghoub, Georges Gardarin (Eds.): *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology*, Avignon, France, March 25-29, 1996, Proceedings. Lecture Notes in Computer Science 1057, EDBT 1996, pp. 18-32, Springer 1996.
- [Mena 1999] Mena, J. *Data Mining Your Website*. Digital Press, 1999.
- [Mendes et al. 2001] Mendes, R. R. F.; Voznika, F. B.; Freitas, A. A. and Nievola, J. C. (2001) "Discovering fuzzy classification rules with genetic programming and co-evolution". *Principles of Data Mining and Knowledge Discovery* (Proc. 5th European Conf., PKDD 2001) - Lecture Notes in Artificial Intelligence 2168: 314-325, 2001.
- [Mercer 1909] Mercer, T. "Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations". *Philos. Trans. Roy. Soc. London*, Vol. A 209, pp. 415-446, 1909.

- [Michalski & Larson 1983] Michalski, R.S.; Larson, J. "Incremental generation of v1 hypotheses: the underlying methodology and the description of program aq11" *ISG 83-5*, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, 1983.
- [Michalski et al. 1986] Michalski, R.S.; Mozetic, I.; Hong, J.; Lavrac, N. "The AQ15 inductive learning system: an overview and experiments" in *Proceedings of IMAL 1986*, Orsay, France, Université de Paris-Sud, 1986.
- [Michie et al. 1994] Michie, D.; Spiegelhalter, D.; Taylor, C.C. *Machine learning, neural and statistical classification* (editors). Ellis Horwood, 1994.
- [Miller & Han 2001] Miller, H.; Han, J. (eds.) *Geographic Data Mining and Knowledge Discovery*, Taylor & Francis, Inc., New York, NY, 2001.
- [Minsky & Papert 1969/1988] Minsky M. and Papert. S. A. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, re-edited 1988, expanded edition, 1969/1988.
- [Minton et al. 1989] Minton, S.; Carbonell, J. G.; Knoblock, C. A.; Kuokka, D. R.; Etzioni, O.; and Gil, Y. 1989. "Explanation-based learning: A problem-solving perspective". *Artificial Intelligence*, 40:63-118, 1989.
- [Mitchell 1997] Mitchell, T.M. *Machine Learning*. McGraw-Hill, 1997.
- [Mitchell 1999] Mitchell, T. M. "Machine Learning and Data Mining" *Communications of the ACM*, 42(11):30-36, 1999.
- [Mladenic D. et al. 2003] Mladenic D.; Lavrac N.; Bohanec M.; Moyle S. "Data Mining and Decision Support: Integration and Collaboration", *Kluwer Academic Publishers*, 2003.
- [Mobasher et al. 2000] Mobasher, B.; Cooley, R.; Srivastava, J. "Automatic Personalization Based on Web Usage Mining", *Communications of the ACM*, 43(8): 142-151, 2000.
- [Montgomery & Peck 1992] Montgomery, D.C.; Peck, E.A. *Introduction to linear regression analysis*. John Wiley, 1992.
- [Moody & Darken 1989] Moody, J.; Darken, C. "Fast learning in networks of locally tuned processing units". *Neural computation*, 1, 281-294. 1989.
- [Mooney & Califf 1995] Mooney, R.J.; Califf, M.E. "Induction of First-Order Decision Lists: Results on Learning the Past Tense of English Verbs", *Journal of Artificial Intelligence Research*, 3: 1-24, 1995.
- [Mooney et al. 2002] Mooney, R.J.; Melville, P.; Tang, L.R.; Shavlik, J.; de Castro, I.; Page, D. "Relational Data Mining with Inductive Logic Programming for Link Discovery", in *Proceedings of the National Science Foundation Workshop on Next Generation Data Mining*, 2002.
- [Morgan & Messenger 1973] Morgan, J.N.; Messenger, R.C. "THAID, a sequential analysis program for the analysis of nominal scale dependent variables", Survey Research Center, U of Michigan, 1973.
- [Morgan & Sonquist 1963] Morgan, J.N.; Sonquist, J.A. "Problems in the Analysis of Survey Data, and a Proposal", *J. Amer. Static. Assoc.*, 58, 415-434, 1963.
- [Morik et al. 1993] Morik, K.; Wrobel, S.; Kietz, J.-U.; Emde, W. *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications*, Academic Press, 1993.
- [Moulinier 1996] Moulinier, I. "A Framework for Comparing Text Categorization Approaches", in *Proceedings of the International Conference on Machine Learning*, 1996.
- [Muggleton & Buntine 1988] Muggleton, S.; Buntine, W. "Machine invention of first-order predicates by inverting resolution", in *Proceedings of the 5th International Conference on Machine Learning*, pp. 339-352, 1988.
- [Muggleton & Feng 1990] Muggleton, S.; Feng, C. "Efficient Induction of Logic Programs", in *Proceedings of the First Conference on Algorithmic Learning Theory*, pp. 368-381, 1990.
- [Muggleton 1990] Muggleton, S. "Inductive Logic Programming", in *Proceedings of the First Conference on Algorithmic Learning Theory*, pp. 42-62, 1990.
- [Muggleton 1991] Muggleton, S. "Inductive Logic Programming", *New Generation Computing*, 8(4):295-318, 1991.
- [Muggleton 1992] Muggleton, S. (editor), *Inductive Logic Programming*, Academic Press, 1992.

- [Muggleton 1995] Muggleton, S. "Inverse entailment and Progol", *New Generation Computing*, 13:245-286, 1995.
- [Muggleton 1996] Muggleton, S. "Learning from positive data", in *Proceedings of the 6th International Workshop on Inductive Logic Programming*, Lecture Notes in Computer Science, vol. 1314, pp. 358-376, 1996.
- [Muggleton 2002] Muggleton, S. "Learning structure and parameters of stochastic logic programs", in *Proceedings of the Twelfth International Conference on Inductive Logic Programming*, LNCS vol 2583, pp. 198-206, 2002.
- [Müller et al. 2001] Müller, K.-R.; Mika, S.; Rätsch, G.; Tsuda, K.; Schölkopf, B. "An Introduction to Kernel-Based Learning Algorithms". *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, pp. 181-202, 2001.
- [Murthy et al. 1994] Murthy, S.K.; Kasif, S.; Salzberg, S. "A System for Induction of Oblique Decision Trees" *Journal of Artificial Intelligence Research*, 2, 1-32, 1994.
- [Muslea 1999] Muslea, I. "Extraction patterns for information extraction tasks: A survey", in *Proceedings of the AAI-99 Workshop on Machine Learning for Information Extraction*, pp. 763-771, 1999.
- [Nelder & Wedderburn 1972] Nelder, J.A.; Wedderburn, R.W.M. "Generalised Linear Models". *Journal of the Royal Statistical Society A*, 135: 370-384, 1972.
- [Nestorov et al. 1998] Nestorov, S.; Abiteboul, S.; Motwani, R. "Extracting Schema from Semistructured Data", in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 295-306, 1998.
- [Ng & Han 1994] Ng, R.T.; Han, J. "Efficient and Effective Clustering Methods for Spatial Data Mining" *In Proc. 1994 International conference on Very Large Data Bases VLDB'94*: 144-155. 1994
- [Ng et al. 1998] Ng, R.; Lakshmanana, L.V.S.; Han, J.; Pang, A. "Exploratory Mining and Pruning Optimizations of Constrained Association Rules" *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, New York, 1998, pp. 13-24
- [Nienhuys-Cheng & de Wolf 1997] Nienhuys-Cheng, S.H.; de Wolf, R. *Foundations of Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, Vol. 1228, 1997.
- [Noda et al. 1999] Noda, E.; Freitas, A.A.; Lopes, H.S. "Discovering Interesting Prediction Rules with a Genetic Algorithm", in *Proceedings of the Congress on Evolutionary Computation* 2: 1322--1329, 1999.
- [Núñez 1988] Núñez, M. "Economic Induction: A case study" *In the proceedings of the 3rd European Working Session on Learning, EWSL-98*, volume 55, pages 139-145, Morgan Kaufmann, 1998.
- [Oliver 1993] Oliver, J.J. "Decision graphs - an extension of decision trees" in *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pp. 343-350, 1993. Extended version available as TR 173, Department of Computer Science, Monash University, Clayton, Victoria 3168, Australia.
- [Ordonez & Omiecinski 1999] Ordonez, C.; Omiecinski, E. "Discovering Association Rules Based on Image Content" *Advances in Digital Libraries*, 38-49, 1999.
- [Ozsu & Valduriez 1999] Ozsu, M.T.; Valduriez, P. *Principles of Distributed Database Systems*, Prentice Hall 1999.
- [Page et al. 1998] Page, L.; Brin, S.; Motwani, R.; Winograd, T. "The PageRank citation ranking: Bringing order on the Web", in *Proceedings of the 7th International World Wide Web Conference*, pp. 161-172, 1998.
- [Parmanto et al. 1996] Parmanto B; Munro P.; Doyle H. "Improving Committee Diagnosis with Resampling Techniques" *Advances in Neural Information Processing Systems*, Vol 8, pp 882-888, 1996.
- [Pearl 1988] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [Pedrycz 1996] Pedrycz, W. "Conditional fuzzy c-means", *Pattern Recognition Letters* 17: 625-632, 1996.
- [Peña & Prieto 2001] Peña, D.; Prieto, F.J. "Robust covariance matrix estimation and multivariate outlier detection", *Technometrics*, 3: 286-310, 2001.
- [Peña 2001] Peña, D. *Fundamentos de Estadística*. Alianza Editorial, Madrid, 2001.

- [Peña 2002a] Peña, D. *Regresión y Diseño de Experimentos*. Alianza Editorial, Madrid, 2002.
- [Peña 2002b] Peña, D. *Ánalisis de datos multivariantes*. McGraw Hill, 2002.
- [Plotkin 1970] Plotkin, G. "A Note on Inductive Generalization", *Machine Intelligence*, 5: 153-163, 1970.
- [Plotkin 1971] Plotkin, G. "A Further Note on Inductive Generalization", *Machine Intelligence*, 6: 101-124, 1971.
- [Provost & Domingos 2003] Provost, F.; Domingos, P. "Tree Induction for Probability-based Ranking", *Machine Learning* 52:3, 2003.
- [Provost & Fawcett 1997] Provost, F.; Fawcett, T. "Analysis and Visualization of Classifier Performance: Comparison Under Imprecise Class and Cost Distributions" *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pp. 43, AAAI press, 1997.
- [Provost & Kolluri 1997] Provost, F.; Kolluri, V. "Scaling up inductive algorithms: An overview" in Heckerman, D.; Mannila, H.; Pregibon, D.; Uthurusamy, R. (eds.) *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 239-242. AAAI Press, 1997.
- [Punch et al. 1993] Punch, W.F.; Goodman, E.D.; Pei, M.; Chia-Shun, L.; Hovland, P.; Enbody, R. "Further Research on Feature Selection and Classification using Genetic Algorithms", in *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93)*: 557-564, 1993.
- [Pyle 2003] Pyle, P. "Systems, Simulation and Data Mining", *The connector*, Vol.1, Issue 4, 2003.
- [Quinlan & Cameron-Jones 1993] Quinlan, J.R.; Cameron-Jones, R.M. "FOIL: A Midterm Report". In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667 of *Lecture Notes in Artificial Intelligence*, pages 3-20. Springer-Verlag, 1993.
- [Quinlan 1983] Quinlan, J.R. "Learning Efficient Classification Procedures and Their Application to Chess End Games" in Michalski, R.; Carbonell, J.; Mitchell, T. (Eds.) *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, San Mateo, CA, 1983.
- [Quinlan 1986] Quinlan, J.R. "Induction of Decision Trees" *Machine Learning*, 1, 81-106, 1986.
- [Quinlan 1987a] Quinlan, J.R. "Simplifying decision trees", *International Journal of Man-Machine Studies*, 27, 221-234, 1987.
- [Quinlan 1987b] Quinlan, J.R.; *Generating production rules Machine Learning*. Morgan Kaufmann, 1987.
- [Quinlan 1990] Quinlan, J.R. "Learning logical definitions from Relations". *Machine Learning*, 5:239-266, 1990.
- [Quinlan 1993] C4.5. *Programs for Machine Learning*, San Francisco, Morgan Kaufmann, 1993.
- [Quinlan 1996a] Quinlan, J.R. "Learning first-order definitions from relations" *Machine Learning*, 5(3), 239-266, 1996.
- [Quinlan 1996b] Quinlan, J.R. "Bagging, Boosting, and C4.5" in *Proceedings of the 30th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence*, pp 725-730, 1996.
- [Rabiner & Juang 1986] Rabiner, L. R.; Juang, B.H. "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, 4-15, 1986.
- [Rätsch 2001] Rätsch, G. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Department of Computer Science, Potsdam, Germany, 2001.
- [Raymer et al. 2000] Raymer, M.L.; Punch, W.F.; Goodman, E.D.; Kuhn, L.A.; Jain, A.K. "Dimensionality reduction using genetic algorithms", *IEEE Transactions on Evolutionary Computation* 4(2): 164-171, 2000.
- [Renche 2002] Renche, A.C. *Methods of Multivariate Analysis*, 2nd. Edition, Wiley-Interscience, 2002.
- [Resnick & Varian 1997] Resnick, P.; Varian, H. R. "Recommender Systems - Introduction to the Special Section" *Communications ACM* 40(3): 56-58, 1997.
- [Rissanen 1978] Rissanen, J. "Modeling by shortest data description". *Automatica*, vol. 14, pp. 465-471, 1978.
- [Roberts 1998] Roberts, Huw "Keys to the Commercial Success of Data Mining" workshop held in KDD'98.
- [Rocchio 1971] Rocchio, J. "Relevance Feedback in Information Retrieval" in *The SMART Retrieval System: Experiments in Automatic Document Processing*, Chapter 14, pages 313-323, Prentice Hall Inc.; 1971.

- [Romao et al. 2002] Romao,W.; Freitas, A.A.; Pacheco, R.C.S. "A Genetic Algorithm for Discovering Interesting Fuzzy Prediction Rules: applications to science and technology data". In Proc. *Genetic and Evolutionary Computation Conference*, 2002.
- [Rosenblatt, 1962] Rosenblatt, F. *Principles of Neurodynamics*. Spartan Books, New York. 1962.
- [Rousseeuw & Leroy 2003] Rousseeuw, P.J.; Leroy, A.M. *Robust Regression and Outlier Detection*, John Wiley and Sons, New York, 1987. 2nd Edition, Wiley-Interscience, 2003.
- [Russell & Norvig 2002] Russell, S; Norvig, P.; *Artificial Intelligence: A Modern Approach*, 2002.
- [Ryan & Rayward-Smith 1998] Ryan, M.D.; Rayward-Smith, V.J. "The evolution of decision trees". In *Genetic Programming 1998: Proceedings of the 3rd. Annual Conference*: 350-358, 1998.
- [Sahami et al. 1996] Sahami, M.; Hearst, M.; Saund, E. "Applying the Multiple Cause Mixture Model to Text Categorization", in *Proceedings of AAAI Spring Symposium on Machine Learning and Information Access*, pp. 435-443, 1996.
- [Salton & McGill 1983] Salton, G.; McGill, M.J. *Introduction to modern information retrieval*. McGraw Hill, New York, 1983.
- [Sapia et al. 1999] Sapia, C. Blaschka, M.; Höfling, G.; Dinter, B. "Extending the E/R model for the multidimensional paradigm" International Workshop on Data Warehouse and Data Mining (DWDM). 1999.
- [Schapire 2002] Schapire, R. E. The Boosting Approach to Machine Learning: An Overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, Lecture Notes in Statistics 171, Springer Verlag 2002.
- [Schölkopf & Smola 2002] Schölkopf, B.; Smola, A. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
- [Schölkopf et al. 1999] Schölkopf, B.; Burges, C.; Smola, A. *Advances in Kernel Methods. Support Vector Learning*. MIT Press, 1999.
- [Schütze et al. 1995] Schütze, H.; Hull, D.; Pedersen, J.O. "A comparison of classifiers and document representations for the routing problem", in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 229-237, 1995.
- [Schwefel 1995] Schwefel, H.P.; *Evolution and Optimum Seeding*, Wiley Inc. 1995.
- [Scott & Matwin 1999] Scott, S.; Matwin, S. "Feature engineering for text classification", in *Proceedings of the International Conference on Machine Learning*, pp. 379-388, 1999.
- [Sellis et al. 1987] Sellis, T.; Roussopoulos, N.; Faloutsos, C. "The R+-Tree: A Dynamic Index For Multi-Dimensional Objects" *The VLDB Journal*, 507-518, 1987.
- [Shafer et al. 1996] Shafer, J.C.; Agrawal, R.; Mehta, M. "SPRINT: A scalable parallel classifier for data mining" in Vijayaramen et al. (eds) Proc. 22nd Int. Conf. Very Large Databases, VLDB, pp. 544-555, Sept 1996.
- [Shapiro 1983] Shapiro, E.Y. *Algorithmic Program Debugging*. MIT Press, Cambridge, MA, 1983.
- [Shardanand & Maes 1995] Shardanand, U.; Maes, P. "Social Information Filtering: Algorithms for Automating 'word of mouth'", in *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pp. 210-217, 1995.
- [Shen et al. 1996] Shen, W.M.; Ong, K.; Mitbander, B.; Zaniolo, C. "Metaqueries for Data Mining" in *Advances in Knowledge Discovery and Data Mining*, U.M.Fayyad et al. (eds.) AAAI, 1996, pp. 375-398.
- [Silverman 1986] Silverman, B.W. *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability, Chapman & Hall, 1986.
- [Silverstein et al. 1997] Silverstein, C.; Brin, S.; Motwani "Beyond market baskets: generalizing associations rules to correlations", in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 265-276, 1997.
- [Simoff 2002] Simoff, S.J.; Djeraba, C.; Zaïane O.R "MDM/KDD 2002: Multimedia Data Mining between Promises and Problems" *SIGKDD Explorations* 4(2): 118-121, 2002.

- [Simonoff 1996] Simonoff, J.S. *Smoothing Methods in Statistics*, Springer, 1996.
- [Smola et al. 2000] Smola, A.; Bartlett, P.; Schölkopf, B.; Schuurmans, D. *Advances in Large Margin Classifiers*. MIT Press, 2000.
- [Solomonoff 1966] Solomonoff, R. "A Formal Theory of Inductive Inference, Part I" *Information and Control*, Part I: Vol 7, No. 1, pp. 1-22, March 1964.
- [Spiliopoulou et al. 1999] Spiliopoulou, M.; Faulstich, L.C.; Wikler, K. "A data miner analysing the navigational behaviour of web users", in *Proceedings of the Workshop on Machine Learning in User Modelling of the ACAI99*, pp. 588-599, 1999.
- [Spirtes et al. 1991] Spirtes, P.; Glymour, C.; Scheines, R. "An algorithm for fast recovery of sparse causal graphs". *Social Science Computing Reviews*, 9:62-72. 1991.
- [Srinivasan 1999] Srinivasan, A. "Note on the Location of Optimal Classifiers in N-dimensional ROC Space" *Technical Report PRG-TR-2-99*, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford. 1999.
- [StatSoft 2004] StatSoft, Inc. *Electronic Statistics Textbook*. Tulsa, StatSoft. <http://www.statsoft.com/textbook/stathome.html>, 2004.
- [Stolfo & Chan 1998] Stolfo, S.; Chan P. "Towards scalable learning with non-uniform class and cost distributions" En *Proceedings of the Fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998.
- [Stolfo et al. 1997] Stolfo, S.J.; Prodromidis, A.L.; Tselepis, S.; Lee, W.; Fan, D.W.; Chan, P.K. "JAM: Java Agents for Meta-Learning over Distributed Databases", in *Knowledge Discovery and Data Mining*, p. 74-81, 1997.
- [Sutton & Barto 1998] Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.
- [Sycara et al. 1992] Sycara, K.; Guttal, R.; Koning, J.; Narasimhan, S.; Navinchandra, D. "CADET: A case-based synthesis tool for engineering design". *International Journal of Expert Systems*, 4(2), 157-188, 1992.
- [Tenenhaus 1998] Tenenhaus, M. *La régression PLS*. Ed. Technip, Paris, 1998.
- [Tettamanzi & Tomassini 2001] Tettamanzi, A.; Tomassini, M. *Soft Computing. Integrating Evolutionary, Neural and Fuzzy Systems*. Springer, 2001.
- [The R Development Core Team 2003] The R Development Core Team. *R: 1.6.2. A language and Environment*. <http://cran.r-project.org/>.
- [Thornton 1997] Thornton, C. "Separability is a learner's best friend". In J.A. Bullinaria, D.W. Glasspool, and G. Houghton, editors, *Proceedings of the Fourth Neural Computation and Psychology Workshop: Connectionist Representations*, pages 40-47. Springer-Verlag, 1997.
- [Thornton 2000] Thornton, C. *Truth from Trash. How Learning Makes Sense*, The MIT Press (A Bradford Book), 2000.
- [Thuraisingham 1999] Thuraisingham, B. *Data Mining. Technologies, Techniques, Tools, and Trends*, CRC Press LLC, 1999.
- [Ting & Low 1997] Ting, K.M.; Low, B.T. "Model combination in the multiple-data-base scenario", in *Proceedings of the European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, vol. 1224, pp. 250-255, 1997.
- [Toivonen 1996] Toivonen, H. "Sampling large databases for association rules", in *Proceedings of International Conference on Very Large Data Bases*, pp. 134-145, 1996.
- [Trueblood & Lovett 2001] Trueblood, R.P.; Lovett, J.N. "Data Mining and Statistical Analysis Using SQL" Apress 2001.
- [Trujillo et al. 2001] Trujillo, J.C.; Palomar, M.; Gómez, J. "Designing data warehouses with OO conceptual models" *IEEE Computer*. 34(12). pp. 66-75. 2001.
- [Tryfona et al. 1999] Tryfona, N.; Busborg, F.; Christiansen, J. "Star ER: a conceptual model for data warehouse design" *International Workshop on Data Warehousing and OLAP (DOLAP99)*. 1999.

- [Tsukimoto et al. 2002] Tsukimoto, H.; Kakimoto, M.; Morita, C.; Kikuchi Y. "Rule Discovery from fMRI Brain Images by Logical Regression Analysis" *Progress in Discovery Science 2002, Lecture Notes in Computer Science*, 232-245. 2002.
- [Tukey 1977] Tukey, J.W. *Exploratory Data Analysis*, Addison-Wesley, 1977.
- [Turksen 1998] Turksen, I.B. "Fuzzy data mining and expert system development". In *Proceedings IEEE International Conference on Systems, Man and Cybernetics*: 2057-2061, 1998.
- [Turney 2000] Turney, P. "Types of Cost in Inductive Concept Learning" *Proceedings Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, 15-21, 2000.
- [Two Crows Corporation 1999] Two Crows Corporation. *Introduction to Data Mining and Knowledge Discovery*, 1999.
- [Utgoff 1989] Utgoff, P.E. "Perceptron Trees: a case study in hybrid concept representations" *Connection Science*, 1(4), 377-391, 1989.
- [Utgoff et al. 1997] Utgoff, P.E.; Berkman, N.C.; Clouse, J.A. "Decision Tree Induction Based on Efficient Tree Restructuring" *Machine Learning*, October 1997.
- [Vafaie & De Jong 1998] Vafaie, H.; De Jong, K. "Evolutionary feature space transformation" In, Liu, H.; Motoda, H. (Eds.) *Feature extraction, Construction and Selection: a data mining perspective*: 307-323, 1998.
- [Valiant 1984] Valiant, L. G. "A theory of the learnable," *Communications of the ACM*, 27(11), pp. 1134- 1142, 1984.
- [van Rijsbergen 1979] Van Rijsbergen, C. *Information Retrieval*, Dept. of Computer Science. University of Glasgow, 2nd edition, 1979. Disponible en <http://www.dcs.gla.ac.uk/Keith/Preface.html>, 1979.
- [Vapnik 1995] Vapnik, V. *The Nature of Statistical Learning Theory*, Springer-Verlag, New York. 1995.
- [Vapnik 1998] Vapnik, V. *Statistical Learning Theory*. Wiley, 1998.
- [Vapnik 1999] Vapnik, V. "An Overview of Statistical Learning Theory". *IEEE Transactions on Neural Networks*, Vol. 10, No. 5, pp. 988-999, 1999.
- [Wahba 1990] Wahba, G. *Spline Models for Observational Data*. SIAM, Philadelphia, 1990.
- [Wand & Jones 1995] Wand, M.P.; M.C. Jones. *Kernel Smoothing*, Chapman and Hall, 1995.
- [Wang & Mendel 1992] Wang, L. X.; Mendel, J. M. "Generating fuzzy rules by examples" *IEEE Transactions on System, Man and Cybernetics*, 22 (6): 1414-1427, 1992.
- [Washio & Motoda 2003] Washio, T.; Motoda, H. "State of the Art of Graph-based Data Mining", *SIGKDD Explorations*, 5(1): 59-68, 2003.
- [Widrow & Hoff, 1960] Widrow, B. and Hoff, M. E. "Adaptive switching circuits" in *IRE WESCON*, pages 96-104, New York. Convention Record. 1960.
- [Wilks 1997] Wilks, Y. "Information Extraction as a core language technology", *Information Extraction*, Lecture Notes in Computer Science, vol. 1299, pp: 1-9, 1997.
- [Witten & Frank 2000] Clark, P.; Boswell, R. Data Mining. *Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, 2000.
- [Wold & Ketaneh 1996] Wold, H.; Ketaneh, N. *SIMCA-P. Version 3.0*. Umetri AB, Umea, 1996.
- [Wold 1985] Wold, S. "Partial Least Squares", *Encyclopedia of Statistical Sciences*, Vol 6, pp 581-591, Kotz S. and Johnson N.L. eds. John Wiley, 1985.
- [Wolpert & Macready 1997] Wolpert, D. Macready, W. "No free lunch theorems for search" SFI-TR-95-02-010, The Santa Fe Institute, 1995. Publicado como "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, 67-82, 1997.
- [Wolpert 1992] Wolpert, D.H. "Stacked Generalization" *Neural Networks*, Vol 5, pp 241-259, 1992.
- [Wong & Leung 2000] Wong, M.L.; Leung, K.S. *Data Mining Using Grammar-Based Genetic Programming and Applications*. Kluwer, 2000.
- [Wong 1999] Wong, P.C. "Visual Data Mining", Special Issue of *IEEE Computer Graphics and Applications*, Sep/Oct 1999, pp. 20-46.

- [Yamanishi 1997] Yamanishi, K. "Distributed cooperative bayesian learning strategies", in *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pp. 250-262, 1997.
- [Yang & Chute 1994] Yang, Y.; Chute, C.G. "An example-based mapping method for text categorization and retrieval", *ACM Transactions on Information Systems*, 12(3): 252-277, 1994.
- [Yang & Honavar 1998] Yang, J.; Honavar, V. "Feature subset selection using a genetic algorithm", *IEEE Intelligent Systems* 13: 44-49, 1998.
- [Yang 1994] Yang, Y. "Expert Network: Effective and efficient learning from human decisions in text categorization and retrieval", in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 13-22, 1994.
- [Zadeh 1965] Zadeh, L.A. "Fuzzy sets", *Information Control* 8: 338-353, 1965.
- [Zadeh 1975] Zadeh, L.A. "The concept of a linguistic variable and its applications to approximate reasoning, Parts I, II, III", *Information Sciences* 8-9: 199-249, 301-357, 43-80, 1975.
- [Zaiâne et al. 1998] Zaiâne O.R.; Han J.; Li Z.; Chee S.; Chiang J. "MultiMediaMiner: A System Prototype for Multimedia Data Mining". In *Proc. SIGMOD Conference* 581-583. 1998.
- [Zhang & Zhang 2002] Zhang, C.; Zhang, S. *Association Rule Mining. Models and Algorithms*, Lecture Notes in Artificial Intelligence, vol. 2307, Springer, 2002.
- [Zhang et al. 1994] Zhang, N. L.; Qi, R.; Poole, D "A computational theory of decision networks" *International Journal of Approximate Reasoning*, 11(2):83-158, 1994.
- [Zhang et al. 1996] Zhang, T.; Ramakrishnan, R.; Livny, M. "BIRCH: An Efficient Data Clustering Method for Very Large Databases" in *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data*, pp. 103-114, Montreal, Canada, 1996.
- [Zhu & Hastie 2003] Zhu, M.; Hastie, T. "Feature extraction for nonparametric discriminant analysis". *J. Comput. Graph. Statist.* 12 (1): 101-120, 2003.

ÍNDICE ANALÍTICO

#

0-subsunción, 308-309, 314, 326

A

ACP. *Véase* análisis de componentes principales

AdaBoost, 378, 380, 489, 497

Adaline. *Véase* regla Adaline adaptación

función de, 35, 386-391, 393-402, 410-411, 415, 448

medida, 386, 388, 394

medida de, 35, 386, 450

agrupamiento jerárquico, 143, 421, 436, 439, 458

algoritmo Apriori, 146, 148, 240-241, 248, 253-254, 256, 319, 612-613

algoritmo B, 269, 273, 490

algoritmo de retropropagación, 147, 332-333, 335-336, 352

algoritmo de Wang y Mendel, 407-409

algoritmo EM (Expectation Maximization), 276

algoritmo K2, 269, 274

algoritmo Naïve Bayes, 258, 260-262, 271, 273, 275, 278-279

algoritmo PAM, 455, 458, 530

algoritmo TAN, 272

algoritmos evolutivos, 34, 135, 383-384, 386-387, 389-390, 399-400, 409, 417-418, 450, 629

algoritmos genéticos, 27, 35, 148, 384, 387, 389-391, 393-396, 398-402, 410, 416-417, 419, 448-450, 617, 623

alislado, 116, 203, 216

almacén de datos, XIII, XIV, XVI, XVIII, 4, 10, 14-15, 19, 21-22, 41, 43-52, 54-63, 65, 67-69, 97, 101, 131-132, 149, 514, 557, 575, 583,

587-590, 593-594, 597, 601-602, 605, 615-616, 618, 640, 648

arquitectura de, 49

carga de, 59

granularidad de, 49

modelo multidimensional, 49-50, 52

operadores de. *Véase* operadores OLAP

alta dimensionalidad, 5, 30, 79, 116, 118, 153, 163, 227, 236, 354, 364, 380,

399, 445, 554, 568, 605

análisis correlacional, 12, 120

análisis de componentes principales, 65-67, 79-83, 93, 118, 122, 124, 142,

184, 189, 199, 209, 343-346, 355,

376, 401, 604, 629

análisis de correspondencias, 65-66, 90, 122-123

análisis de residuos. *Véase* residuos

análisis de varianza. *Véase* ANOVA

análisis discriminante, 86, 148, 203, 208-211, 213, 230, 233, 496

lineal, 86

análisis exploratorio de datos. *Véase* exploratory data analysis

análisis factorial, 65-66, 79, 82-83, 148

análisis inteligente de datos, 418

análisis multivariante, XV, 41, 65-66, 79, 82, 93-94, 121-122

análisis por modelo lineal, 121

análisis ROC, 15, 39, 144, 202, 211, 469-471, 515, 519, 601

anómalos, 77, 144, 189, 423, *Véase* también valor anómalo

ANOVA, 92, 122, 178

anytime, 154

aprendizaje

de Hebb, 327, 343-347, 639

inductivo, 38, 146, 148-149, 161, 307, 310, 553

no supervisado, 144, 252, 327, 330, 343, 347, 351, 377, 421

supervisado, 144, 216, 252, 317-318, 327, 330, 332, 341, 343, 351, 367, 377, 446-447, 537-538, 541-542, 604

aprendizaje PAC (Probabilistic

Approximate Correct), 153-154

aprendizaje por consultas, 150

Apriori. *Véase* algoritmo Apriori

AQ, 289, 297

árboles de decisión, 5, 12, 15, 19, 25, 27, 30-32, 75, 78, 88, 90, 93, 99-100,

117, 123, 125, 135, 139, 146, 148,

162, 258, 260, 281-285, 287-301,

303, 316, 322, 373, 389, 398, 406,

445, 474, 479, 486, 488, 491, 495-

497, 499, 502, 505-507, 512-514,

521, 529, 554, 602, 604, 609-614,

616-617, 619-621, 623

Area Under the ROC Curve, 287, 470-472, 515, 636

asociación. *Véase* reglas de asociación

atípico. *Véase* valor:anómalo

atributos relevantes, 99, 445, 457, 587

aumento de dimensionalidad. *Véase* dimensionalidad:aumento

AutoClass, 622

autocorrelación, 533-535

autoridades, 551, 560-561

B

bagging, 115, 298, 488-489, 491-492,

494, 497-498, 505, 604, 613, 619

bag-of-words. *Véase* vector de palabras

base de datos

distribuida, 603

documental, 11

espacial, 11, 525-528, 530-531

inductiva, 316, 318-319

multimedia, 11, 539

relacional, XVI, 9-11, 43-44, 55-56,

127, 161, 302-303, 306, 313-314,

317, 407, 547, 557, 592, 603, 616,

631

temporal, 11

transaccional, XIV, 44-48, 56, 59-61,

63, 575, 590

Bayes. *Véase* teorema de Bayes, *Véase* también algoritmo Naïve Bayes

B-course, 279

bias, 138, 310-311, 313-314, 325-326,

328, 354

binarización, 145, 312, 325, 376, 493,

Véase también numerización

bodega de datos. *Véase* almacén de

datos

bolsa de palabras. *Véase* vector de

palabras

boosting, 298, 487, 489-492, 497-499, 612-613, 619, 623, 630, 634, 637, 641, 643, 646-647
bootstrapping, 37, 115, 466, 477, 488
búsqueda de proyecciones exploratorias, EPP (Exploratory Projection Pursuit), 82, 345-346

C

C4.5, 125, 146, 148, 262-263, 275, 284-285, 287, 291-292, 294-295, 297, 316, 323, 373-374, 400, 445-446, 491, 497, 505, 611, 613, 619, 623, 639, 646
C5.0. *Véase* See5
calidad de datos, 23, 60, 66-67, 76, 153, 575, 582, 584, 587
campañas de marketing
diseño de, 516
capa oculta, 33, 330-332, 336-337, 339, 341, 343
características
creación de, 89, 401
vector de, 541
carga de un almacén de datos. *Véase* también sistema ETL
CART, 25, 146, 148, 284, 287, 293-295, 297, 299, 323, 499-502, 591, 611, 616, 620-621
CART-LC, 295
case-based reasoning, 149
categorización, 140, 145, 283, 368, 370-372, 476, 492, 539-540, 549, 552, 554, 568
de textos, 368, 371, 476, 540, 552, 554
CBR. *Véase* case-based reasoning centrado, 93
centroide, 207, 426, 429, 438
CLARA, 455, 458, 530
clasificación suave, 77, 140-141, 196, 202, 472-473, 516
clasificador bayesiano naïve. *Véase* algoritmo Naïve Bayes
clasificador suave. *Véase* también estimador de probabilidades
CLAUDIEN, 317, 325
Clementine, XIV, XV, 83, 132, 212, 252-255, 297-299, 337, 342, 352, 434, 497, 499-501, 506, 508, 591, 611-613
clustering. *Véase* agrupamiento
CN2, 146, 148, 289, 297, 326, 611, 632
cobertura, 27, 37-38, 113, 148, 237-241, 243-244, 246-248, 250-251, 253-255, 283, 287-291, 297, 311, 313, 323, 326, 371, 396, 413, 482, 565
coeficiente
de aprendizaje, 331, 333, 338-339, 345-346, 349
de autocorrelación, 534-535
de correlación, 26, 179, 247, 423
de determinación, 179, 181, 220

de elasticidad, 169
VIF. *Véase* Variance Inflation Factor (VIF)
colinealidad, 181-184, 192, 194, 209
combinación de modelos, 160, 459, 485-486, 500, *Véase* también métodos multicasificadores
Common Warehouse Metadata, 58, 131
componentes principales. *Véase* análisis de componentes principales
comprendibilidad de modelos, 6, 99-100, 137, 161, 482, 504
computación evolutiva, 34, 384, 386, 417-419, *Véase* también algoritmos evolutivos
operadores, 388-391, 393, 410, 450, *Véase* también mutación, *Véase* también cruce
computación flexible, 147, 383-384, 418
concentradores, 551, 561-562
confianza, 19, 31, 38, 99, 116, 127, 145, 150, 165, 182, 201, 238-244, 246-248, 253-255, 323-324, 372, 396, 403, 407, 409, 411-413, 415-417, 466-467, 477, 482, 535, 557, 565
confusión. *Véase* matriz de confusión
conjunto de entrenamiento, 33, 35-37, 150, 180, 202, 210, 222, 262, 272, 275, 289, 336, 340-342, 424, 430, 434, 447, 461-462, 464, 466, 475, 487-489, 492, 602
conjunto de ítems frecuentes, 32, 240, 319
conjunto de prueba, 35-37, 115, 150, 210, 276, 336, 342, 368, 371, 434, 464, 466, 468, 481
conjunto de test. *Véase* conjunto de prueba
conjunto de validación, 36, 371
conjunto difuso. *Véase* lógica difusa
conocimiento previo, 22, 39, 98-102, 125, 127, 131-132, 138, 153, 157, 161-162, 307, 406, 482, 583, 586-587, 606
contingencia
matriz de, 436
matriz de. *Véase* también matriz de confusión
tabla de, 122, 146, 244-245, 371-372, 475
correlación, 26, 83-84, 120, 125, 165, 174, 181, 185, 192, 199, 207, 210, 247, 296, 344, 368, 500, 533-535, *Véase* también análisis correlacional
CRISP-DM, 131, 571, 581, 583, 585
criterio de parada, 289, 291, 490, 498, 505, 566
criterio de partición, 120, 284, 287, 293, 297, 479, 505, 514
CRM (Customer Relationship Management), 417, 516, 579, 588-589, 594, 618
cromosoma, 388-390, 395-396, 399, 401, 413
cruce, 34, 386-389, 392, 394, 396, 398, 401, 414, 450

D

Darwin. *Véase* Oracle Data Mining Suite
data warehouse. *Véase* almacén de datos
DataEngine, 418
dataguide, 638
datamart, 51-52, 56-58, 107, 110
datos

calidad de. *Véase* calidad de datos
integración de, 60, 582
muestreo de, 236, 242, 419, 605, 616, *Véase* también muestreo
reconocimiento de, 102
selección de, 66, 112, 582, 592, *Véase* también selección de características
transformación de, 60, 63, 78, 351, 401, 483, 613

DB2 Intelligent Miner, XIV, 104, 568, 591, 617-618, 639

DBMiner, 591, 616-617, 638

dendrograma, 440, 509, *Véase* también agrupamiento jerárquico

dependencia funcional, 108, 119, 143, 237, 247, 263, 314, 325, 391, 393, 407, *Véase* también reglas de dependencias

desviación, 70-71, 81, 83, 94, 123, 181, 187, 195-196, 198, 201, 210, 235, 261, 293, 340, 396, 627

diagrama de caja o de bigotes, 72

DIANA, 458

difuso. *Véase* lógica difusa

dimensión VC, 154, 159

dimensionalidad
aumento de, 67, 85, *Véase* también núcleo
maldición de la. *Véase* maldición de la dimensionalidad
reducción de, 67, 79, 82-83, 112, 117, 124, 330, 344, 351

Discrete Wavelet Transform, 536, 541

discretización, 24, 65, 67, 78, 89-92, 120, 162, 236, 243, 253, 279, 391-392, 395-396, 483, 613

discriminación

cuadrática, 206

función de Fisher. *Véase*

discriminante de Fisher

lineal, 206-207

no paramétrica, 231, 233

paramétrica, 205, 229

discriminante de Fisher, 28, 177, 195, 208-213, 355, 376, 421, 425-427, 440, 540, 636

distancia, 19, 34, 85, 93-95, 123, 144, 147, 153, 157, 159, 163, 207, 225, 246, 303, 315, 320, 322, 361, 422-425, 429, 432-433, 435, 438-439, 444, 455, 481, 530, 541, 561, *Véase*

también métodos basados en distancia
aproximación relacional, 320
de Mahalanobis, 207, 423, 444
de Chebychev, 423
de enlace, 438, 440
de Manhattan, 423
del coseno, 423
euclídea, 34, 422, 429-430, 432-433, 449, 535
distribución
binomial, 116, 195, 197
de Poisson, 123, 195
del error, 195
gamma, 195
normal, 94, 174, 187, 194-195, 205, 235, 261-262, 466
DMQ, 127
Document Object Model, 559, *Véase también* XML
Document Type Definition, 510, 512
drill, 48, 52-54, 57, 110, 619
DSS. *Véase* sistemas para la toma de decisión
DTD. *Véase* Document Type Definition

E

Easy NN-plus, 622
EDA. *Véase* exploratory data analysis
eigenvalues, 80-81, 84, 199
eigenvectors, 80, 82
EIS. *Véase* Executive Information Systems
ELVIRA, 275, 278, 635
ensemble methods. *Véase* métodos multiclásificadores
Enterprise Miner, XIV
entropía, 90, 287, 400, 480, 483
EPP. *Véase* búsqueda de proyecciones exploratorias
ERP (Enterprise Resource Planning), 579, 588-589
error cuadrático medio, 19, 26, 38, 99, 150, 159, 167, 179-180, 183, 222, 293, 332, 340-341, 476, 584
error cuadrático relativo, 476
error esperado, 286, 291, 298
escalabilidad, 15, 297, 326, 395, 399, 419, 597, 601-602, 605
escalado
multidimensional, 65-66, 94-95
softmax o sigmoidal, 77, 93-94
espacio de características, 347, 355, 357, 361-367, 376-377
estadística, XIII, XIV, XV, XVI, 3, 5, 15, 27, 86, 92, 104, 135, 139, 146, 162, 165-166, 203, 212, 221, 236, 245, 263, 344, 353, 383, 428, 461, 535, 540, 575, 592-593, 598, 606, 609, 621, 623, *Véase también* modelización estadística
estadístico PRESS. *Véase* PRESS

estimador de probabilidades, 140-141, 202, 294, 472-473, 492
ETL. *Véase* sistema ETL
evaluación, 36-37, 150, 459, 461-462, 612, *Véase también* subajuste, *Véase también* sobreajuste
análisis ROC. *Véase* análisis ROC basada en costes, 462, 467, 472
bootstrapping. *Véase* bootstrapping conjunto de entrenamiento. *Véase* conjunto de entrenamiento
conjunto de prueba. *Véase* conjunto de prueba
de la clasificación, 462
de la regresión, 476
del agrupamiento, 480
reglas de asociación, 481
validación cruzada. *Véase* validación cruzada
executive information system, 46, 62, 102, 588-589, 640
expectation maximization. *Véase* algoritmo EM (Expectation Maximization)
explanation-based learning, 149
exploratory data analysis, 102, 630, 649
expresividad de modelos, 116, 137-138, 153, 155-157, 159-163, 258, 285, 312, 315, 317, 320, 341, 403, 405, 443
Extensible Markup Language (XML), XV, XVII, 59, 131, 317, 325, 510, 512, 523, 525, 547, 550-551, 559, 590, 617
extracción de información, 540, 548

F

factorización, 82
FFOIL, 289, 326
Fisher. *Véase* discriminante de Fisher fitness. *Véase* medida de adaptación FOIL, 289, 310, 323, 326, 646 forest, 487 funciones de activación, 332-334 funciones de densidad, 194, 204, 206, 225, 229-230, 232, 440-442, 480 funciones de puntuación, 150

G

GainRatio, 287, 297
generalización menos general, 310, 326
GID3, 287
GOLEM, 310, 326
gráfica de dispersión, 72-73

H

Hebb. *Véase* aprendizaje de Hebb
herramientas de minería de datos. *Véase* sistemas de minería de datos

hiperplano, 158, 160, 188, 354-355, 357-360, 362, 364, 378, 426
hipótesis MAP, 259-260, 478
histograma, 15, 65-67, 70-72, 77, 103, 122, 187-188, 193, 229, 612, 616
HOLAP, 55
hub. *Véase* concentradores
HUGIN, 279
Hyperion Enterprise, 56

I

IBM Intelligent Miner for Text, 568
ID3, 146, 148, 262-263, 275, 284-285, 297, 529, 611
ILP. *Véase* programación lógica inductiva
incremental, 116, 149, 240, 242, 292, 333, 377-378, 454, 520-521, 602
incrementalidad, 520, 602, *Véase también* incremental
IND, 295, 297
inducción. *Véase* aprendizaje inductivo
Informix, 56, 616
inteligencia artificial, XIII, 15, 257, 263, 383, 455, 547
Intelligent Miner. *Véase* DB2 Intelligent Miner
interfaces visuales, 103, 131-133

J

JBNC, 275, 279
jerarquía de conceptos, 247-248
ji-cuadrado, 123, 245-246, 254, 293

K

K medias, 147, 341, 421, 428, 432, 434-436, 444, 448, 455, 458, 494, 509, 610, 612-613, 616, 620
K vecinos. *Véase* vecinos más próximos
Kepler, 591, 614-615
kernel. *Véase* núcleo
Kohonen. *Véase* mapa de Kohonen

L

Laplace, corrección de, 116-117, 261-262, 294, 473
lenguajes de consulta, 10, 41, 44, 97, 126-127, 130, 316-317, 605
lgg. *Véase* generalización menos general LIBSVM, 363, 378-379
limpieza de datos, 23, 43, 67, 103, 144, 582, 587, 605
link distance. *Véase* distancia de enlace
LINUS, 89, 304, 315-316, 642
LISP, 161
lógica difusa, 35, 383-384, 403, 405-407, 409, 417-419, 623

LVQ (Learning Vector Quantization), 155, 347, 421, 446-448, 458

M

maldición de la dimensionalidad (curse of dimensionality), 79, 85, 153, 226-227, 232-233, 235

mapa auto-organizado de Kohonen.

Véase mapa de Kohonen

mapa auto-organizativo de Kohonen.

Véase mapa de Kohonen

mapa de características, 347

mapa de Kohonen, 348, 350, 352, 421, 428-430, 432-434, 458

máquinas de vectores soporte, XVIII, 28, 86, 135, 148, 152, 320, 353-367, 368-382, 406, 476, 554, 610, 613, 617

margen blando, 356, 364-367, 370, 373, 376-377

margen máximo, 354-366

Markov. *Véase* modelos ocultos de

Markov

matriz de confusión, 39, 202, 210, 394, 468-470, 475, *Véase también* contingencia

matriz de costes, 39, 467-468, 515, 626 máxima verosimilitud, 195, 214, 234, 261, 267, 272

MDL. *Véase* Minimum Description Length

m-estimado, 117, 261, 294, 554

Metacube, 56

metadatos, 60-61, 131

meta-modelo, 485, 495-498, 501-502, 505

métodos anticipativos, 151, 424

métodos basados en casos, 135, 421

métodos basados en distancias, 440

métodos basados en refuerzo, 567

métodos bottom-up, 308-310, 326

métodos colaborativos, 555, 567

métodos de fusión, 485, 492-493

métodos de minería de datos, 23, 77-78,

93, 99, 107, 118, 132, 301, 504, 506, 541

comparación de, 161, 291, 491

métodos multiclasicadores, 485, 487-488, 491, 497-499, 502

métodos relacionales, 135, 301

métodos retardados, 151-152, 422, 424-425, 444, 455, 458

métodos top-down, 308-309, 313-314, 325-326, 641

métodos wrappers, 59, 125, 557

Microstrategy, 56

minería de datos distribuida, 601, 603

minería de datos espaciales, 523, 525-526, 531

minería de datos multimedia, 539, 541, 543

minería de datos relacional, 18, 107, 147, 303, 315, 317

minería de datos temporales, 526, 531

minería de grafos, 301, 317, 320

minería de hipertexto, 559

minería de marcado, 551, 557

minería de textos, 545, 551-552

minería web, 11, 303, 325, 371, 545-550,

569

de contenido, 545, 559

de estructura, 523, 559

de uso, 523, 551, 563

Mineset, 507, 591

mínimo global, 335

mínimo local, 295, 334-335, 341

Minimum Description Length, 91, 150, 287, 291, 477-479, 481, 636

Minitab, 200

MIS (Management Information

Systems), 310-311, 588

MLC++, 130-131, 458, 507, 591, 610-611

MOBAL, 311, 326

Model Inference System, 310-311, 588

modelización estadística

no paramétrica, 135, 213

paramétrica, 135, 165

modelo

aditivo, 227-228, 233

de dependencias, 264

descriptivo, 12, 257-258, 270, 462,

481, 486, 604

lineal, 121, 123, 168-169, 179, 189-

190, 216, 235, 293, 360

lineal generalizado, 235

predictivo, 12, 35-36, 486, 604

modelo multidimensional de datos, 49-50, 52

modelos ocultos de Markov, 537

MOLAP, 55-59

M-SQL, 127

muestreo, 15, 37, 65, 67, 97, 105, 112-

115, 153, 242, 399, 445, 466, 612,

619, *Véase también* submuestreo,

Véase también sobrealmuestreo

aleatorio estratificado, 114

aleatorio simple, 114

de grupos, 114

exhaustivo, 115

multiclasificador, 486-487, 491-492, 495

Multimedia Miner, 542

mutación, 386-389, 394, 396, 398, 401, 414, 450

N

Naïve Bayes. *Véase también* algoritmo

Naïve Bayes

navaja de Occam, 150, 477-479, 485

Neural Planner, 622

NeuroDiet, 622

neurona, 328-329, 334-335, 340, 343-344, 347-349

NeuroShell, 622-623

n-gramas, 372, 375, 537, 553

normalización

de rango, 93, *Véase también* escalado lineal, 93

núcleo, 28, 86, 147, 152, 160, 163, 213, 216-222, 224-226, 229-231, 234-235, 318-319, 354-356, 361-366, 368-370, 372-381, 444, 593

numerización, 65, 67, 78, 89, 92, 120, 143, 160, 162, 185-186, 191, 194, 247, 342, 380

O

Object Exchange Model, 557-559

objetivos de minería de datos, 21, 573, 582, 584, 587

objetivos de negocio, 102, 574-575, 577, 580, 582-584, 587

Occam. *Véase* navaja de Occam

OLAP, 4-5, 14, 22, 41, 43, 46-47, 55-58, 62-63, 66, 78, 97, 99, 104, 108-110, 126, 131, 514, 547, 587, 589, 592, 616-617, 619, 648

operadores. *Véase* operadores OLAP

OLE DB for Data Mining, 128-130

OLTP, 4, 21, 46-47

OMG, 58, 131

operadores genéticos, 388-391, 393, 410, 450

operadores OLAP, 52, 107, 111, 122, 132

Oracle Data Mining Suite, 591, 615

Oracle Discoverer, 56

Oracle Express, 56

outlier. *Véase* valor:anómalo

overfitting. *Véase* sobreajuste

oversampling. *Véase* sobrealmuestreo

P

PAM. *Véase* algoritmo PAM

parámetro de suavizado, 218-225, 230

patrones de navegación, 549, 563, 567

PCA. *Véase* análisis de componentes

principales

perceptrón, 82, 147, 158, 295, 327, 330-332, 337-338, 340, 342, 355, 376, 494, 611-612

personalización, 563, 566

PETs (Probability Estimation Trees), 294

pick&mix, 160

pivotamiento, 78, 107, 111

PMML. *Véase* Predictive Model

Markup Language

poda, 162, 289-294, 297, 445

PRediction Error Sum of Square. *Véase* PRESS

Predictive Model Markup Language, 129, 131, 503, 510, 512, 578, 590, 610, 612

preparación de datos, XVI

PRESS, 180-181, 192-193

principio MDL. *Véase* Minimum Description Length
privacidad, 579, 597, 599-600
probabilidad a priori, 204, 207, 209, 259-260
procesamiento analítico (en línea). *Véase* OLAP
procesamiento transaccional (en línea). *Véase* OLTP
proceso de extracción de conocimiento, 13, 19-22, 24, 35, 103, 549, 631, 636, 647
Progol, 313-314, 326, 645
programa de minería de datos, XVII, 571, 573-574, 576-581, 583, 585-587, 590-594
programación dinámica, 375
programación lógica, 27, 32, 107, 147, 161, 303-306, 317, 324
programación lógica inductiva, 27, 32, 107, 147, 161, 301, 303-304, 306-307, 309-319, 324-326, 398, 479, 505
programación lógico-funcional, 324
proposicionalización, 315
protección de datos, 62, 599

R

Radial Basis Functions, 339-342, 352, 487, 494, 612, 619
Randomisation, 488, 491
razonamiento basado en casos. *Véase* case-based reasoning
RBF. *Véase* Radial Basis Functions
realimentación, XVIII, 344, 346, 521, 581
recuperación de información, 11, 14, 48, 263, 353, 367, 373, 407, 475, 545-546, 548-549, 551, 553, 557
recursividad, 156, 311-312
red bayesiana, 146, 162, 239, 257-258, 260, 263, 265-271, 273-274, 276, 278-279, 301, 320, 506-508, 513
red de Kohonen, 82-83, 85, 147-148, 431, 612
red neuronal artificial, 5, 12, 18-19, 27, 32-33, 35, 77, 82, 87, 99, 135, 139, 146-148, 155, 162, 216, 260, 281, 298, 301, 327, 329-330, 336-338, 341-343, 351-352, 355, 359, 367, 384, 406, 418, 483, 488, 491, 495-496, 504-505, 510, 512, 554, 609-610, 612, 614, 616, 619-622, 640
regla Adaline, 330-332, 335
reglas de asociación, 12, 19, 25-27, 32, 35, 37-38, 90, 92, 99-100, 108, 110-111, 119, 123, 127, 129, 135, 138-139, 143-145, 147-148, 150, 237-244, 246-250, 252-254, 256-257, 271, 288, 301, 303, 312-314, 317, 319, 323-324, 383-384, 391, 395-396, 398, 400, 406, 409-410, 412, 481-482, 506, 508, 512, 530, 538, 542, 548-

549, 556, 563, 568, 604, 610, 612-613, 620
multinivel, 247
secuenciales, 25, 27, 143, 249-250, 548
reglas de dependencias, 243
reglas difusas, 135, 383, 403-405, 407-411, 413, 415, 418, *Véase* también lógica difusa
regresión
con variables categóricas, 185
evaluación de modelos de, 476
lineal, 15, 26-28, 86-89, 92, 116-117, 121, 138, 143, 146, 148, 157, 168, 170, 174, 181, 185, 195-196, 209, 211, 213-214, 216-217, 219, 223, 227, 235, 293, 297, 455, 496, 505, 612, 616, 619
logística, 92, 99, 125, 141, 146, 148, 196, 198-199, 203, 210-212, 234-235, 499, 501-502
múltiple, 167, 209, 215, 224, 226-227, 233-235, 620
no paramétrica, 213, 215, 219, 224, 226, 230, 232-233
paramétrica, 223
simple, 167, 174, 215-216, 218, 226
regresores, 166, 173, 175-176, 178-179, 181-183, 190, 192, 200, 219, 224, 486-487
repositorio UCI, 38, 324, 467-468, 611, 625-627, 630
residuos, 170-172, 175, 177, 184, 187-190, 193-194
análisis, 187
resolución inversa, 310
ROC analysis. *Véase* análisis ROC
ROLAP, 55-59
roll, 48, 52-54, 57, 110, 122
R-project, paquete estadístico, 212, 227, 236, 458, 648
ruido, 151, 161-162, 175, 177, 184, 194, 196, 221, 289-292, 298, 328, 356, 364, 366, 418, 491

S

SAS Enterprise Miner, 104, 132, 585, 591, 618
SAS Text Miner, 568, 618
SCM (Supply Chain Management), 579, 588-589
scree diagram, 81, 84
See5, 25, 148, 297-300, 499-502, 591, 611-612, 623
segmentación. *Véase* agrupamiento
selección de características, 23, 29, 65-66, 78, 82, 100, 112, 117-118, 120, 123-124, 153, 181-183, 242, 263, 273, 275, 278-279, 294, 342, 376, 379, 400-403, 408, 582, 592, 602, 619

selección de datos. *Véase* también muestreo, *Véase* selección de características
selección de variables. *Véase* selección de características
separabilidad
geométrica, 158
geométrica radial, 158
lineal, 157-158, 160, 332, 354
sistema ETL, 59-61, 67
sistema ETL (Extraction, Transformation, Load), 59-61, 67
sistemas de información ejecutivos. *Véase* executive information systems
sistemas para la toma de decisión, XIII, XIV, XV, 3-4, 13, 15, 17, 21, 45-47, 238, 263, 503-504, 510, 512-514, 516, 575, 586, 588-589, 592
sistemas recomendadores, 567
SLIQ, 297, 643
smoothing. *Véase* alisado
SNNS, 351
sobreajuste, 155, 157, 160, 203, 206, 221, 263, 275, 289-290, 295, 330, 336, 355, 380, 463-464, 478, 487
sobrealmuestreo, 114-115, 469, 514
soft computing. *Véase* computación flexible
SOM. *Véase* mapa de Kohonen
SPAD, 199, 212
SPSS Clementine. *Véase* Clementine
SQL, 4, 10, 22, 44, 48, 52, 55, 57, 59, 70, 97, 107, 109, 126-131, 296, 323, 568, 588, 590, 612, 616-617, 619, 648
STATISTICA Data Miner, 619-620
subajuste, 156-157, 464
submuestreo, 114, 469, 514
suite de minería de datos, 417, 611
suma de cuadrados, 177-179, 193, 196
sumarización, 5, 26, 48, 50, 52, 65, 107-110, 142, 407, 547

T

TAN. *Véase* algoritmo TAN
tareas de minería de datos, 12, 19, 44, 103, 119, 297, 312, 317, 325, 384, 394, 422, 574, 577, 585, 587
técnicas estocásticas, 163
teorema de Bayes, 146, 259, 441, 443, 478
teorema de Gauss-Markov, 172, 180
teorema de Mercer, 364
TERTIUS, 314-315
TILDE, 322
tipificación, 94, 174
toma de decisiones, XIII, XIV, XV, XVII, 3-4, 13, 15, 17, 21, 45-47, 238, 263, 503-504, 510, 512-514, 516, 531, 575, 582, 586, 588, 592, *Véase* también sistemas para la toma de decisión

transformación de atributos. *Véase*

datos:transformación

transformación de datos. *Véase*

datos:transformación

transformación discreta de Fourier, 536

Trepan, 505

t-test, 477

V

UCI. *Véase* repositorio UCI

UML, 58, 131

underfitting. *Véase* subajuste

undersampling. *Véase* submuestreo

V

validación cruzada, XIV, 15, 36, 150, 180-181, 210, 222-224, 230, 368,

380, 464-466, 477, 496-498

validación simple, 36

valor

anómalo, 62, 65, 68, 76-78, 93, 142, 144, 162, 189, 423, 645

erróneo, 20, 77, 118, 151

faltante, 23, 74-76, 162, 184, 298

variables categóricas, 30, 165, 185-187, 190-191, 194

Variance Inflation Factor (VIF), 181-182, 191-192

varianza, XV, 75, 80-85, 93, 165, 171-172, 174-175, 177, 179-181, 183-185, 189-191, 199, 208, 215-216, 221, 224-225, 293, 345, 465, 535

vecinos más próximos, 34, 77, 147-148, 155, 216, 224, 316, 322, 421, 425, 427, 442-444, 446, 448, 455, 458, 481, 613-614, 616

vector de palabras, 372, 375, 552-554

vista minable, 10, 22, 55, 97-101, 103, 107-109, 111-112, 119, 125-129, 131, 138, 302, 304, 582, 602, 617

visualización

posterior, 103, 504, 506

previa, 20, 97, 103-104

W

WEKA, XIV, XV, 91, 124-125, 130-131, 212, 252-254, 262, 275, 278-279, 297, 315, 336, 352, 379, 445, 458, 497-499, 591, 609, 613-614

Winrosa, 418

X

Xelopes, 458, 591, 610

XML. *Véase* Extensible Markup Language

Y

Yale, 417

